ACL 2007

Proceedings of the Student Research Workshop

June 25–26, 2007

Prague, Czech Republic

Production and Manufacturing by
Omnipress
2600 Anderson Street
Madison, WI 53704
USA

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

Preface

On behalf of the Organizing Committee, we are pleased to present the proceedings of the Student Research Workshop held at the 45th Annual Meeting of the Association for Computational Linguistics (ACL) in Prague, Czech Republic, June 25–27, 2007. The Student Research Workshop is an established tradition at the Annual Meetings of the Association for Computational Linguistics and builds on the success of no less than 16 previous student sessions at ACL meetings.

Students in Computational Linguistics, Natural Language Processing and related fields were offered the possibility to present their work in a setting embedded in the main conference. The workshop plays an integral role in ACL's efforts to build and maintain a research community by investing in young researchers that will shape the field in the years to come. The workshop aimed at providing feedback from senior to beginner researchers. In the call for papers, we explicitly aimed at students in an early stage of their Ph.D. work. We felt that this group could gain the most benefit from this event, as the experts' feedback can still influence their research directions.

The Program Committee was compiled such that about half of the reviewers were students or young researchers, and the other half consisted of senior scientists. This mixture ensures that the scientific quality of reviews is high, while student-specific issues are well understood by the committee members. We are indebted to our 52 reviewers for their elaborate, thoughtful and high quality reviews, which will also be of great help to those students whose work could not be accepted for presentation.

We received 52 submissions from all over the world, of which 16 were accepted for presentation: 9 for oral presentation and 7 for poster presentation. The presentation format was assigned based on thoughts about how the work could be presented best, and does not indicate a quality difference among papers, which are all fixed to the same length of 6 pages.

This year's workshop features contributions from a wide range of topics. Various issues on grammar are dealt with in five papers: Richard Johansson uses logistic online learning for incremental dependency parsing, Nathan C. Sanders measures syntactic differences in British English, Elias Ponvert induces combinatory categorial grammars with genetic algorithms, Bart Cramer investigates limitations of current grammar induction techniques, and Aleksander Buczyński describes an implementation that combines partial parsing and morphosyntactic disambiguation.

Another five contributions can be subsumed under the scope of semantics: Radosław Moszczyński provides a classification of multi-word expressions especially for highly inflected languages, Paul Nulty classifies noun phrases along semantic properties using web counts and machine learning, Diarmuid Ó Séaghdha annotates and learns compound noun semantics, Kata Gábor and Enikő Héja cluster Hungarian verbs by complementation patterns, and Silke Scheible lays out foundations of a computational treatment of superlatives.

Research on dialects and different languages is carried out by three papers: Andrea Mulloni performs cognate prediction in a bilingual setting, Yves Scherrer presents adaptive measures to graphemic similarity for inducing dialect lexicons, and Jelena Prokić identifies linguistic structure in a quantitative analysis of Bulgarian dialects. For opinionated Chinese Information Retrieval, Taras Zagibalov examines the utility of various features. Structuring texts is the topic of two papers: Olena Medelyan

uses graph clustering to compute lexical chains, and Martina Naughton exploits structure for event discovery using the MDI algorithm.

Following the workshop tradition, a panel of senior researchers will take part in the presentation of papers, providing in-depth comments on the work of each author either immediately after the oral presentation or in front of the poster. We would like to thank the panelists in advance for fulfilling such an important role.

Many people contributed to the success of this year's Student Research Workshop. Apart from Program Committee members and panelists, we would like to thank the ACL conference organizers for involving us in their planning, the webmasters for swiftly handling update requests for the SRW page, the publication chair for providing us the facilities to compile this volume, and, most of all, the students for their hard work in preparing their submissions. Finally, we are grateful to the National Science Foundation for generously sponsoring our event: All student presenters received reimbursement of registration and accommodation as well as almost full coverage of travel costs.

The ACL 2007 Student Research Workshop Co-Chairs
Chris Biemann, Violeta Seretan, Ellen Riloff

Organizers

Chairs:

Chris Biemann, University of Leipzig, Germany
Violeta Seretan, University of Geneva, Switzerland

Faculty advisor:

Ellen Riloff, University of Utah, USA

Program Committee:

Laura Alonso i Alemany, Universidad de la República, Uruguay
and Universidad Nacional de Córdoba, Argentina
Galia Angelova, Bulgarian Academy of Sciences, Bulgaria
Timothy Baldwin, University of Melbourne, Australia
Raffaella Bernardi, Free University of Bozen-Bolzano, Italy
Stephan Bloehdorn, University of Karlsruhe, Germany
Gemma Boleda, Universitat Pompeu Fabra, Spain
Kalina Bontcheva, University of Sheffield, UK
Monojit Choudhury, Indian Institute of Technology, Kharagpur, India
Philipp Cimiano, University of Karlsruhe, Germany
Alexander Clark, Royal Holloway, University of London, UK
Gaël Harry Dias, University of Beira Interior, Portugal
Katrin Erk, University of Texas at Austin, USA
Stefan Evert, University of Osnabrück, Germany
Afsaneh Fazly, University of Toronto, Canada
Alexander Gelbukh, National Polytechnic Institute, Mexico
Alfio Gliozzo, ITC-irst, Trento, Italy
Yoav Goldberg, Ben-Gurion University of the Negev, Israel
Jean-Philippe Goldman, University of Geneva, Switzerland
Günther Görz, University of Erlangen, Germany
Iryna Gurevych, Darmstadt University of Technology, Germany
Catalina Hallett, The Open University, UK
Laura Hasler, University of Wolverhampton, UK
Janne Bondi Johannessen, University of Oslo, Norway
Philipp Koehn, University of Edinburgh, UK
Zornitsa Kozareva, University of Alicante, Spain
Chin-Yew Lin, Microsoft Research Asia, China
Berenike Loos, European Media Laboratory GmbH, Heidelberg, Germany
Bernardo Magnini, ITC-irst, Trento, Italy
Irina Matveeva, University of Chicago, USA
Rada Mihalcea, University of North Texas, USA
Andrea Mulloni, University of Wolverhampton, UK

Roberto Navigli, University of Roma “La Sapienza”, Italy
Malvina Nissim, University of Bologna, Italy
Joakim Nivre, Växjö University and Uppsala University, Sweden
Constantin Orasan, University of Wolverhampton, UK
Rainer Osswald, FernUniversität in Hagen, Germany
Sebastian Padó, Saarland University, Germany
Adam Przepiórkowski, Polish Academy of Sciences, Poland
Reinhard Rapp, Universitat Rovira i Virgili, Tarragona, Spain
Bettina Schrader, University of Osnabrück, Germany
Sabine Schulte im Walde, Universität Stuttgart, Germany
Serge Sharoff, University of Leeds, UK
Yihai Shen, The Hong Kong University of Science and Technology, Hong Kong
Anders Søgaard, University of Copenhagen, Denmark
Lucia Specia, University of São Paulo, Brasil
Joel Tetreault, University of Pittsburgh, USA
Reut Tsarfaty, University of Amsterdam, The Netherlands
Begoña Villada Moirón, University of Groningen, The Netherlands
Stephen Wan, Macquarie University, Australia
Janyce M. Wiebe, University of Pittsburgh, USA
Hans Friedrich Witschel, University of Leipzig, Germany
Bing Zhao, Carnegie Mellon University, USA

Table of Contents

Conference Program

Monday, June 25, 2007

14:45–16:35 Poster Session

Posters

Measuring Syntactic Difference in British English
Nathan C. Sanders

Inducing Combinatory Categorial Grammars with Genetic Algorithms
Elias Ponvert

An Implementation of Combined Partial Parser and Morphosyntactic Disambiguator
Aleksander Buczyński

A Practical Classification of Multiword Expressions
Radosław Moszczyński

Automatic Prediction of Cognate Orthography Using Support Vector Machines
Andrea Mulloni

Exploiting Structure for Event Discovery Using the MDI Algorithm
Martina Naughton

Kinds of Features for Chinese Opinionated Information Retrieval
Taras Zagibalov

Tuesday, June 26, 2007

9:15–9:25 Opening Remarks

Grammar and the Lexicon

09:25–09:50 *Limitations of Current Grammar Induction Algorithms*
Bart Cramer

09:50-10:15 *Logistic Online Learning Methods and Their Application to Incremental Dependency Parsing*
Richard Johansson

10:15-10:40 *Adaptive String Distance Measures for Bilingual Dialect Lexicon Induction*
Yves Scherrer

Quantitative and Formal Linguistics

14:30-14:55 *Identifying Linguistic Structure in a Quantitative Analysis of Dialect Pronunciation*
Jelena Prokić

14:55-15:20 *Towards a Computational Treatment of Superlatives*
Silke Scheible

Semantics

15:45-16:10 *Annotating and Learning Compound Noun Semantics*
Diarmuid Ó Séaghdha

16:10-16:35 *Semantic Classification of Noun Phrases Using Web Counts and Learning Algorithms*
Paul Nulty

16:35-17:00 *Computing Lexical Chains with Graph Clustering*
Olena Medelyan

17:00-17:25 *Clustering Hungarian Verbs on the Basis of Complementation Patterns*
Kata Gábor and Enikő Héja

Measuring Syntactic Difference in British English

Nathan C. Sanders
Department of Linguistics
Indiana University
Bloomington, IN 47405, USA
ncsander@indiana.edu

Abstract

Recent work by Nerbonne and Wiersma (2006) has provided a foundation for measuring syntactic differences between corpora. It uses part-of-speech trigrams as an approximation to syntactic structure, comparing the trigrams of two corpora for statistically significant differences.

This paper extends the method and its application. It extends the method by using leaf-path ancestors of Sampson (2000) instead of trigrams, which capture internal syntactic structure—every leaf in a parse tree records the path back to the root.

The corpus used for testing is the International Corpus of English, Great Britain (Nelson et al., 2002), which contains syntactically annotated speech of Great Britain. The speakers are grouped into geographical regions based on place of birth. This is different in both nature and number than previous experiments, which found differences between two groups of Norwegian L2 learners of English. We show that dialectal variation in eleven British regions from the ICE-GB is detectable by our algorithm, using both leaf-ancestor paths and trigrams.

1 Introduction

In the measurement of linguistic distance, older work such as Séguy (1973) was able to measure distance in most areas of linguistics, such as phonology, morphology, and syntax. The features used for comparison were hand-picked based on linguistic knowledge of the area being surveyed. These features, while probably lacking in completeness of coverage, certainly allowed a rough comparison of distance in all linguistic domains. In contrast, computational methods have focused on a single area of language. For example, a method for determining phonetic distance is given by Heeringa (2004). Heeringa and others have also done related work on phonological distance in Nerbonne and Heeringa (1997) and Gooskens and Heeringa (2004). A measure of syntactic distance is the obvious next step: Nerbonne and Wiersma (2006) provide one such method. This method approximates internal syntactic structure using vectors of part-of-speech trigrams. The trigram types can then be compared for statistically significant differences using a permutation test.

This study can be extended in a few ways. First, the trigram approximation works well, but it does not necessarily capture all the information of syntactic structure such as long-distance movement. Second, the experiments did not test data for geographical dialect variation, but compared two generations of Norwegian L2 learners of English, with differences between ages of initial acquisition.

We address these areas by using the syntactically annotated speech section of the International Corpus of English, Great Britain (ICE-GB) (Nelson et al., 2002), which provides a corpus with full syntactic annotations, one that can be divided into groups for comparison. The sentences of the corpus, being represented as parse trees rather than a vector of POS tags, are converted into a vector of leaf-ancestor paths, which were developed by Sampson (2000) to aid in parser evaluation by providing a way to compare gold-standard trees with parser output trees.

In this way, each sentence produces its own vec-

Proceedings of the ACL 2007 Student Research Workshop, pages 1–6,
Prague, June 2007.

tor of leaf-ancestor paths. Fortunately, the permutation test used by Nerbonne and Wiersma (2006) is already designed to normalize the effects of differing sentence length when combining POS trigrams into a single vector per region. The only change needed is the substitution of leaf-ancestor paths for trigrams.

The speakers in the ICE-GB are divided by place of birth into geographical regions of England based on the nine Government Office Regions, plus Scotland and Wales. The average region contains a little over 4,000 sentences and 40,000 words. This is less than the size of the Norwegian corpora, and leaf-ancestor paths are more complex than trigrams, meaning that the amount of data required for obtaining significance should increase. Testing on smaller corpora should quickly show whether corpus size can be reduced without losing the ability to detect differences.

Experimental results show that differences can be detected among the larger regions: as should be expected with a method that measures statistical significance, larger corpora allow easier detection of significance. The limit seems to be around 250,000 words for leaf-ancestor paths, and 100,000 words for POS trigrams, but more careful tests are needed to verify this. Comparisons to judgments of dialectologists have not yet been made. The comparison is difficult because of the difference in methodology and amount of detail in reporting. Dialectology tends to collect data from a few informants at each location and to provide a more complex account of relationship than the like/unlike judgments provided by permutation tests.

2 Methods

The methods used to implement the syntactic difference test come from two sources. The primary source is the syntactic comparison of Nerbonne and Wiersma (2006), which uses a permutation test, explained in Good (1995) and in particular for linguistic purposes in Kessler (2001). Their permutation test collects POS trigrams from a random subcorpus of sentences sampled from the combined corpora. The trigram frequencies are normalized to neutralize the effects of sentence length, then compared to the trigram frequencies of the complete corpora.

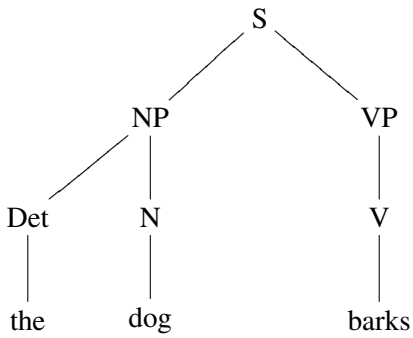
The principal difference between the work of Nerbonne and Wiersma (2006) and ours is the use of leaf-ancestor paths. Leaf-ancestor paths were developed by Sampson (2000) for estimating parser performance by providing a measure of similarity of two trees, in particular a gold-standard tree and a machine-parsed tree. This distance is not used for our method, since for our purposes, it is enough that leaf-ancestor paths represent syntactic information, such as upper-level tree structure, more explicitly than trigrams.

The permutation test used by Nerbonne and Wiersma (2006) is independent of the type of item whose frequency is measured, treating the items as atomic symbols. Therefore, leaf-ancestor paths should do just as well as trigrams as long as they do not introduce any additional constraints on how they are generated from the corpus. Fortunately, this is not the case; Nerbonne and Wiersma (2006) generate $N - 2$ POS trigrams from each sentence of length N; we generate N leaf-ancestor paths from each parsed sentence in the corpus. Normalization is needed to account for the frequency differences caused by sentence length variation; it is presented below. Since the same number (minus two) of trigrams and leaf-ancestor paths are generated for each sentence the same normalization can be used for both methods.

2.1 Leaf-Ancestor Paths

Sampson's leaf-ancestor paths represent syntactic structure by aggregating nodes starting from each leaf and proceeding up to the root—for our experiment, the leaves are parts of speech. This maintains constant input from the lexical items of the sentence, while giving the parse tree some weight in the representation.

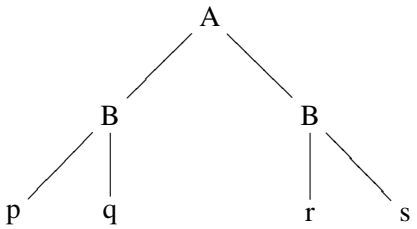
For example, the parse tree



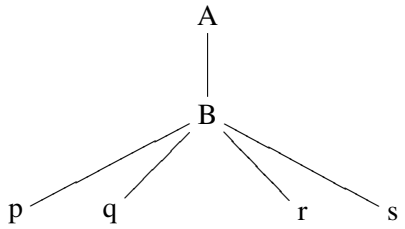
creates the following leaf-ancestor paths:

- S-NP-Det-The
- S-NP-N-dog
- S-VP-V-barks

There is one path for each word, and the root appears in all four. However, there can be ambiguities if some node happens to have identical siblings. Sampson gives the example of the two trees



and



which would both produce

- A-B-p
- A-B-q
- A-B-r
- A-B-s

There is no way to tell from the paths which leaves belong to which B node in the first tree, and there is no way to tell the paths of the two trees apart despite their different structure. To avoid this ambiguity, Sampson uses a bracketing system; brackets are inserted at appropriate points to produce

- [A-B-p
- A-B]-q
- A-[B-r
- A]-B-s

and

- [A-B-p
- A-B-q
- A-B-r
- A]-B-s

Left and right brackets are inserted: at most one in every path. A left bracket is inserted in a path containing a leaf that is a leftmost sibling and a right bracket is inserted in a path containing a leaf that is a rightmost sibling. The bracket is inserted at the highest node for which the leaf is leftmost or rightmost.

It is a good exercise to derive the bracketing of the previous two trees in detail. In the first tree, with two B siblings, the first path is A-B-p. Since p is a leftmost child, a left bracket must be inserted, at the root in this case. The resulting path is [A-B-p. The next leaf, q, is rightmost, so a right bracket must be inserted. The highest node for which it is rightmost is B, because the rightmost leaf of A is s. The resulting path is A-B]-q. Contrast this with the path for q in the second tree; here q is not rightmost, so no bracket is inserted and the resulting path is A-B-q. r is in almost the same position as q, but reversed: it is the leftmost, and the right B is the highest node for which it is the leftmost, producing A-[B-r. Finally, since s is the rightmost leaf of the entire sentence, the right bracket appears after A: A]-B-s.

At this point, the alert reader will have noticed that both a left bracket and right bracket can be inserted for a leaf with no siblings since it is both leftmost and rightmost. That is, a path with two brackets on the same node could be produced: A-[B]-c. Because of this redundancy, single children are excluded by the bracket markup algorithm. There is still no ambiguity between two single leaves and a single node with two leaves because only the second case will receive brackets.

2.2 Permutation Significance Test

With the paths of each sentence generated from the corpus, then sorted by type into vectors, we now try to determine whether the paths of one region occur in significantly different numbers from the paths of another region. To do this, we calculate some measure to characterize the difference between two vectors as a single number. Kessler (2001) creates a

simple measure called the RECURRENCE metric (R hereafter), which is simply the sum of absolute differences of all path token counts c_{ai} from the first corpus A and c_{bi} from the second corpus B.

$$R = \Sigma_i |c_{ai} - \bar{c_i}| \text{ where } \bar{c_i} = \frac{c_{ai} + c_{bi}}{2}$$

However, to find out if the value of R is significant, we must use a permutation test with a Monte Carlo technique described by Good (1995), following closely the same usage by Nerbonne and Wiersma (2006). The intuition behind the technique is to compare the R of the two corpora with the R of two random subsets of the combined corpora. If the random subsets' Rs are greater than the R of the two actual corpora more than p percent of the time, then we can reject the null hypothesis that the two were are actually drawn from the same corpus: that is, we can assume that the two corpora are different.

However, before the R values can be compared, the path counts in the random subsets must be normalized since not all paths will occur in every subset, and average sentence length will differ, causing relative path frequency to vary. There are two normalizations that must occur: normalization with respect to sentence length, and normalization with respect to other paths within a subset.

The first stage of normalization normalizes the counts for each path within the pair of vectors a and b. The purpose is to neutralize the difference in sentence length, in which longer sentences with more words cause paths to be relatively less frequent. Each count is converted to a frequency f

$$f = \frac{c}{N}$$

where c is either c_{ai} or c_{bi} from above and N is the length of the containing vector a or b. This produces two frequencies, f_{ai} and f_{bi}.Then the frequency is scaled back up to a redistributed count by the equation

$$\forall j \in a, b : c'_{ji} = \frac{f_{ji}(c_{ai} + c_{bi})}{f_{ai} + f_{bi}}$$

This will redistribute the total of a pair from a and b based on their relative frequencies. In other words, the total of each path type $c_{ai} + c_{bi}$ will remain the same, but the values of c_{ai} and c_{bi} will be balanced by their frequency within their respective vectors.

For example, assume that the two corpora have 10 sentences each, with a corpus a with only 40 words and another, b, with 100 words. This results in $N_a = 40$ and $N_b = 100$. Assume also that there is a path i that occurs in both: $c_{ai} = 8$ in a and $c_{bi} = 10$ in b. This means that the relative frequencies are $f_{ai} = 8/40 = 0.2$ and $f_{bi} = 10/100 = 0.1$. The first normalization will redistribute the total count (18) according to relative size of the frequencies. So

$$c'_{ai} = \frac{0.2(18)}{0.2 + 0.1} = 3.6/0.3 = 12$$

and

$$c'_{bi} = \frac{0.1(18)}{0.2 + 0.1} = 1.8/0.3 = 6$$

Now that 8 has been scaled to 12 and 10 to 6, the effect of sentence length has been neutralized. This reflects the intuition that something that occurs 8 of 40 times is more important than something that occurs 10 of 100 times.

The second normalization normalizes all values in both permutations with respect to each other. This is simple: find the average number of times each path appears, then divide each scaled count by it. This produces numbers whose average is 1.0 and whose values are multiples of the amount that they are greater than the average. The average path count is $N/2n$, where N is the number of path tokens in both the permutations and n is the number of path types. Division by two is necessary since we are multiplying counts from a single permutation by token counts from both permutations. Each type entry in the vector now becomes

$$\forall j \in a, b : s_{ji} = \frac{2nc'_{ji}}{N}$$

Starting from the previous example, this second normalization first finds the average. Assuming 5 unique paths (types) for a and 30 for b gives

$$n = 5 + 30 = 35$$

and

$$N = N_a + N_b = 40 + 100 = 140$$

Therefore, the average path type has $140/2(35) = 2$ tokens in a and b respectively. Dividing c'_{ai} and c'_{bi} by this average gives $s_{ai} = 6$ and $s_{bi} = 3$. In other words, s_{ai} has 6 times more tokens than the average path type.

Region	sentences	words
East England	855	10471
East Midlands	1944	16924
London	24836	244341
Northwest England	3219	27070
Northeast England	1012	10199
Scotland	2886	27198
Southeast England	11090	88915
Southwest England	939	7107
West Midlands	960	12670
Wales	2338	27911
Yorkshire	1427	19092

Table 1: Subcorpus size

Region	Significantly different ($p < 0.05$)
London	East Midlands, NW England SE England, Scotland
SE England	Scotland

Table 2: Significant differences, leaf-ancestor paths

Region	Significantly different ($p < 0.05$)
London	East Midlands, NW England, NE England, SE England, Scotland, Wales
SE England	London, East Midlands, NW England, Scotland
Scotland	London, SE England, Yorkshire

Table 3: Significant differences, POS trigrams

3 Experiment and Results

The experiment was run on the syntactically annotated part of the International Corpus of English, Great Britain corpus (ICE-GB). The syntactic annotation labels terminals with one of twenty parts of speech and internal nodes with a category and a function marker. Therefore, the leaf-ancestor paths each started at the root of the sentence and ended with a part of speech. For comparison to the experiment conducted by Nerbonne and Wiersma (2006), the experiment was also run with POS trigrams. Finally, a control experiment was conducted by comparing two permutations from the same corpus and ensuring that they were not significantly different.

ICE-GB reports the place of birth of each speaker, which is the best available approximation to which dialect a speaker uses. As a simple, objective partitioning, the speakers were divided into 11 geographical regions based on the 9 Government Office Regions of England with Wales and Scotland added as single regions. Some speakers had to be thrown out at this point because they lacked brithplace information or were born outside the UK. Each region varied in size; however, the average number of sentences per corpus was 4682, with an average of 44,726 words per corpus (see table 1). Thus, the average sentence length was 9.55 words. The average corpus was smaller than the Norwegian L2 English corpora of Nerbonne and Wiersma (2006), which had two groups, one with 221,000 words and the other with 84,000.

Significant differences (at $p < 0.05$) were found when comparing the largest regions, but no significant differences were found when comparing small regions to other small regions. The significant differences found are given in table 2 and 3. It seems that summed corpus size must reach a certain threshold before differences can be observed reliably: about 250,000 words for leaf-ancestor paths and 100,000 for trigrams. There are exceptions in both directions; the total size of London compared to Wales is larger than the size of London compared to the East Midlands, but the former is not statistically different. On the other hand, the total size of Southeast England compared to Scotland is only half of the other significantly different comparisons; this difference may be a result of more extreme syntactic differences than the other areas. Finally, it is interesting to note that the summed Norwegian corpus size is around 305,000 words, which is about three times the size needed for significance as estimated from the ICE-GB data.

4 Discussion

Our work extends that of Nerbonne and Wiersma (2006) in a number of ways. We have shown that an alternate method of representing syntax still allows the permutation test to find significant differences between corpora. In addition, we have shown differences between corpora divided by geographical area rather than language proficiency, with many more corpora than before. Finally, we have shown that the size of the corpus can be reduced somewhat

and still obtain significant results.

Furthermore, we also have shown that both leaf-ancestor paths and POS trigrams give similar results, although the more complex paths require more data.

However, there are a number of directions that this experiment should be extended. A comparison that divides the speakers into traditional British dialect areas is needed to see if the same differences can be detected. This is very likely, because corpus divisions that better reflect reality have a better chance of achieving a significant difference.

In fact, even though leaf-ancestor paths should provide finer distinctions than trigrams and thus require more data for detectable significance, the regional corpora presented here were smaller than the Norwegian speakers' corpora in Nerbonne and Wiersma (2006) by up to a factor of 10. This raises the question of a lower limit on corpus size. Our experiment suggests that the two corpora must have at least 250,000 words, although we suspect that better divisions will allow smaller corpus sizes.

While we are reducing corpus size, we might as well compare the increasing numbers of smaller and smaller corpora in an advantageous order. It should be possible to cluster corpora by the point at which they fail to achieve a significant difference when split from a larger corpus. In this way, regions could be grouped by their detectable boundaries, not a priori distinctions based on geography or existing knowledge of dialect boundaries.

Of course this indirect method would not be needed if one had a direct method for clustering speakers, by distance or other measure. Development of such a method is worthwhile research for the future.

References

Phillip Good. 1995. *Permutation Tests*. Springer, New York.

Charlotte S. Gooskens and Wilbert J. Heeringa. 2004. Perceptive evaluations of levenshtein dialect distance measurements using norwegian dialect data. *Language Variation and Change*, 16(3):189–207.

Wilbert J. Heeringa. 2004. *Measuring Dialect Pronunciation Differences using Levenshtein Distance*. Doctoral dissertation, University of Groningen.

Brett Kessler. 2001. *The Significance of Word Lists*. CSLI Press, Stanford.

Gerald Nelson, Sean Wallis, and Bas Aarts. 2002. *Exploring Natural Language: working with the British component of the International Corpus of English*. John Benjamins Publishing Company, Amsterdam/Philadelphia.

John Nerbonne and Wilbert Heeringa. 1997. Measuring dialect distance phonetically. In John Coleman, editor, *Workshop on Computational Phonology*, pages 11–18, Madrid. Special Interest Group of the Assocation for Computational Linguistics.

John Nerbonne and Wybo Wiersma. 2006. A measure of aggregate syntactic distance. In John Nerbonne and Erhard Hinrichs, editors, *Linguistic Distances*, pages 82–90, Sydney, July. International Committee on Computational Linguistics and the Assocation for Computational Linguistics.

Geoffrey Sampson. 2000. A proposal for improving the measurement of parse accuracy. *International Journal of Corpus Linguistics*, 5(1):53–68, August.

Jean Séguy. 1973. La dialectometrie dans l'atlas linguistique de la gascogne. *Revue de linguistique romane*, 37:1–24.

Inducing Combinatory Categorial Grammars with Genetic Algorithms

Elias Ponvert
Department of Linguistics
University of Texas at Austin
1 University Station B5100
Austin, TX 78712-0198 USA
ponvert@mail.utexas.edu

Abstract

This paper proposes a novel approach to the induction of Combinatory Categorial Grammars (CCGs) by their potential affinity with the Genetic Algorithms (GAs). Specifically, CCGs utilize a rich yet compact notation for lexical categories, which combine with relatively few grammatical rules, presumed universal. Thus, the search for a CCG consists in large part in a search for the appropriate categories for the data-set's lexical items. We present and evaluates a system utilizing a simple GA to successively search and improve on such assignments. The fitness of categorial-assignments is approximated by the coverage of the resulting grammar on the data-set itself, and candidate solutions are updated via the standard GA techniques of reproduction, crossover and mutation.

1 Introduction

The discovery of grammars from unannotated material is an important problem which has received much recent research. We propose a novel approach to this effort by leveraging the theoretical insights of Combinatory Categorial Grammars (CCG) (Steedman, 2000), and their potential affinity with Genetic Algorithms (GA) (Goldberg, 1989). Specifically, CCGs utilize an extremely small set of grammatical rules, presumed near-universal, which operate over a rich set of grammatical categories, which are themselves simple and straightforward data structures. A search for a CCG grammar for a language can be construed as a search for accurate category assignments to the words of that language, albeit over a large landscape of potential solutions. GAs are biologically-inspired general purpose search/optimization methods that have succeeded in these kinds of environments: wherein solutions are straightforwardly coded, yet nevertheless the solution space is complex and difficult.

We evaluate a system that uses a GA to successively refine a population of categorial lexicons given a collection of unannotated training material.

This is an important problem for several reasons. First of all, the development of annotated training material is expensive and difficult, and so schemes to discover linguistic patterns from unannotated text may help cut down the cost of corpora development. Also, this project is closely related to the problem of resolving lexical gaps in parsing, which is a dogged problem for statistical parsing systems in CCG, even trained in a supervised manner. Carrying over techniques from this project to that could help solve a major problem in CCG parsing technology.

Statistical parsing with CCGs is an active area of research. The development of CCGbank (Hockenmaier and Steedman, 2005) based on the Penn Treebank has allowed for the development of wide-coverage statistical parsers. In particular, Hockenmaier and Steedman (2001) report a generative model for CCG parsing roughly akin to the Collins parser (Collins, 1997) specific to CCG. Whereas Hockenmaier's parser is trained on (normal-form) CCG derivations, Clark and Curran (2003) present a CCG parser trained on the dependency structures within parsed sentences, as well as the possible derivations for them, using a log-linear (Maximum-Entropy) model. This is one of the most accurate parsers for producing deep dependencies currently available. Both systems, however, suffer from gaps

Proceedings of the ACL 2007 Student Research Workshop, pages 7–12,
Prague, June 2007.

in lexical coverage.

The system proposed here was evaluated against a small corpus of unannotated English with the goal of inducing a categorial lexicon for the fragment. The system is not ultimately successful and fails to achieve the baseline category assignment accuracy, however it does suggest directions for improvement.

2 Background

2.1 Genetic Algorithms

The basic insight of a GA is that, given a problem domain for which solutions can be straightforwardly encoded as *chromosomes*, and for which candidate solutions can be evaluated using a faithful *fitness function*, then the biologically inspired operations of *reproduction*, *crossover* and *mutation* can in certain cases be applied to multisets or *populations* of candidate solutions toward the discovery of true or approximate solutions.

Among the applications of GA to computational linguistics, (Smith and Witten, 1995) and (Korkmaz and Üçoluk, 2001) each present GAs for the induction of phrase structure grammars, applied successfully over small data-sets. Similarly, (Losee, 2000) presents a system that uses a GA to learn part-of-speech tagging and syntax rules from a collection of documents. Other proposals related specifically to the acquisition of categorial grammars are cited in §2.3.

2.2 Combinatory Categorial Grammar

CCG is a mildly context sensitive grammatical formalism. The principal design features of CCG is that it posits a small set of grammatical rules that operate over rich grammatical categories. The categories are, in the simplest case, formed by the atomic categories s (for *sentence*), np (*noun phrase*), n (*common noun*), etc., closed under the slash operators $/$, $\backslash$. There is not a substantive distinction between lexical and phrasal categories. The intuitive interpretation of non-atomic categories is as follows: a word for phrase of type A/B is looking for an item of type B on the right, to form an item of type A. Likewise, an item of type $A\backslash B$ is looking for an item of type B on the left. type A. For example, in the derivation in Figure 1, "scores" combines with the np "another goal" to form the verb phrase "scores another goal". This, in turn, combines with the np "Ronaldinho" to form a sentence.

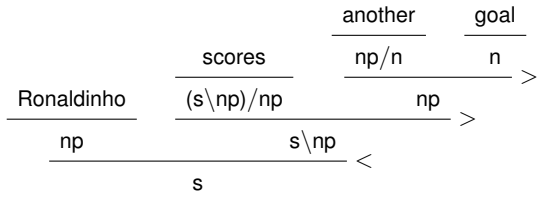


Figure 1: Example CCG derivation

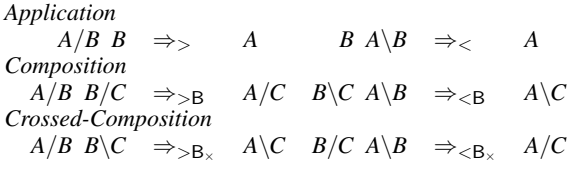


Figure 2: CCG Rules

The example illustrates the rule of Application, denoted with $<$ and $>$ in derivations. The schemata for this rule, along with the Composition rule (B) and the Crossed-Composition rule ($B_\times$), are given in Figure 2. The rules of CCG are taken as universals, thus the acquisition of a CCG grammar can be seen as the acquisition of a categorial lexicon.

2.3 Related Work

In addition to the supervised grammar systems outlined in §1, the following proposals have been put forward toward the induction of categorial grammars.

Watkinson and Mandahar (2000) report a Categorial Grammar induction system related to that proposed here. They generate a Categorial Grammar using a fixed and limited set of categories and, utilizing an unannotated corpus, successively refine the lexicon by testing it against the corpus sentences one at a time. Using a constructed corpus, their strategy worked extremely well: 100% accuracy on lexical category selection as well as 100% parsing accuracy with the resulting statistical CG parser. With naturally occurring text, however, their system does not perform as well: approximately 77% lexical accuracy and 37% parsing accuracy.

One fundamental difference between the strategy proposed here and that of Watkinson and Manda-

har is that we propose to successively generate and evaluate *populations* of candidate solutions, rather than refining a single solution. Also, while Watkinson and Mandahar use logical methods to construct a probabilistic parser, the present system uses approximate methods and yet derives symbolic parsing systems. Finally, Watkinson and Mandahar utilize an extremely small set of known categories, smaller than the set used here.

Clark (1996) outlines a strategy for the acquisition of Tree-Adjoining Grammars (Joshi, 1985) similar to the one proposed here: specifically, he outlines a learning model based on the *co-evolution* of a parser, which builds parse trees given an input string and a set of category-assignments, and a *shredder*, which chooses/discovers category-assignments from parse-trees. The proposed strategy is not implemented and tested, however.

Briscoe (2000) models the acquisition of categorial grammars using evolutionary techniques from a different perspective. In his experiments, language agents induced parameters for languages from other language agents generating training material. The acquisition of languages is not induced using GA per se, but the evolutionary development of languages is modeled using GA techniques.

Also closely related to the present proposal is the work of Villavicencio (2002). Villavicencio presents a system that learns a unification-based categorial grammar from a semantically-annotated corpus of child-directed speech. The learning algorithm is based on a Principles-and-Parameters language acquisition scheme, making use of logical forms and word order to induce possible categories within a typed feature-structure hierarchy. Her system has the advantage of not having to pre-compile a list of known categories, as did Watkinson and Mandahar as well as the present proposal. However, Villavicencio does make extensive use of the semantics of the corpus examples, which the current proposal does not. This is related to the divergent motivations of two proposals: Villavicencio aims to present a psychologically realistic language learner and takes it as psychologically plausible that logical forms are accessible to the language learner; the current proposal is preoccupied with grammar induction from unannotated text, and assumes (sentence-level) logical forms to be inaccessible.

```
n is the size of the population
A are candidate category assignments
F are fitness scores
E are example sentences
m is the likelihood of mutation

Initialize:
  for i ← 1 to n :
    A[i] ← RANDOMASSIGNMENT()
Loop:
  for i ← 1 to length[A] :
    F[i] ← 0
    P ← NEWPARSER(A[i])
    for j ← 1 to length[E] :
      F[i] ← F[i] + SCORE(P.PARSE(E[i]))
  A ← REPRODUCE(A, F)
  ▷ Crossover:
  for i ← 1 to n − 1 :
    CROSSOVER(A[i], A[i + 1])
  ▷ Mutate:
  for i ← 1 to n :
    if RANDOM() < m :
      MUTATE(A[i])
Until: End conditions are met
```

Figure 3: Pseudo-code for CCG induction GA.

3 System

As stated, the task is to choose the correct CCG categories for a set of lexical items given a collection of unannotated or minimally annotated strings. A candidate solution *genotype* is an assignment of CCG categories to the lexical items (types rather than tokens) contained in the textual material. A candidate *phenotype* is a CCG parser initialized with these category assignments. The fitness of each candidate solution is evaluated by how well its phenotype (parser) parses the strings of the training material.

Pseudo-code for the algorithm is given in Fig. 3. For the most part, very simple GA techniques were used; specifically:

- REPRODUCE The reproduction scheme utilizes roulette wheel technique: initialize a weighted roulette wheel, where the sections of the wheel correspond to the candidates and the weights of the sections correspond to the fitness of the candidate. The likelihood that a candidate is selected in a roulette wheel spin is directly proportionate to the fitness of the candidate.

- CROSSOVER The crossover strategy is a simple partition scheme. Given two candidates C and

D, choose a center point $0 \leq i \leq n$ where n the number of genes (category-assignments), swap C[0, i] $\leftarrow$ D[0, i] and D[i, n] $\leftarrow$ C[i, n].

- MUTATE The mutation strategy simply swaps a certain number of individual assignments in a candidate solution with others. For the experiments reported here, if a given candidate is chosen to be mutated, 25% of its genes are modified. The probability a candidate was selected is 10%.

In the implementation of this strategy, the following simplifying assumptions were made:

- A given candidate solution only posits a single CCG category for each lexical item.
- The CCG categories to assign to the lexical items are known a priori.
- The parser only used a subset of CCG – pure CCG (Eisner, 1996) – consisting of the Application and Composition rules.

3.1 Chromosome Encodings

A candidate solution is a simplified assignment of categories to lexical items, in the following manner. The system creates a candidate solution by assigning lexical items a random category selection, as in:

Ronaldinho	(s\np)/np
Barcelona	pp
kicks	(s\np)/(s\np)
⋮	

Given the fixed vocabulary, and the fixed category list, the representation can be simplified to lists of indices to categories, indexed to the full vocabulary list:

0	Ronaldinho
1	Barcelona
2	kicks
⋮	

⋮	
15	(s\np)/np
⋮	
37	(s\np)/(s\np)
⋮	

Then the category assignment can be construed as a finite function from word-indices to category-indices $\{0 \mapsto 15, 1 \mapsto 42, 2 \mapsto 37, ...\}$ or simply the vector $\langle 15, 42, 37, ... \rangle$. The chromosome encodings for the GA scheme described here are just this: vectors of integer category indices.

3.2 Fitness

The parser used is straightforward implementation of the normal-form CCG parser presented by Eisner (1996). The fitness of the parser is evaluated on its parsing coverage on the individual strings, which is a score based on the chart output. Several chart fitness scores were evaluated, including:

- SPANS The number of spans parsed
- RELATIVE The number of spans the string parsed divided by the string length
- WEIGHTED The sum of the lengths of the spans parsed

See §5.1 for a comparison of these fitness metrics.

Additionally, the following also factored into scoring parses:

- S-BONUS Add an additional bonus to candidates for each sentence they parse completely.
- PSEUDO-SMOOTHING Assign all parses at least a small score, to help avoid premature convergence. The metrics that count singleton spans do this informally.

4 Evaluation

The system was evaluated on a small data-set of examples taken from the World Cup test-bed included with the OpenCCG grammar development system[1] and simplified considerably. This included 19 example sentences with a total of 105 word-types and 613 tokens from (Baldridge, 2002).

In spite of the simplifying assumption that an individual candidate only assigns a single category to a lexical item, one can derive a multi-assignment of categories to lexemes from the population by choosing the top category elected by the candidates. It is on the basis of these derived assignments that the system was evaluated. The examples chosen require only 1-to-1 category assignment, hence the relevant category from the test-bed constitutes the gold standard (minus Baldridge (2002)'s modalities). The baseline for this dataset, assigning np to all lexical items, was 28.6%. The hypothesis is that optimizing

[1] http://openccg.sf.net

Fitness Metric	Accuracy
COUNT	18.5
RELATIVE	22.0
WEIGHTED	20.4

Table 1: Final accuracy of the metrics

parsing coverage with a GA scheme would correlate with improved category-accuracy.

The end-conditions apply if the parsing coverage for the derived grammar exceeds 90%. Such end-conditions generally were not met; otherwise, experiments ran for 100 generations, with a population of 50 candidates. Because of the heavy reliance of GAs on pseudo-random number generation, individual experiments can show idiosyncratic success or failure. To control for this, the experiments were replicated 100 times each. The results presented here are averages over the runs.

5 Results

5.1 Fitness Metrics

The various fitness metrics were each evaluated, and their final accuracies are reported in Table 1. The results were negative, as category accuracy did not approach the baseline. Examining the average system accuracy over time helps illustrate some of the issues involved. Figure 4 shows the growth of category accuracy for each of the metrics. Pathologically, the random assignments at the start of each experiment have better accuracy than after the application of GA techniques.

Figure 5 compares the accuracy of the category assignments to the GA's internal measure of its fitness, using the Count Spans metric as a point of reference. (The fitness metric is scaled for comparison with the accuracy.) While fitness, in the average case, steadily increases, accuracy does not increase with such steadiness and degrades significantly in the early generations.

The intuitive reason for this is that, initially, the random assignment of categories succeeds by chance in many cases, however the likelihood of accurate or even compatible assignments to words that occur adjacent in the examples is fairly low. The GA promotes these assignments over others, apparently committing the candidates to incorrect assignments early on and not recovering from these commitments. The WEIGHTED and RELATIVE metrics are designed to try to overcome these effects by promoting grammars that parse longer spans, but they do not succeed. Perhaps exponential rather than linear bonus for parsing spans of length greater than two would be effective.

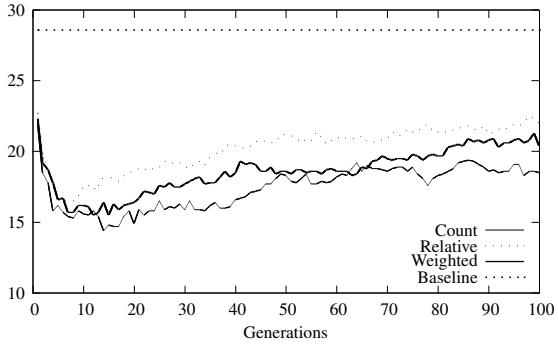


Figure 4: Comparison of fitness metrics

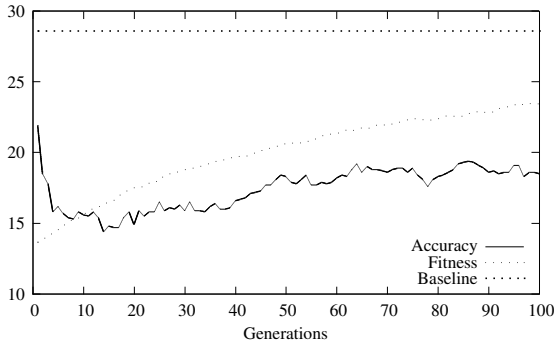


Figure 5: Fitness and accuracy: COUNT

6 Conclusions

This project attempts to induce a grammar from unannotated material, which is an extremely difficult problem for computational linguistics. Without access to training material, logical forms, or other relevant features to aid in the induction, the system attempts to learn from string patterns alone. Using GAs may aid in this process, but, in general, induction from string patterns alone takes much larger data-sets than the one discussed here.

The GA presented here takes a global perspective on the progress of the candidates, in that the individual categories assigned to the individual words are not evaluated directly, but rather as members of candidates that are scored. For a system such as

this to take advantage of the patterns that arise out of the text itself, a much more fine-grained perspective is necessary, since the performance of individual category-assignments to words being the focus of the task.

7 Acknowledgements

I would like to thank Jason Baldridge, Greg Kobele, Mark Steedman, and the anonymous reviewers for the ACL Student Research Workshop for valuable feedback and discussion.

References

Jason Baldridge. 2002. *Lexically Specified Derivational Control in Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Ted Briscoe. 2000. Grammatical acquisition: Inductive bias and coevolution of language and the language acquisition device. *Language*, 76:245–296.

Stephen Clark and James R Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of EMNLP-03*, pages 97–105, Sapporo, Japan.

Robin Clark. 1996. Complexity and the induction of Tree Adjoining Grammars. Unpublished manuscript, University of Pennsylvania.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL-97*, pages 16–23, Madrid, Spain.

Jason Eisner. 1996. Efficient normal-form parsing for Combinatory Categorial Grammar. In *Proceedings of ACL-96*, pages 79–86, Santa Cruz, USA.

David E. Goldberg. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Julia Hockenmaier and Mark Steedman. 2001. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of ACL*, pages 335–342, Philadelphia, USA.

Julia Hockenmaier and Mark Steedman. 2005. CCGbank: User's manual. Technical Report MC-SIC-05-09, Department of Computer and Information Science, University of Pennsylvania.

Aravind Joshi. 1985. An introduction to Tree Adjoining Grammars. In A. Manaster-Ramer, editor, *Mathematics of Language*. John Benjamins.

Emin Erkan Korkmaz and Göktürk Üçoluk. 2001. Genetic programming for grammar induction. In *2001 Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 245–251, San Francisco, USA.

Rober M. Losee. 2000. Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules. *Information Processing and Management*, 32:185–197.

Tony C. Smith and Ian H. Witten. 1995. A genetic algorithm for the induction of natural language grammars. In *Proc. of IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*, pages 17–24, Montreal, Canada.

Mark Steedman. 2000. *The Syntactic Process*. MIT, Cambridge, Mass.

Aline Villavicencio. 2002. *The Acquisition of a Unification-Based Generalised Categorial Grammar*. Ph.D. thesis, University of Cambridge.

Stephen Watkinson and Suresh Manandhar. 2000. Unsupervised lexical learning with categorial grammars using the LLL corpus. In James Cussens and Sašo Džeroski, editors, *Language Learning in Logic*, pages 16–27, Berlin. Springer.

An Implementation of Combined Partial Parser and Morphosyntactic Disambiguator

Aleksander Buczyński
Institute of Computer Science
Polish Academy of Sciences
Ordona 21, 01-237 Warszawa, Poland
olekb@ipipan.waw.pl

Abstract

The aim of this paper is to present a simple yet efficient implementation of a tool for simultaneous rule-based morphosyntactic tagging and partial parsing formalism. The parser is currently used for creating a treebank of partial parses in a valency acquisition project over the IPI PAN Corpus of Polish.

1 Introduction

1.1 Motivation

Usually tagging and partial parsing are done separately, with the input to a parser assumed to be a morphosyntactically fully disambiguated text. Some approaches (Karlsson et al., 1995; Schiehlen, 2002; Müller, 2006) interweave tagging and parsing. (Karlsson et al., 1995) is actually using the same formalism for both tasks — it is possible, because all words in this dependency-based approach come with all possible syntactic tags, so partial parsing is reduced to rejecting wrong hypotheses, just as in case of morphosyntactic tagging.

Rules used in rule-based tagging often implicitly identify syntactic constructs, but do not mark such constructs in texts. A typical such rule may say that when an unambiguous dative-taking preposition is followed by a number of possibly dative adjectives and a noun ambiguous between dative and some other case, then the noun should be disambiguated to dative. Obviously, such a rule actually identifies a PP and some of its structure.

Following the observation that both tasks, morphosyntactic tagging and partial constituency parsing, involve similar linguistic knowledge, a formalism for simultaneous tagging and parsing was proposed in (Przepiórkowski, 2007). This paper presents a revised version of the formalism and a simple implementation of a parser understanding rules written according to it. The input to the rules is a tokenised and morphosyntactically annotated XML text. The output contains disambiguation annotation and two new levels of constructions: syntactic words and syntactic groups.

2 The Formalism

2.1 Terminology

In the remainder of this paper we call the smallest interpreted unit, i.e., a sequence of characters together with their morphosyntactic interpretations (lemma, grammatical class, grammatical categories) a *segment*. A *syntactic word* is a non-empty sequence of segments and/or syntactic words. Syntactic words are named entities, analytical forms, or any other sequences of tokens which, from the syntactic point of view, behave as single words. Just as basic words, they may have a number of morphosyntactic interpretations. By a *token* we will understand a segment or a syntactic word. A *syntactic group* (in short: group) is a non-empty sequence of tokens and/or syntactic groups. Each group is identified by its syntactic head and semantic head, which have to be tokens. Finally, a *syntactic entity* is a token or a syntactic group; it follows that syntactic groups may be defined as a non-empty sequence of entities.

Proceedings of the ACL 2007 Student Research Workshop, pages 13–18,
Prague, June 2007.

2.2 The Basic Format

Each rule consists of up to 4 parts: `Match` describes the sequence of syntactic entities to find; `Left` and `Right` — restrictions on the context; `Actions` — a sequence of morphological and syntactic actions to be taken on the matching entities.

For example:

```
Left:
Match: [pos~~"prep"][base~"co|kto"]
Right:
Actions: unify(case,1,2);
         group(PG,1,2)
```

means:

- find a sequence of two tokens such that the first token is an unambiguous preposition (`[pos~~"prep"]`), and the second token is a possible form of the lexeme CO 'what' or KTO 'who' (`[base~"co|kto"]`),
- if there exist interpretations of these two tokens with the same value of case, reject all interpretations of these two tokens which do not agree in case (cf. `unify(case,1,2)`);
- if the above unification did not fail, mark thus identified sequence as a syntactic group (`group`) of type `PG` (prepositional group), whose syntactic head is the first token (`1`) and whose semantic head is the second token (`2`; cf. `group(PG,1,2)`);

`Left` and `Right` parts of a rule may be empty; in such a case the part may be omitted.

2.3 Left, Match and Right

The contents of parts `Left`, `Match` and `Right` have the same syntax and semantics. Each of them may contain a sequence of the following specifications:

- token specification, e.g., `[pos~~"prep"]` or `[base~"co|kto"]`; these specifications adhere to segment specifications of the Poliqarp (Janus and Przepiórkowski, 2006) corpus search engine; in particular there is a distinction between certain and uncertain information — a specification like `[pos~~"subst"]` says that *all* morphosyntactic interpretations of a given token are nominal (substantive), while `[pos~"subst"]` means that there *exists* a nominal interpretation of a given token;
- group specification, extending the Poliqarp query as proposed in (Przepiórkowski, 2007), e.g., `[semh=[pos~~"subst"]]` specifies a syntactic group whose semantic head is a token whose all interpretations are nominal;
- one of the following specifications:
  - `ns`: no space,
  - `sb`: sentence beginning,
  - `se`: sentence end;
- an alternative of such sequences in parentheses.

Additionally, each such specification may be modified with one of the three standard regular expression quantifiers: `?`, `*` and `+`.

An example of a possible value of `Left`, `Match` or `Right` might be:

```
[pos~"adv"] ([pos~~"prep"]
[pos~"subst"] ns?  [pos~"interp"]?
se | [synh=[pos~~"prep"]])
```

2.4 Actions

The `Actions` part contains a sequence of morphological and syntactic actions to be taken when a matching sequence of syntactic entities is found. While morphological actions delete some interpretations of specified tokens, syntactic actions group entities into syntactic words or syntactic groups. The actions may also include conditions that must be satisfied in order for other actions to take place, for example case or gender agreement between tokens.

The actions may refer to entities matched by the specifications in `Left`, `Match` and `Right` by numbers. These specifications are numbered from 1, counting from the first specification in `Left` to the last specification in `Right`. For example, in the following rule, there should be case agreement between the adjective specified in the left context and the adjective and the noun specified in the right context (cf. `unify(case,1,4,5)`), as well as case agreement (possibly of a different case) between the adjective and noun in the match (cf. `unify(case,2,3)`).

```
Left:  [pos~~"adj"]
Match: [pos~~"adj"][pos~~"subst"]
```

```
Right: [pos~~"adj"][pos~~"subst"]
Actions: unify(case,2,3);
         unify(case,1,4,5)
```

The exact repertoire of actions still evolves, but the most frequent are:

- `agree(<cat>,...,<tok>,...)` - check if the grammatical categories (`<cat>,...`) of entities specified by subsequent numbers (`<tok>,...`) agree;
- `unify(<cat>,...,<tok>,...)` - as above, plus delete interpretations that do not agree;
- `delete(<cond>,<tok>,...)` - delete all interpretations of specified tokens matching the specified condition (for example `case~"gen|acc"`)
- `leave(<cond>,<tok>,...)` - leave only the interpretations matching the specified condition;
- `nword(<tag>,<base>)` - create a new syntactic word with given tag and base form;
- `mword(<tag>,<tok>)` - create a new syntactic word by copying and appropriately modifying all interpretations of the token specified by number;
- `group(<type>,<synh>,<semh>)` - create a new syntactic group with syntactic head and semantic head specified by numbers.

The actions `agree` and `unify` take a variable number of arguments: the initial arguments, such as `case` or `gender`, specify the grammatical categories that should *simultaneously* agree, so the condition `agree(case gender,1,2)` is properly stronger than the sequence of conditions: `agree(case,1,2)`, `agree(gender,1,2)`. Subsequent arguments of `agree` are natural numbers referring to entity specifications that should be taken into account when checking agreement.

A reference to entity specification refers to all entities matched by that specification, so, e.g., in case `1` refers to specification `[pos~adj]*`, `unify(case,1)` means that all adjectives matched by that specification must be rid of all interpretations whose case is not shared by all these adjectives.

When a reference refers to a syntactic group, the action is performed on the syntactic head of that group. For example, assuming that the following rule finds a sequence of a nominal segment, a multi-segment syntactic word and a nominal group, the action `unify(case,1)` will result in the unification of case values of the first segment, the syntactic word as a whole and the syntactic head of the group.

```
Match: ([pos~~"subst"] |
        [synh=[pos~~"subst"]])+
Action: unify(case,1)
```

The only exception to this rule is the semantic head parameter in the `group` action; when it references a syntactic group, the semantic, not syntactic, head is inherited.

For `mword` and `nword` actions we assume that the orthographic form of the created syntactic word is always a simple concatenation of all orthographic forms of all tokens immediately contained in that syntactic word, taking into account information about space or its lack between consecutive tokens.

The `mword` action is used to copy and possibly modify all interpretations of the specified token. For example, a rule identifying negated verbs, such as the rule below, may require that the interpretations of the whole syntactic word be the same as the interpretations of the verbal segment, but with `neg` added to each interpretation.

```
Left: ([pos!~"prep"]|[case!~"acc"])
Match: [orth~"[Nn]ie"][pos~~"verb"]
       (ns [orth~"by[mś]?"])?
       (ns [pos~~"aglt"])?
Actions: leave(pos~"qub", 2);
         mword(neg,3)
```

The `nword` action creates a syntactic word with a new interpretation and a new base form (lemma). For example, the rule below will create, for a sequence like *mimo tego, że* or *Mimo że* 'in spite of, despite', a syntactic word with the base form MIMO ŻE and the conjunctive interpretation.

```
Match: [orth~"[Mm]imo"]
       [orth~"to|tego"]?
       (ns [orth~","])? [orth~"że"]
Actions: leave(pos~"prep",1);
```

```
leave(pos~"subst",2);
nword(conj, mimo że)
```

The `group(<type>,<synh>,<semh>)` action creates a new syntactic group, where `<type>` is the categorial type of the group (e.g., `PG`), while `<synh>` and `<semh>` are references to appropriate token specifications in the `Match` part. For example, the following rule may be used to create a numeral group, syntactically headed by the numeral and semantically headed by the noun:

```
Left:   [pos~~"prep"]
Match:  [pos~"num"][pos~"adj"]*
        [pos~"subst"]
Actions: group(NumG,2,4)
```

Of course, the rules should be constructed in such a way that references `<synh>` and `<semh>` refer to specifications of single entities, e.g., `([pos~"subst"]|[synh=[pos~"subst"]])` but not `[case~"nom"]+`

3 The Implementation

3.1 Objectives

The goal of the implementation was a combined partial parser and tagger that would be reasonably fast, but at the same time easy to modify and maintain. At the time of designing and implementing the parser, neither the set of rules, nor the specific repertoire of possible actions within rules was known, hence, the flexibility and modifiability of the design was a key issue.

3.2 Input and Output

The parser currently takes as input the version of the XML Corpus Encoding Standard (Ide et al., 2000) assumed in the IPI PAN Corpus of Polish (`korpus.pl`). The tagset is configurable, therefore the tool can be possibly used for other languages as well.

Rules may modify the input in one of two ways. Morphological actions may delete certain interpretations of certain tokens; this fact is marked by the attribute `disamb="0"` added to `<lex>` elements representing these interpretations. On the other hand, syntactic actions modify the input by adding `<syntok>` and `<group>` elements, marking syntactic words and groups.

3.3 Algorithm Overview

During the initialisation phase, the parser loads the external tagset specification and the ruleset, and converts the latter to a set of compiled regular expressions and actions. Then input files are parsed one by one (for each input file a corresponding output file containing parsing results is created). To reduce memory usage, the parsing is done by chunks defined in the input files, such as sentences or paragraphs. In the remainder of the paper we assume the chunks are sentences.

During the parsing, a sentence has dual representation:

1. object-oriented syntactic entity tree, used for easy manipulation of entities (for example disabling certain interpretations or creating new syntactic words) and preserving all necessary information to generate the final output;
2. compact string for quick regexp matching, containing only the informations important for these rules which have not been applied yet.

The entity tree is initialised as a flat (one level deep) tree with all leaves (segments and possibly special entities, like no space, sentence beginning, sentence end) connected directly to the root. Application of a syntactic action means inserting a new node (syntacting word or group) to the tree, between the root and a few of the existing nodes. As the parsing processes, the tree changes its shape: it becomes deeper and narrower.

The string representations is consistently updated to always represent the top level of the tree (the children of the root). Therefore, the searched string's length tends to decrease with every action applied (as opposed to increasing in a naïve implementation, with single representation and syntactic / disambiguation markup added). This is not a strictly monotonous process, as creating new syntactic entities containing only one segment may temporarily increase the length, but the increase is offset with the next rule applied to this entity (and generally the point of parsing is to eventually find groups longer than one segment).

Morphological actions do not change the shape of the tree, but also reduce the string representation

length by deleting from the string certain interpretations. The interpretations are preserved in the tree to produce the final output, but are not interesting for further stages of parsing.

3.4 Representation of Sentence

The string representation is a compromise between XML and binary representation, designed for easy, fast and precise matching, with the use of existing regular expression libraries.

The representation describes the top level of the current state of the sentence tree, including only the informations that may be used by rule matching. For each child of the tree root, the following informations are preserved in the string: type (token / group / special) and identifier (allowing to find the entity in the tree in case an action should be applied to it). The further part of the string depends on the type — for token it is orthografic forms and a list of interpretations; for group — number of heads of the group and lists of interpretations of syntactic and semantic head.

Every interpretation consists of a base form and a morphosyntactic tag (part of speech, case, gender, numer, degree, etc.). Because the tagset used in the IPI PAN Corpus is intended to be human readable, the morphosyntactic tag is fairly descriptive (long values) and, on the other hand, compact (may have many parts ommited, for example when the category is not applicable to the given part of speech). To make pattern matching easier, the tag is converted to a string of fixed width. In the string, each character corresponds to one morphological category from the tagset (first part of speech, then number, case, gender etc.) as for example in the Czech positional tag system (Hajič and Hladká, 1997). The characters — upper- and lowercase letters, or 0 (zero) for categories non-applicable for a given part of speech — are assigned automatically, on the basis of the external tagset definition read at initialisation. A few examples are presented in table 1.

IPI PAN tag	fixed length tag
`adj:pl:acc:f:sup`	`UBDD0C0000000`
`conj`	`B000000000000`
`fin:pl:sec:imperf`	`bB00B0A000000`

Table 1: Examples of tag conversion between human readable and inner positional tagset.

3.5 Rule Matching

The conversion from the `Left`, `Match` and `Right` parts of the rule to a regular expression over the string representation is fairly straightforward. Two exceptions — regular expressions as morphosyntactic category values and the distinction between existential and universal quantification over interpretations — will be described in more detail below.

First, the rule might be looking for a token whose grammatical category is described by a regular expresion. For example, `[gender~"m."]` should match human masculine (`m1`), animate masculine (`m2`), and inanimate masculine (`m3`) tokens; `[pos~"ppron[123]+|siebie"]` should match various pronouns; `[pos!~"num.*"]` should match all segments except for main and collective numerals; etc. Because morphosyntactic tags are converted to fixed length representations, the regular expressions also have to be converted before compilation.

To this end, the regular expression is matched against all possible values of the given category. Since, after conversion, every value is represented as a single character, the resulting regexp can use square brackets to represent the range of possible values.

The conversion can be done only for attributes with values from a well-defined, finite set. Since we do not want to assume that we know all the text to parse before compiling rules, we assume that the dictionary is infinite (this is different from Poliqarp, where dictionary is calculated during compilation of corpus to binary form). The assumption makes it difficult to convert requirements with negated `orth` or `base` (for example `[orth!~"[Nn]ie"]`). As for now, such requirements are not included in the compiled regular expression, but instead handled as an extra condition in the `Action` part.

Another issue that has to be taken into careful consideration is the distinction between certain and uncertain information. A segment may have many interpretations and sometimes a rule may apply only when all the interpretations meet the specified condition (for example `[pos~~"subst"]`), while in other cases one matching interpretation should be

enough to trigger the rule (`[pos~"subst"]`). The aforementioned requirements translate respectively to the following regular expressions:[1]

- `(<N[^<>]+)+`
- `(<[^<>]+)*(<N[^<>]+)(<[^<>]+)*`

Of course, a combination of existential and universal requirements is a valid requirement as well, for example: `[pos~~"subst" case~"gen|acc"]` (all interpretations noun, at least one of them in genitive or accusative case) should translate to:

```
(<N[^<>]+)*(<N.[BD][^<>]+)
(<N[^<>]+)*
```

3.6 Actions

When a match is found, the parser runs a sequence of actions connected with the given rule, described in 2.4. Each action may be condition, morphological action, syntactic action or a combination of the above (for example unify is both a condition and a morphological action). The parser executes the sequence until it encounters an action which evaluates to false (for example, unification of cases fails).

The actions affect both the tree and the string representation of the parsed sentence. The tree is updated instantly (cost of update is constant or linear to match lenght), but the string update (cost linear to sentence length) is delayed until it is really needed (at most once per rule).

4 Conclusion and Future Work

Althought morphosyntactic disambiguation rules and partial parsing rules often encode the same linguistic knowledge, we are not aware of any partial (or shallow) parsing systems accepting morphosyntactically ambiguous input and disambiguating it with the same rules that are used for parsing. This paper presents a formalism and a working prototype of a tool implementing simultaneous rule-based disambiguation and partial parsing.

Unlike other partial parsers, the tool does not expect a fully disambiguated input. The simplicity of the formalism and its implementation makes it possible to integrate a morphological analyser into parser and allow a greater flexibility in input formats.

On the other hand, the rule syntax can be extended to take advantage of the metadata present in the corpus (for example: style, media, or date of publishing). Many rules, both morphological and syntactic, may be applicable only to specific kinds of texts — for example archaic or modern, official or common.

[1] < and > were chosen as convenient separators of interpretations and entities, because they should not happen in the input data (they have to be escaped in XML).

References

Jan Hajič and Barbara Hladká. 1997. Tagging of inflective languages: a comparison. In *Proceedings of the ANLP'9y*, pages 136–143, Washington, DC.

Nancy Ide, Patrice Bonhomme, and Laurent Romary. 2000. XCES: An XML-based standard for linguistic corpora. In *Proceedings of the Linguistic Resources and Evaluation Conference*, pages 825–830, Athens, Greece.

Daniel Janus and Adam Przepiórkowski. 2006. Poliqarp 1.0: Some technical aspects of a linguistic search engine for large corpora. In Jacek Waliński, Krzysztof Kredens, and Stanisław Goźdź-Roszkowski, editors, *The proceedings of Practical Applications of Linguistic Corpora 2005*, Frankfurt am Main. Peter Lang.

F. Karlsson, A. Voutilainen, J. Heikkilä, and A. Anttila, editors. 1995. *Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text*. Mouton de Gruyter, Berlin.

Frank Henrik Müller. 2006. *A Finite State Approach to Shallow Parsing and Grammatical Functions Annotation of German*. Ph. D. dissertation, Universität Tübingen. Pre-final Version of March 11, 2006.

Adam Przepiórkowski. 2007. A preliminary formalism for simultaneous rule-based tagging and partial parsing. In Georg Rehm, Andreas Witt, and Lothar Lemnitzer, editors, *Datenstrukturen für linguistische Ressourcen und ihre Anwendungen – Proceedings der GLDV-Jahrestagung 2007*, Tübingen. Gunter Narr Verlag.

Adam Przepiórkowski. 2007. On heads and coordination in valence acquisition. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing (CICLing 2007)*, Lecture Notes in Computer Science, Berlin. Springer-Verlag.

Michael Schiehlen. 2002. Experiments in German noun chunking. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, Taipei.

A Practical Classification of Multiword Expressions

Radosław Moszczyński
Institute of Computer Science
Polish Academy of Sciences
Ordona 21, 01-237 Warszawa, Poland
rm@ipipan.waw.pl

Abstract

The paper proposes a methodology for dealing with multiword expressions in natural language processing applications. It provides a practically justified taxonomy of such units, and suggests the ways in which the individual classes can be processed computationally. While the study is currently limited to Polish and English, we believe our findings can be successfully employed in the processing of other languages, with emphasis on inflectional ones.

1 Introduction

radosław moszczyńskiIt is generally acknowledged that multiword expressions constitute a serious difficulty in all kinds of natural language processing applications (Sag et al., 2002). It has also been shown that proper handling of such expressions can result in significantly better results in parsing (Zhang et al., 2006).

The difficulties in processing multiword expressions result from their lexical variability, and the fact that many of them can undergo syntactic transformations. Another problem is that the label "multiword expressions" covers many linguistic units that often have little in common. We believe that the past approaches to formalize the phenomenon, such as IDAREX (Segond and Breidt, 1995) and Phrase Manager (Pedrazzini, 1994), suffered from trying to cover all multiword expressions as a whole. Such an approach, as is shown below, cannot efficiently cover all the phenomena related to multiword expressions.

Therefore, in the present paper we formulate a proposal of a taxonomy for multiword expressions, useful for the purposes of natural language processing. The taxonomy is based on the stages in the NLP workflow in which the individual classes of units can be processed successfully. We also suggest the tools that can be used for processing the units in each of the classes.

2 An NLP Taxonomy of Multiword Expressions

At this stage of work, our taxonomy is composed of two groups of multiword expressions. The first one consists of units that should be processed before syntactic analysis, and the other one includes expressions whose recognition should be combined with the syntactic analysis process. The next sections describe both groups in more detail.

2.1 Morphosyntactically Idiosyncratic Expressions

The first group consists of morphosyntactically idiosyncratic units. They follow unusual morphological and syntactic patterns, which causes difficulties for automatic analyzers.

By morphological idiosyncrasies we mean two types of units. First of all, there are bound words that do not inflect and cannot be used independently outside of the given multiword expression. In Polish, there are many such units, which are typically prepositional phrases functioning as complex adverbials, e.g.:[1]

[1]The asterisk in this and the following examples indicates an untranslatable bound word.

Proceedings of the ACL 2007 Student Research Workshop, pages 19–24,
Prague, June 2007.

(1) *na wskroś*
on *
'thoroughly'

Secondly, there are unusual forms of otherwise ordinary words that only appear in strictly defined multiword expressions. An example is the following unit, in which the genitive form of the noun 'daddy' is different than the one used outside this particular construction:

(2) *nie rób z tata*
Neg do-Imperative of *daddy-Gen
wariata
fool
'stop making a fool of me'

Morphological idiosyncrasies can be referred to as "objective" in the sense that it can be proved by doing corpus research that particular words only appear in a strictly limited set of constructions. Since outside such constructions the words do not have any meaning of their own, it is pointless to put them in the lexicon of a morphological analyzer. From the processing point of view, they are parts of complex multiword lexemes which should be considered as indivisible wholes.

Syntactically idiosyncratic phrases are those whose structure or behavior is incorrect from the point of view of a given grammar. In this sense, they are "subjective", because they depend on the rules underlying a particular parser.

A typical parser of Polish is expected to accept full sentences, i.e. phrases that contain a finite verb phrase, but possibly not many phraseologisms that are extremely common in texts and speech, and do not constitute proper sentences from the point of view of the grammar. This qualifies such phrases to be included and formalized among the first group we have distinguished. In Polish, such phrases include, e.g.:

(3) *Precz z łapami!*
off with hands-Inst
'Get your hands off!'

Another group of multiword expressions that should be processed before parsing consists of complex adverbials that do not include any bound words, but that could be interpreted wrongly by the syntactic analyzer. Consider the following multiword expression:

(4) *na kolanach*
on knees-Loc
'on one's knees' ('groveling')

This expression can be used in constructions of the following type:

(5) *Na kolanach Kowalskiego*
on knees-Loc Kowalski-Gen
będą błagać.
be-Future;Pl;3rd beg-Infinitive
'They will beg Kowalski on their knees.'

In the above example *na kolanach* is an adjunct that is not subcategorized for by any of the remaining constituents. However, since *Kowalskiego* is genitive, the parser would be fooled to believe that one of the possible interpretations is 'They will beg on Kowalski's knees', which is not correct and semantically odd. Such complex adverbials are very common in Polish, which is why we believe that formalizing them as wholes would allow us to achieve better parsing results.

The last type of units that it is necessary to formalize for syntactic analysis are multiword text cohesion devices and interjections, whose syntactic structure is hard to establish, as their constituents belong to weakly defined classes. They can also directly violate the grammar rules, as the coordination in the English example does:

(6) *bądź co bądź*
be-Imperative;Sg what be-Imperative;Sg
'after all'

(7) *by and large*

Since the recognition and tagging of all the above units will be performed before syntactic analysis, it seems natural to combine this process with a generalized mechanism of named entity recognition. We intend to build a preprocessor for syntactic analysis, along the lines of the ideas presented by Sagot and Boullier (2005). However, in addition to the set of named entities presented by the authors, we also intend to formalize multiword expressions of

the types presented above, possibly with the use of `lxtransduce`.[2] This will allow us to prepare the input to the parser in such a way as to eliminate all the unparsable elements. This in turn should result in significantly better parsing coverage.

2.2 Semantically Idiosyncratic Expressions

The other group in our classification consists of multiword expressions that are idiosyncratic from the point of view of semantics. It includes such units as:

(8) *NP-Nom* *wziąć* *nogi* *za*
NP-Nom to take legs-Acc under
pas
belt-Acc
'to run away'

From the syntactic analysis point of view, such units are not problematic, as they follow regular grammatical patterns. They create difficulties in other types of NLP-based applications, as their meaning is not compositional, and cannot be predicted from the meaning of their constituents. Examples of such applications include electronic dictionaries, which should be able to recognize idioms and provide an appropriate, non-literal translation (Prószéky and Földes, 2005).

Such expressions can be extremely complex due to the lexical and word order variations they can undergo, which is especially the case in such languages as Polish. The set of syntactic variations that are possible in unit (8) is very large. First of all, there is the subject (NP-Nom). English multiword expressions are usually encoded disregarding the subject, as it can never break the continuity of the other constituents. In Polish it is different — the subject can be absent altogether, it can appear at the very beginning of the multiword expression without breaking its continuity, but it can also appear after the verb, between the core constituents. The subject can be of arbitrary length and needs to agree in morphosyntactic features (number, gender, and person) with the verb.

The verb can be modified with adverbial phrases, both on the left hand side and the right hand side. However, if the subject is postponed to a position after the verb, all the potential right hand side adverbials need to be attached after the subject, and not directly after the verb. Thus, taking all the variation possibilities into account, it is not unlikely to encounter such phrases in Polish:

(9) *Wziął* *pan*
take-1sg;Masc;Past you-1sg;Masc;Nom
przed *wszystkimi* *nogi* *za*
before everyone legs-Acc under
pas!
belt-Acc
'You ran away before everyone else!'

Some of the English multiword expressions also display properties that make them difficult to process automatically. Although the word order is more rigid, it is still necessary to handle, e.g., passivization and nominalization. This concerns the canonical example of *spill the beans*, and many others.

It follows that the units in the second group should not, and probably cannot, be reliably encoded with the same means as the simpler units from Section 2.1, which can be accounted for properly with simple methods based on regular grammars and surface processing.

One possible solution is to encode the complex units with the rules of a formal grammar of the given language. Another solution could be constructing an appropriate valence dictionary for verbs in such expressions. Both possibilities imply that the recognition process should be performed simultaneously with syntactic analysis.

3 Rationale

The above classification was formulated during an examination of the available formalisms for encoding multiword expressions, which was a part of the present work.

The attempts to formalize multiword expressions for natural language processing can be roughly divided into two groups. There are approaches that aim at encoding such units with the rules of an existing formal grammar, such as the approach described by Debusmann (2004). On the other hand, specialized, limited formalisms have been created,

[2] `http://www.cogsci.ed.ac.uk/~richard/ltxml2/lxtransduce.html`

whose purpose is to encode only multiword expressions. Such formalisms include the already mentioned IDAREX (Segond and Breidt, 1995) and Phrase Manager (Pedrazzini, 1994).

The first approach has two drawbacks. One of them is that using the rules of a given grammar to encode multiword expressions seems to have sense only if the rest of the language is formalized in the same way. Thus, such an approach makes the lexicon of multiword expressions heavily dependant on a particular grammar, which might make its reuse difficult or impossible.

The other disadvantage concerns complexity. While full-blown grammars do have the means to handle the most complex multiword expressions and their transformational potential, they create too much overhead in the case of simple units, such as idiomatic prepositional phrases that function as adverbials, which have been presented above.

Thus, we decided to encode Polish multiword expressions with an existing, specialized formalism. However, after an evaluation of such formalisms none of the ones we were able to find proved to be adequate for Polish. This is mostly due to the properties of the language — Polish is highly inflectional and has a relatively free word order. Both of these properties also apply to multiword expressions, which implies that in order to capture all their possible variations in Polish, it is necessary to use a powerful formalism (cf. the example in (9)).

Our analysis revealed that IDAREX, which is a simple formalism based on regular grammars, is not appropriate for handling expressions that have a very variable word order and allow many modifications. In IDAREX, each multiword unit is encoded with a regular expression, whose symbols are words or POS-markers. The words are described in terms of two-level morphology, and can appear either on the lexical level (which permits inflection) or the surface level (which restricts the word to the form present in the regular expression). An example is provided below:

(10) `kick:  :the :bucket;`

Encoding the multiword expression in (8) with IDAREX in such a way as to include all the possible variations leads to a description that suffers from overgeneration. Also, IDAREX does not include any unification mechanisms. This makes it unsuitable for any generation purposes (and reliable recognition purposes, too), as Polish requires a means to enforce agreement between constituents.

Phrase Manager makes encoding multiword expressions difficult for other reasons. The methodology employed in the formalism requires each expression to be assigned to a predefined syntactic class which determines the unit's constituents, as well as the modifications and transformations that it can undergo:[3]

(11)
```
SYNTAX-TREE
(VP V (NP Art Adj N AdvP))
MODIFICATIONS
V >
TRANSFORMATIONS
Passive, N-Adj-inversion
```

Since it is sometimes the case that multiword expressions belonging to the same class differ in respect of the syntactic operations they can undergo, the classes are arranged into a tree-like structure in which a class might be subdivided further on into a subclass that allows passivization, another one that allows nominalization and subject-verb inversion, etc.

The problem with this approach is that it leads to a proliferation of classes. At least in Polish, multiword expressions that follow the same general syntactic pattern often differ in the transformations they allow. Besides, the formalism creates too much overhead in the case of simple multiword expressions. Consider the following example in Polish:

(12) *No nie!*
oh no
'Oh, come on!'

In Phrase Manager it would be necessary to define a syntactic class for this unit, which seems to be both superfluous and problematic, as it is hard to establish what parts of speech are the constituents without taking purely arbitrary decisions.

To complicate matters further, the expression in the example has a variant in which both constituents

[3]The transformations need to be defined with separate rules elsewhere. The whole description is abbreviated.

switch their positions (with the meaning preserved). In the case of such a simple expression, it is impossible to "name" this transformation and assign any syntactic or semantic prominence to it — it can safely be treated as a simple permutation. However, Phrase Manager requires each operation to be named and precisely defined in syntactic terms, which in this case is more than it is worth.

In our opinion both those formalisms are inadequate for encoding all the phenomena labeled as "multiword expressions", especially in inflectional languages. Such approaches might be successful to a large extent in the case of fixed order languages, such as English — both IDAREX and Phrase Manager are reported to have been successfully employed for such purposes (Breidt and Feldweg, 1997; Tschichold, 2000). However, they fail with languages that have richer inflection and permit more word order variations. When used for Polish, the surface processing oriented IDAREX reaches the limits of its expressiveness; Phrase Manager is inadequate for different reasons — the assumptions it is based on would require something not far from writing a complete grammar of Polish, a task to which it is not suitable due to its limitations. And on the other hand, it is much too complicated for simple multiword expressions, such as (12).

4 Previous Classifications

There are numerous classifications available in linguistic literature, and we considered three of them in turn. From the practical point of view, none of them proved to be adequate for our needs. More precisely, none of them partitioned the field of multiword expressions into manageable classes that could be handled individually by uniform mechanisms.

The classification presented by Brundage et al. (1992) approaches the whole problem from an angle similar to what is required in Phrase Manager. It is based on a study of ca. 300 English and German multiword expressions, which were divided into classes based on their syntactic constituency and the transformations they are able to undergo.

Such an approach seems to be a dead end for exactly the same reasons that Phrase Manager has been criticized above. The study was limited to 300 units, which made the whole undertaking manageable. We believe that a really extensive study would lead to an unpredictable proliferation of very similar classes, which would make the whole classification too fine-grained and unpractical for any processing purposes.

The categorization that has been examined next is the one presented by Sag et al. (2002). It consists of three categories: fixed expressions (absolutely immutable), semi-fixed expressions (strictly fixed word order, but some lexical variation is allowed), syntactically-flexible expressions (mainly decomposable idioms — cf. (8)), and institutionalized phrases (statistical idiosyncrasies). Unfortunately, such a categorization is hard to use in the case of some Polish multiword expressions. Consider this example:

(13) *Niech* *to* *szlag* *trafi!*
let it-Acc * hit-Future
'Damn it!'

It is hard to establish which of the above categories does it belong to. The only lexically variable element is *it*, which can be substituted with another noun. This would qualify the expression to be included in the second category. However, it has a very free word order (*Niech to trafi szlag!*, *Szlag niech to trafi!*, and *Niech trafi to szlag!* are all acceptable). This in turn qualifies it to the third category, but it is not a decomposable idiom, and the word order variations are not semantically justified transformations, but rather permutations, as in (12). To make matters worse, the main element — *szlag* — is a word with a very limited distribution. This intuitively makes the unit fit more into the first category of unproductive expressions. This is even more obvious considering the fact that the word order variations do not change the meaning.

Another classification was presented by Guenthner and Blanco (2004). Their categories are very numerous, and the whole undertaking suffers from the fact that they are not formally defined. It also lacks a coherent purpose – it is neither a linguistic, nor a natural language processing classification, as it tries to put very different phenomena into one bag.

The categories are sometimes more lexicographically, and sometimes more syntactically oriented. For example, on the one hand the authors distinguish compound expressions (nouns, adverbs, etc.), and on the other hand collocations. In our opinion the categories should not be considered as parts of the same classification, as members of the former category belong to the lexicon, and the latter are a purely distributional phenomenon. Therefore, in the present form, the classification has no practical use.

5 Conclusions and Further Work

We have shown that trying to provide a formal description of *all* phenomena labeled as multiword expressions as a *whole* is not possible, which becomes obvious if one goes beyond English and tries to describe multiword expressions in heavily inflectional and relatively free word order languages, such as Polish. We have also shown the inadequacy of the available classifications of multiword expressions for computational processing of such languages.

In our opinion, a successful computational description of multiword expressions requires distinguishing two groups of units: idiosyncratic from the point of view of morphosyntax and idiosyncratic from the point of view of semantics. Such a division allows for efficient use of existing tools without the need of creating a cumbersome formalism.

We believe that the practically oriented classification presented above will allow us to build robust tools for handling both types of multiword expressions, which is the aim of our further research. The immediate task is to build the syntactic preprocessor. We also plan to extend the classification to make it slightly more fine-grained, which hopefully will make even more efficient processing possible.

References

Elisabeth Breidt and Helmut Feldweg. 1997. Accessing foreign languages with COMPASS. *Machine Translation*, 12(1/2):153–174.

Jennifer Brundage, Maren Kresse, Ulrike Schwall, and Angelika Storrer. 1992. Multiword lexemes: A monolingual and contrastive typology for NLP and MT. Technical Report IWBS 232, IBM Deutschland GmbH, Institut für Wissenbasierte Systeme, Heidelberg.

Ralph Debusmann. 2004. Multiword expressions as dependency subgraphs. In *Proceedings of the ACL 2004 Workshop on Multiword Expressions: Integrating Processing*, Barcelona, Spain.

Frantz Guenthner and Xavier Blanco. 2004. Multilexemic expressions: an overview. In Christian Lèclere; Éric Laporte; Mireille Piot; Max Silberztein, editor, *Syntax, Lexis, and Lexicon-Grammar*, volume 24 of *Linguisticæ Investigationes Supplementa*, pages 239–252. John Benjamins.

Sandro Pedrazzini. 1994. *Phrase Manager: A System for Phrasal and Idiomatic Dictionaries*. Georg Olms Verlag, Hildeseim, Zürich, New York.

Gábor Prószéky and András Földes. 2005. An intelligent context-sensitive dictionary: A Polish-English comprehension tool. In *Human Language Technologies as a Challenge for Computer Science and Linguistics. 2nd Language & Technology Conference April 21–23, 2005,*, pages 386–389, Poznań, Poland.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proc. of the 3rd International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2002)*, pages 1–15, Mexico City, Mexico.

Benoît Sagot and Pierre Boullier. 2005. From raw corpus to word lattices: robust pre-parsing processing. *Archives of Control Sciences, special issue of selected papers from LTC'05*, 15(4):653–662.

Frédérique Segond and Elisabeth Breidt. 1995. IDAREX: Formal description of German and French multi-word expressions with finite state technology. Technical Report MLTT-022, Rank Xerox Research Centre, Grenoble.

Cornelia Tschichold. 2000. *Multi-word units in natural language processing*. Georg Olms Verlag, Hildeseim, Zürich, New York.

Yi Zhang, Valia Kordoni, Aline Villavicencio, and Marco Idiart. 2006. Automated multiword expression prediction for grammar engineering. In *Proceedings of the Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, pages 36–44, Sydney, Australia. Association for Computational Linguistics.

Automatic Prediction of Cognate Orthography Using Support Vector Machines

Andrea Mulloni
Research Group in Computational Linguistics
HLSS, University of Wolverhampton
MB114 Stafford Street, Wolverhampton, WV1 1SB, United Kingdom
andrea2@wlv.ac.uk

Abstract

This paper describes an algorithm to automatically generate a list of cognates in a target language by means of Support Vector Machines. While Levenshtein distance was used to align the training file, no knowledge repository other than an initial list of cognates used for training purposes was input into the algorithm. Evaluation was set up in a cognate production scenario which mimed a real-life situation where no word lists were available in the target language, delivering the ideal environment to test the feasibility of a more ambitious project that will involve language portability. An overall improvement of 50.58% over the baseline showed promising horizons.

1 Introduction

Cognates are words that have similar spelling and meaning across different languages. They account for a considerable portion of technical lexicons, and they found application in several NLP domains. Some major applications fields include relevant areas such as bilingual terminology compilation and statistical machine translation.

So far algorithms for cognate recognition have been focussing predominantly on the detection of cognate words in a text, e.g. (Kondrak and Dorr 2004). Sometimes, though, the detection of cognates in free-flowing text is rather impractical: being able to predict the possible translation in the target language would optimize algorithms that make extensive use of the Web or very large corpora, since there would be no need to scan the whole data each time in order to find the correspondent item. The proposed approach aims to look at the same problem from a totally different perspective, that is to produce an information repository about the target language that could then be exploited in order to predict how the orthography of a "possible" cognate in the target language should look like. This is necessary when no plain word list is available in the target language or the list is incomplete. The proposed algorithm merges for the first time two otherwise well-known methods, adopting a specific tagger implementation which suggests new areas of application for this tool. Furthermore, once language portability will be in place, the cognate generation exercise will allow to reformulate the recognition exercise as well, which is indeed a more straightforward one. The algorithm described in this paper is based on the assumption that linguistic mappings show some kind of regularity and that they can be exploited in order to draw a net of implicit rules by means of a machine learning approach.

Section 2 deals with previous work done on the field of cognate recognition, while Section 3 describes in detail the algorithm used for this study. An evaluation scenario will be drawn in Section 4, while Section 5 will outline the directions we intend to take in the next months.

2 Previous Work

The identification of cognates is a quite challenging NLP task. The most renowned approach to cognate recognition is to use spelling similarities between the two words involved. The most important contribution to this methodology has been given by Levenshtein (1965), who calculated the changes needed in order to transform one word into another by applying four different edit operations – match,

Proceedings of the ACL 2007 Student Research Workshop, pages 25–30,
Prague, June 2007.

substitution, insertion and deletion – which became known under the name of edit distance (ED). A good case in point of a practical application of ED is represented by the studies in the field of lexicon acquisition from comparable corpora carried out by Koehn and Knight (2002) – who expand a list of English-German cognate words by applying well-established transformation rules (e.g. substitution of *k* or *z* by *c* and of *–tät* by *–ty*, as in German *Elektizität* – English *electricity*) – as well as those that focused on word alignment in parallel corpora (e.g. Melamed (2001) and Simard et al. (1999)). Furthermore, Laviosa (2001) showed that cognates can be extremely helpful in translation studies, too.

Among others, ED was extensively used also by Mann and Yarowsky (2001), who try to induce translation lexicons between cross-family languages via third languages. Lexicons are then expanded to intra-family languages by means of cognate pairs and cognate distance. Related techniques include a method developed by Danielsson and Mühlenbock (2000), who associate two words by calculating the number of matching consonants, allowing for one mismatched character. A quite interesting spin-off was analysed by Kondrak (2004), who first highlighted the importance of genetic cognates by comparing the phonetic similarity of lexemes with the semantic similarity of the glosses.

A general overview of the most important statistical techniques currently used for cognate detection purposes was delivered by Inkpen et al. (2005), who addressed the problem of automatic classification of word pairs as cognates or false friends and analysed the impact of applying different features through machine learning techniques. In her paper, she also proposed a method to automatically distinguish between cognates and false friends, while examining the performance of seven different machine learning classifiers.

Further applications of ED include Mulloni and Pekar (2006), who designed an algorithm based on normalized edit distance aiming to automatically extract translation rules, for then applying them to the original cognate list in order to expand it, and Brew and McKelvie (1996), who used approximate string matching in order to align sentences and extract lexicographically interesting word-word pairs from multilingual corpora.

Finally, it is worth mentioning that the work done on automatic named entity transliteration often crosses paths with the research on cognate recognition. One good pointer leads to Kashani et al. (2006), who used a three-phase algorithm based on HMM to solve the transliteration problem between Arabic and English.

All the methodologies described above showed good potential, each one in its own way. This paper aims to merge some successful ideas together, as well as providing an independent and flexible framework that could be applied to different scenarios.

3 Proposed Approach

When approaching the algorithm design phase, we were faced with two major decisions: firstly, we had to decide which kind of machine learning (ML) approach should be used to gather the necessary information, secondly we needed to determine how to exploit the knowledge base gathered in the most appropriate and productive way. As it turned out, the whole work ended up to revolve around the intuition that a simple tagger could lead to quite interesting results, if only we could scale down from sentence level to word level, that is to produce a tag for single letters instead of whole words. In other words, we wanted to exploit the analogy between PoS tagging and cognate prediction: given a sequence of symbols – i.e. source language unigrams – and tags aligned with them – i.e. target language n-grams –, we aim to predict tags for more symbols. Thereby the context provided by the neighbors of a symbol and the previous tags are used as evidence to decide its tag. After an extensive evaluation of the major ML-based taggers available, we decided to opt for SVMTool, a generator of sequential taggers based on Support Vector Machines developed by Gimenez and Marquez (2004). In fact, various experiments carried out on similar software showed that SVMTool was the most suitable one for the type of data being examined, mainly because of its flexible approach to our input file. Also, SVMTool allows to define context by providing an adjustable sliding window for the extraction of features. Once the model was trained, we went on to create the most orthographically probable cognate in the target language. The following sections exemplify the cognate creation algorithm, the learning step and the exploitation of the information gathered.

3.1 Cognate Creation Algorithm

Figure 1 shows the cognate creation algorithm in detail.

Input: *C1*, a list of English-German cognate pairs {*L1,L2*}; *C2*, a test file of cognates in *L1*
Output: *AL*, a list of artificially constructed cognates in the target language

1 **for** *c* in *C1* **do**:
2 determine the edit operations to arrive from *L1* to *L2*
3 use the edit operations to produce a formatted training file for the SVM tagger
4 **end**
5 Learn orthographic mappings between *L1* and *L2* (L1 unigram = instance, L2 n-gram = category)
6 Align all words of the test file vertically in a letter-by-letter fashion (unigram = instance)
7 Tag the test file with the SVM tagger
8 Group the tagger output into words and produce a list of cognate pairs

Figure 1. The cognate creation algorithm.

Determination of the Edit Operations

The algorithm takes as input two distinct cognate lists, one for training and one for testing purposes. It is important to note that the input languages need to share the same alphabet, since the algorithm is currently still depending on edit distance. Future developments will allow for language portability, which is already matter of study. The first sub-step (Figure 1, Line 2) deals with the determination of the edit operations and its association with the cognate pair, as shown in Figure 2. The four options provided by edit distance, as described by Levenshtein (1965), are Match, Substitution, Insertion and Deletion.

```
toilet/toilette
t    |o    |i    |l    |e    |t    |   |
t    |o    |i    |l    |e    |t    |t  |e
MATCH|MATCH|MATCH|MATCH|MATCH|MATCH|INS|INS

tractor/traktor
t    |r    |a    |c    |t    |o    |r
t    |r    |a    |k    |t    |o    |r
MATCH|MATCH|MATCH|SUBST|MATCH|MATCH|MATCH

absolute/absolut
a    |b    |s    |o    |l    |u    |t    |e
a    |b    |s    |o    |l    |u    |t    |
MATCH|MATCH|MATCH|MATCH|MATCH|MATCH|MATCH|DEL
```

Figure 2. Edit operation association

Preparation of the Training File

This sub-step (Figure 1, Line 3) turned out to be the most challenging task, since we needed to produce the input file that offered the best layout possible for the machine learning module. We first tried to insert several empty slots between letters in the source language file, so that we could cope with maximally two subsequent insertions. While all words are in lower case, we identified the spaces with a capital X, which would have allowed us to subsequently discard it without running the risk to delete useful letters in the last step of the algorithm. The choice of manipulating the source language file was supported by the fact that we were aiming to limit the features of the ML module to 27 at most, that is the letters of the alphabet from "a" to "z" plus the upper case "X" meaning blank. Nonetheless, we soon realized that the space feature outweighed all other features and biased the output towards shorter words. Also, the input word was so interspersed that it did not allow the learning machine to recognize recurrent patterns. Further empirical activity showed that far better results could be achieved by sticking to the original letter sequence in the source word and allow for an indefinite number of feature to be learned. This was implemented by grouping letters on the basis of their edit operation relation to the source language. Figure 3 exemplifies a typical situation where insertions and deletions are catered for.

```
START START          START START
a a                  m m
b b                  a a
i i                  c k
o o                  r ro
g g                  o e
e e                  e e
n n                  c k
e e                  o o
t t                  n n
i i                  o o
c X                  m m
a X                  i is
l s                  c ch
l c                  . END
y h
. END
```

Figure 3. Layout of the training entries macroeconomic/makrooekonomisch and abiogenetically/abiogenetisch, showing insertions and deletions

As shown in Figure 3, German diacritics have been substituted by their extended version – i.e. "ö" as been rendered as "oe": this was due to the inability of SVMTool to cope with diacritics. Figure 3 also shows how insertions and deletions

were treated. This design choice caused a non-foreseeable number of features to be learned by the ML module. While apparently a negative issue that could cause data to be too sparse to be relevant, we trusted our intuition that the feature growing graph would just flat out after an initial spike, that is the number of insertion edits would not produce an explosion of source/target n-gram equivalents, but only a short expansion to the original list of mapping pairings. This proved to be correct by the evaluation phase described below.

Learning Mappings Across Languages

Once the preliminary steps had been taken care of, the training file was passed on to SVMTlearn, the learning module of SVMTool. At this point the focus switches over to the tool itself, which learns regular patterns using Support Vector Machines and then uses the information gathered to tag any possible list of words (Figure 1, Line 5). The tool chooses automatically the best scoring tag, but – as a matter of fact – it calculates up to 10 possible alternatives for each letter and ranks them by probability scores: in the current paper the reported results were based on the best scoring "tag", but the algorithm can be easily modified in order to accommodate the outcome of the combination of all 10 scores. As it will be shown later in Section 4, this is potentially of great interest if we intend to work in a cognate creation scenario.

As far the last three steps of the algorithm are concerned, they are closely related to the practical implementation of our methodology, hence they will be described extensively in Section 4.

4 Evaluation

In order to evaluate the cognate creation algorithm, we decided to set up a specific evaluation scenario where possible cognates needed to be identified but no word list to choose from existed in the target language. Specifically, we were interested in producing the correct word in the target language, starting from a list of possible cognates in the source language. An alternative evaluation setting could have been based on a scenario which included a scrambling and matching routine, but after the good results showed by Mulloni and Pekar (2006), we thought that yet a different environment would have offered more insight into the field. Also, we wanted to evaluate the actual strength of our approach, in order to decide if future work should be heading this way.

4.1 Data

The method was evaluated on an English-German cognate list including 2105 entries. Since we wanted to keep as much data available for testing as possible, we decided to split the list in 80% training (1683 entries) and 20% (422 entries) testing.

4.2 Task Description

The list used for training/testing purposes included cognates only. Therefore, the optimal outcome would have been a word in the target language that perfectly matched the cognate of the corresponding source language word in the original file. The task was therefore a quite straightforward one: train the SVM tagger using the training data file and – starting from a list of words in the source language (English) – produce a word in the target language (German) that looked as close as possible to the original cognate word. Also, we counted all occurrences where no changes across languages took place – i.e. the target word was spelled in the very same way as the source word – and we set this number as a baseline for the assessment of our results.

Preparation of the Training and Test Files

The training file was formatted as described in Section 3.1. In addition to that, the training and test files featured a START/START delimiter at the beginning of the word and ./END delimiter at the end of it (Figure 1, Line 6).

Learning Parameters

Once formatting was done, the training file was passed on to SVMTlearn. Notably, SVMTool comes with a standard configuration: for the purpose of this exercise we decided to keep most of the standard default parameters, while tuning only the settings related to the definition of the feature set. Also, because of the choices made during the design of the training file – i.e. to stick to a strict linear layout in the *L1* word – we felt that a rather small context window of 5 with the core position set to 2 – that is, considering a context of 2 features before and 2 features after the feature currently examined – could offer a good trade-off between accuracy and acceptable working times. Altogether 185 features were learnt, which confirmed the intuition mentioned in Section 3.1. Furthermore, when considering the feature definition, we decided to stick to unigrams, bigrams and trigrams, even if

up to five-grams were obviously possible. Notably, the configuration file pictured below shows how a Model 0 and a global left-right-left tagging option were applied. Both choices were made after an extensive empirical observation of several model/direction combinations. This file is highly configurable and offers a vast range of possible combinations. Future activities will concentrate to a greater extent on the experimentations of other possible configuration scenarios in order to find the tuning that performs best. Gimenez and Marquez (2004) offer a detailed description of the models and all available options, as well as a general introduction to the use of SVMtool, while Figure 4 shows the feature set used to learn mappings from a list of English/German cognate pairs.

```
#ambiguous-right [default]
A0k = w(-2) w(-1) w(0) w(1) w(2) w(-2,-1)
w(-1,0) w(0,1) w(1,2) w(-1,1) w(-2,2)
w(-2,1) w(-1,2) w(-2,0) w(0,2) w(-2,-1,0)
w(-2,-1,1) w(-2,-1,2) w(-2,0,1) w(-2,0,2)
w(-1,0,1) w(-1,0,2) w(-1,1,2) w(0,1,2) p(-2)
p(-1) p(0) p(1) p(2) p(-2,-1) p(-1,0) p(0,1)
p(1,2) p(-1,1) p(-2,2) p(-2,1) p(-1,2)
p(-2,0) p(0,2) p(-2,-1,0) p(-2,-1,1)
p(-2,-1,2) p(-2,0,1) p(-2,0,2) p(-1,0,1)
p(-1,0,2) p(-1,1,2) p(0,1,2) k(0) k(1) k(2)
m(0) m(1) m(2)
```

Figure 4. Feature set for known words (A0k). The same feature set is used for unknown words (A0u), as well.

Tagging of the Test File and Cognate Generation

Following the learning step, a tagging routine was invoked, which produced the best scoring output for every single line – i.e. letter or word boundary – of the test file, which now looked very similar to the file we used for training (Figure 1, Line 7). At this stage, we grouped test instances together to form words and associated each *L1* word with its newly generated counterpart in *L2* (Figure 1, Line 8).

4.3 Results

The generated words were then compared with the words included in the original cognate file.

When evaluating the results we decided to split the data into three classes, rather than two: "Yes" (correct), "No" (incorrect) and "Very Close". The reason why we chose to add an extra class was that when analysing the data we noticed that many important mappings were correctly detected, but the word was still not perfect because of minor orthographic discrepancies that the tagging module did get right in a different entry. In such cases we felt that more training data would have produced a stronger association score that could have eventually led to a correct output. Decisions were made by an annotator with a well-grounded knowledge of Support Vector Machines and their behaviour, which turned out to be quite useful when deciding which output should be classified as "Very Close". For fairness reasons, this extra class was added to the "No" class when delivering the final results. Examples of the "Very Close" class are reported in Table 1.

Original EN	Original DE	Output DE
majestically	majestatetisch	majestisch
setting	setzend	settend
machineries	maschinerien	machinerien
naked	nakkt	nackt
southwest	suedwestlich	suedwest

Table 1. Examples of the class "Very Close".

In Figure 5 we show the accuracy of the SVM-based cognate generation algorithm versus the baseline, adding the "Very Close" class to both the "Yes" class (correct) and the "No" class (incorrect).

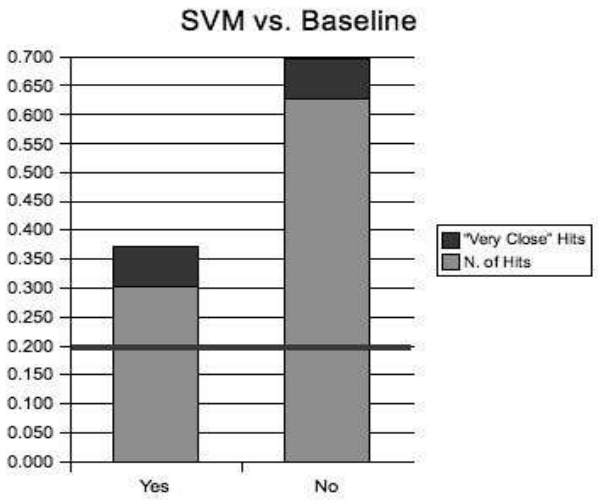


Figure 5. Accuracy of the SVM-based algorithm vs. the baseline (blue line).

The test file included a total of 422 entries, with 85 orthographically identical entries in *L1* and *L2* (baseline). The SVM-based algorithm managed to produce 128 correct cognates, making errors in 264

cases. The "Very Close" class was assigned to 30 entries. Figure 5 shows that 30.33% of the total entries were correctly identified, while an increase of 50.58% over the baseline was achieved.

5 Conclusions and Future Work

In this paper we proposed an algorithm for the automatic generation of cognates from two different languages sharing the same alphabet. An increase of 50.58% over the baseline and a 30.33% of overall accuracy were reported. Even if accuracy is rather poor, if we consider that no knowledge repository other than an initial list of cognates was available, we feel that the results are still quite encouraging.

As far as the learning module is concerned, future ameliorations will focus on the fine tuning of the features used by the classifier as well as on the choice of the model, while main research activities are still concerned with the development of a methodology allowing for language portability: as a matter of fact, n-gram co-occurrencies are currently being investigated as a possible alternative to Edit Distance.

References

Chris Brew and David McKelvie. 1996. Word-Pair Extraction for Lexicography. *Proceedings of the Second International Conference on New Methods in Language Processing,* 45-55.

Pernilla Danielsson and Katarina Muehlenbock. 2000. Small but Efficient: The Misconception of High-Frequency Words in Scandinavian Translation. *Proceedings of the 4th Conference of the Association for Machine Translation in the Americas on Envisioning Machine Translation in the Information Future*, 158-168.

Jesus Gimenez and Lluis Marquez. 2004. SVMTool: A General POS Tagger Generator Based on Support Vector Machines. *Proceedings of LREC '04*, 43-46.

Diana Inkpen, Oana Frunza and Grzegorz Kondrak. 2005. Automatic Identification of Cognates and False Friends in French and English. *Proceedings of the International Conference Recent Advances in Natural Language Processing*, 251-257.

Mehdi M. Kashani, Fred Popowich, and Fatiha Sadat. 2006. Automatic Translitteration of Proper Nouns from Arabic to English. *The Challenge of Arabic For NLP/MT*, 76-84.

Philipp Koehn and Kevin Knight. 2002. Estimating Word Translation Probabilities From Unrelated Monolingual Corpora Using the EM Algorithm. *Proceedings of the 17th AAAI conference*, 711-715.

Grzegorz Kondrak. 2004. Combining Evidence in Cognate Identification. *Proceedings of Canadian AI 2004: 17th Conference of the Canadian Society for Computational Studies of Intelligence*, 44-59.

Grzegorz Kondrak and Bonnie J. Dorr. 2004. Identification of confusable drug names. *Proceedings of COLING 2004: 20th International Conference on Computational LInguistics*, 952-958.

Sara Laviosa. 2001. *Corpus-based Translation Studies: Theory, Findings, Applications.* Rodopi, Amsterdam.

Vladimir I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 163(4):845-848.

Gideon S. Mann and David Yarowsky. 2001. Multipath Translation Lexicon Induction via Bridge Languages. *Proceedings of NAACL 2001: 2nd Meeting of the North American Chapter of the Association for Computational Linguistics*, 151-158.

I. Dan Melamed. 1999. Bitext Maps and Alignment via Pattern Recognition. *Computational Linguistics*, 25(1):107-130.

I. Dan Melamed. 2001. *Empirical Methods for Exploiting Parallel Texts.* MIT Press, Cambridge, MA.

Andrea Mulloni and Viktor Pekar. 2006. Automatic Detection of Orthographic Cues for Cognate Recognition. *Proceedings of LREC '06, 2387-2390.*

Michel Simard, George F. Foster and Pierre Isabelle. 1992. Using Cognates to Align Sentences in Bilingual Corpora. *Proceedings of the 4th International Conference on Theoretical and Methodological Issues in Machine Translation*, Montreal, Canada, 67-81.

Exploiting Structure for Event Discovery Using the MDI Algorithm

Martina Naughton
School of Computer Science & Informatics
University College Dublin
Ireland
martina.naughton@ucd.ie

Abstract

Effectively identifying events in unstructured text is a very difficult task. This is largely due to the fact that an individual event can be expressed by several sentences. In this paper, we investigate the use of clustering methods for the task of grouping the text spans in a news article that refer to the same event. The key idea is to cluster the sentences, using a novel distance metric that exploits regularities in the sequential structure of events within a document. When this approach is compared to a simple bag of words baseline, a statistically significant increase in performance is observed.

1 Introduction

Accurately identifying events in unstructured text is an important goal for many applications that require natural language understanding. There has been an increased focus on this problem in recent years. The Automatic Content Extraction (ACE) program[1] is dedicated to developing methods that automatically infer meaning from language data. Tasks include the detection and characterisation of Entities, Relations, and Events. Extensive research has been dedicated to entity recognition and binary relation detection with significant results (Bikel et al., 1999). However, event extraction is still considered as one of the most challenging tasks because an individual event can be expressed by several sentences (Xu et al., 2006).

In this paper, we primarily focus on techniques for identifying events within a given news article. Specifically, we describe and evaluate clustering methods for the task of grouping sentences in a news article that refer to the same event. We generate sentence clusters using three variations of the well-documented Hierarchical Agglomerative Clustering (HAC) (Manning and Schütze, 1999) as a baseline for this task. We provide convincing evidence suggesting that inherent structures exist in the manner in which events appear in documents. In Section 3.1, we present an algorithm which uses such structures during the clustering process and as a result a modest increase in accuracy is observed.

Developing methods capable of identifying all types of events from free text is challenging for several reasons. Firstly, different applications consider different types of events and with different levels of granularity. A change in state, a horse winning a race and the race meeting itself can be considered as events. Secondly, interpretation of events can be subjective. How people understand an event can depend on their knowledge and perspectives. Therefore in this current work, the type of event to extract is known in advance. As a detailed case study, we investigate event discovery using a corpus of news articles relating to the recent Iraqi War where the target event is the *"Death"* event type. Figure 1 shows a sample article depicting such events.

The remainder of this paper is organised as follows: We begin with a brief discussion of related work in Section 2. We describe our approach to Event Discovery in Section 3. Our techniques are experimentally evaluated in Section 4. Finally, we conclude with a discussion of experimental observations and opportunities for future work in Section 5.

2 Related Research

The aim of Event Extraction is to identify any instance of a particular class of events in a natural

[1] http://www.nist.gov/speech/tests/ace/

Proceedings of the ACL 2007 Student Research Workshop, pages 31–36,
Prague, June 2007.

World News

Insurgents Kill 17 in Iraq

In Tikrit, gunmen killed 17 Iraqis as they were heading to work Sunday at a U.S. military facility. Capt. Bill Coppernoll, said insurgents fired at several buses of Iraqis from two cars.

. . . . . .

Elsewhere, an explosion at a market in Baqubah, about 30 miles north of Baghdad late Thursday. The market was struck by mortar bombs according to U.S. military spokesman Sgt. Danny Martin.

. . . . . .

Figure 1: Sample news article that describes multiple events.

language text, extract the relevant arguments of the event, and represent the extracted information into a structured form (Grishman, 1997). The types of events to extract are known in advance. For example, *"Attack"* and *"Death"* are possible event types to be extracted. Previous work in this area focuses mainly on linguistic and statistical methods to extract the relevant arguments of a event type. Linguistic methods attempt to capture linguists knowledge in determining constraints for syntax, morphology and the disambiguation of both. Statistical methods generate models based in the internal structures of sentences, usually identifying dependency structures using an already annotated corpus of sentences. However, since an event can be expressed by several sentences, our approach to event extraction is as follows: First, identify all the sentences in a document that refer to the event in question. Second, extract event arguments from these sentences and finally represent the extracted information of the event in a structured form.

Particularly, in this paper we focus on clustering methods for grouping sentences in an article that discuss the same event. The task of clustering similar sentences is a problem that has been investigated particularly in the area of text summarisation. In SimFinder (Hatzivassiloglou et al., 2001), a flexible clustering tool for summarisation, the task is defined as finding text units (sentences or paragraphs) that contain information about a specific subject. However, the text features used in their similarity metric are selected using a Machine Learning model.

3 Identifying Events within Articles

We treat the task of grouping together sentences that refer to the same event(s) as a clustering problem. As a baseline, we generate sentence clusters using average-link, single-link and complete-link Hierarchical Agglomerative Clustering. HAC initially assigns each data point to a singleton cluster, and repeatedly merges clusters until a specified termination criteria is satisfied (Manning and Schütze, 1999). These methods require a similarity metric between two sentences. We use the standard cosine metric over a bag-of-words encoding of each sentence. We remove stopwords and stem each remaining term using the Porter stemming algorithm (Porter, 1997). Our algorithms begin by placing each sentence in its own cluster. At each iteration we merge the two closest clusters. A fully-automated approach must use some termination criteria to decide when to stop clustering. In experiments presented here, we adopt two manually supervised methods to set the desired number of clusters (k): "correct" k and "best" k. "Correct" sets k to be the actual number of events. This value was obtained during the annotation process (see Section 4.1). "Best" tunes k so as to maximise the quality of the resulting clusters.

3.1 Exploiting Article Structure

Our baseline ignores an important constraint on the event associated with each sentence: the position of the sentence within the document. Documents consist of sentences arranged in a linear order and nearby sentences in terms of this ordering typically refer to the same topic (Zha, 2002). Similarly we assume that adjacent sentences are more likely to refer to the same event, later sentences are likely to introduce new events, etc. In this Section, we describe an algorithm that exploits this document structure during the sentence clustering process.

The basic idea is to learn a model capable of capturing document structure, i.e. the way events are reported. Each document is treated as a sequence of labels (1 label per sentence) where each label represents the event(s) discussed in that sentence. We define four generalised event label types: N, represents a new event sentence; C, represents a continuing event sentence (i.e. it discusses the same event as the preceding sentence); B, represents a back-reference to an earlier event; X, represents a sentence that does not reference an event. This model takes the form of a Finite State Automaton (FSA) where:

- *States* correspond to event labels.
- *Transitions* correspond to adjacent sentences that mention the pair of events.

More formally, E = (S, s_0, F, L, T) is a model where S is the set of states, $s_0 \in S$ is the initial state, $F \subseteq S$ is the set of final states, L is the set of edge labels and $T \subseteq (S \times L) \times S$ is the set of transitions. We note that it is the responsibility of the learning algorithm to discover the correct number of states.

We treat the task of discovering an event model as that of learning a regular grammar from a set of positive examples. Following Golds research on learning regular languages (Gold, 1967), the problem has received significant attention. In our current experiments, we use Thollard et al's MDI algorithm (Thollard et al., 2000) for learning the automaton. MDI has been shown to be effective on a wide range of tasks, but it must be noted that any grammar inference algorithm could be substituted.

To estimate how much sequential structure exists in the sentence labels, the document collection was randomly split into training and test sets. The automaton produced by MDI was learned using the training data, and the probability that each test sequence was generated by the automaton was calculated. These probabilities were compared with those of a set of random sequences (generated to have the same distribution of length as the test data). The probabilities of event sequences from our dataset and the randomly generated sequences are shown in Figure 2. The test and random sequences are sorted by probability. The vertical axis shows the rank in each sequence and the horizontal axis shows the negative log probability of the sequence at each rank. The data suggests that the documents are indeed structured, as real document sequences tend to be much more likely under the trained FSA than randomly generated sequences.

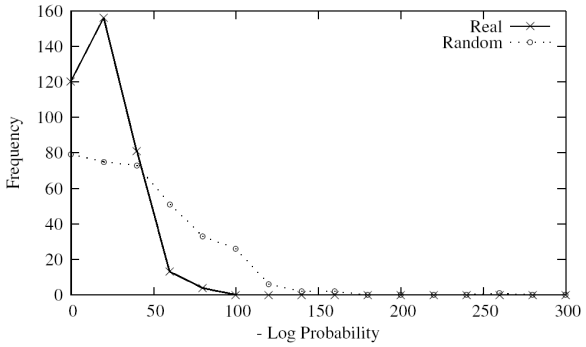


Figure 2: Distribution in the probability that actual and random event sequences are generated by the automaton produced by MDI.

We modify our baseline clustering algorithm to utilise the structural information omitted by the automaton as follows: Let $L(c_1, c_2)$ be a sequence of labels induced by merging two clusters c_1 and c_2. If $P(\mathrm{L}(c_1, c_2))$ is the probability that sequence $L(c_1, c_2)$ is accepted by the automaton, and let $\cos(c_1, c_2)$ be the cosine distance between c_1 and c_2. We can measure the similarity between c_1 and c_2 as:

$$SIM(c_1, c_2) = \cos(c_1, c_2) \times P(\mathrm{L}(c_1, c_2)) \quad (1)$$

Let r be the number of clusters remaining. Then there are $\frac{r(r-1)}{2}$ pairs of clusters. For each pair of clusters c_1,c_2 we generate the resulting sequence of labels that would result if c_1 and c_2 were merged. We then input each label sequence to our trained FSA to obtain the probability that it is generated by the automaton. At each iteration, the algorithm proceeds by merging the most similar pair according to this metric. Figure 3 illustrates this process in more detail. To terminate the clustering process, we adopt either the "correct" k or "best" k halting criteria described earlier.

4 Experiments

4.1 Experimental Setup

In our experiments, we used a corpus of news articles which is a subset of the Iraq Body Count (IBC)

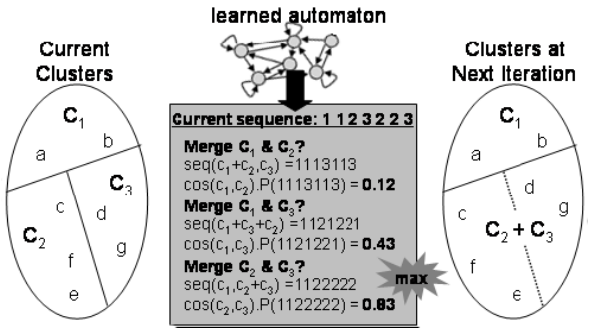


Figure 3: The sequence-based clustering process.

dataset[2]. This is an independent public database of media-reported civilian deaths in Iraq resulting directly from military attack by the U.S. forces. Casualty figures for each event reported are derived solely from a comprehensive manual survey of online media reports from various news sources. We obtained a portion of their corpus which consists of 342 new articles from 56 news sources. The articles are of varying size (average sentence length per document is 25.96). Most of the articles contain references to multiple events. The average number of events per document is 5.09. Excess HTML (image captions etc.) was removed, and sentence boundaries were identified using the Lingua::EN::Sentence perl module available from CPAN[3].

To evaluate our clustering methods, we use the definition of precision and recall proposed by (Hess and Kushmerick, 2003). We assign each pair of sentences into one of four categories: (i) clustered together (and annotated as referring to the same event); (ii) not clustered together (but annotated as referring to the same event); (iii) incorrectly clustered together; (iv) correctly not clustered together. Precision and recall are thus found to be computed as $P = \frac{a}{a+c}$ and $R = \frac{a}{a+b}$, and $F1 = \frac{2PR}{P+R}$.

The corpus was annotated by a set of ten volunteers. Within each article, events were uniquely identified by integers. These values were then mapped to one of the four label categories, namely "N", "C", "X", and "B". For instance, sentences describing previously unseen events were assigned a new integer. This value was mapped to the label category "N" signifying a new event. Similarly, sentences referring to events in a preceding sentence were assigned the same integer identifier as that assigned to the preceding sentence and mapped to the label category "C". Sentences that referenced an event mentioned earlier in the document but not in the preceding sentence were assigned the same integer identifier as that sentence but mapped to the label category "B". Furthermore, If a sentence did not refer to any event, it was assigned the label 0 and was mapped to the label category "X". Finally, each document was also annotated with the distinct number of events reported in it.

In order to approximate the level of inter-annotation agreement, two annotators were asked to annotate a disjoint set of 250 documents. Inter-rater agreements were calculated using the kappa statistic that was first proposed by (Cohen, 1960). This measure calculates and removes from the agreement rate the amount of agreement expected by chance. Therefore, the results are more informative than a simple agreement average (Cohen, 1960; Carletta, 1996). Some extensions were developed including (Cohen, 1968; Fleiss, 1971; Everitt, 1968; Barlow et al., 1991). In this paper the methodology proposed by (Fleiss, 1981) was implemented. Each sentence in the document set was rated by the two annotators and the assigned values were mapped into one of the four label categories ("N", "C", "X", and "B"). For complete instructions on how kappa was calculated, we refer the reader to (Fleiss, 1981). Using the annotated data, a kappa score of 0.67 was obtained. This indicates that the annotations are somewhat inconsistent, but nonetheless are useful for producing tentative conclusions.

To determine why the annotators were having difficulty agreeing, we calculated the kappa score for each category. For the "N", "C" and "X" categories, reasonable scores of 0.69, 0.71 and 0.72 were obtained respectively. For the "B" category a relatively poor score of 0.52 was achieved indicating that the raters found it difficult to identify sentences that referenced events mentioned earlier in the document. To illustrate the difficulty of the annotation task an example where the raters disagreed is depicted in Figure 4. The raters both agreed when assigning labels to sentence 1 and 2 but disagreed when assigning a label to Sentence 23 . In order to correctly annotate this sentence as referring to the event de-

[2]http://iraqbodycount.org/

[3]http://cpan.org/

Sentence 1: A suicide attacker set off a bomb that tore through a funeral **tent** jammed with Shiite mourners Thursday.
Rater 1: label=1. **Rater 2:** label=1
Sentence 2: The explosion, in a working class neighbourhood of **Mosul**, destroyed **the tent** killing **nearly 50 people**.
Rater 1: label=1. **Rater 2:** label=1.
.....
Sentence 23: At the hospital of this **northern city**, doctor Saher Maher said that **at least 47 people** were killed.
Rater 1: label=1. **Rater 2: label=2**.

Figure 4: Sample sentences where the raters disagreed.

Algorithm	a-link	c-link	s-link
BL(correct k)	40.5 %	39.2%	39.6%
SEQ(correct k)	47.6%*	45.5%*	44.9%*
BL(best k)	52.0%	48.2%	50.9%
SEQ(best k)	61.0%*	56.9%*	58.6%*

Table 1: % F1 achieved using average-link (a-link), complete-link (c-link) and single-link (s-link) variations of the baseline and sequence-based algorithms when the correct and best k halting criteria are used. Scores marked with * are statistically significant to a confidence level of 99%.

scribe in sentence 1 and 2, the rater have to resolve that "the northern city" is referring to "Mosul" and that "nearly 50" equates to "at least 47". These and similar ambiguities in written text make such an annotation task very difficult.

4.2 Results

We evaluated our clustering algorithms using the F1 metric. Results presented in Table 1 were obtained using 50:50 randomly selected train/test splits averaged over 5 runs. For each run, the automaton produced by MDI was generated using the training set and the clustering algorithms were evaluated using the test set. On average, the sequence-based clustering approach achieves an 8% increase in F1 when compared to the baseline. Specifically the average-link variation exhibits the highest F1 score, achieving 62% when the "best" k termination method is used.

It is important to note that the inference produced by the automaton depends on two values: the threshold α of the MDI algorithm and the amount of label sequences used for learning. The closer α is to 0, the more general the inferred automaton becomes. In an attempt to produce a more general automaton, we chose $\alpha = 0.1$. Intuitively, as more training data is used to train the automaton, more accurate inferences are expected. To confirm this we calculated the %F1 achieved by the average-link variation of the method for varying levels of training data. Overall, an improvement of approx. 5% is observed as the percentage training data used is increased from 10% to 90%.

5 Discussion

Accurately identifying events in unstructured text is a very difficult task. This is partly because the description of an individual event can spread across several sentences. In this paper, we investigated the use of clustering for the task of grouping sentences in a document that refer to the same event. However, there are limitations to this approach that need to be considered. Firstly, results presented in Section 4.2 suggest that the performance of the clusterer depends somewhat on the chosen value of k (i.e. the number of events in the document). This information is not readily available. However, preliminary analysis presented in (Naughton et al., 2006) indicate that is possible to estimate this value with reasonable accuracy. Furthermore, promising results are observed when this estimated value is used halt the clustering process. Secondly, labelled data is required to train the automation used by our novel clustering method. Evidence presented in Section 4.1 suggests that reasonable inter-annotation agreement for such an annotation task is difficult to achieve. Nevertheless, clustering allows us to take into account that the manner in which events are described is not always linear. To assess exactly how beneficial this is, we are currently treating this problem as a text segmentation task. Although this is a

crude treatment of the complexity of written text, it will help us to approximate the benefit (if any) of applying clustering-based techniques to this task.

In the future, we hope to further evaluate our methods using a larger dataset containing more event types. We also hope to examine the interesting possibility that inherent structures learned from documents originating from one news source (e.g. Aljazeera) differ from structures learned using documents originating from another source (e.g. Reuters). Finally, a single sentence often contains references to multiple events. For example, consider the sentence "These two bombings have claimed the lives of 23 Iraqi soldiers". Our algorithms assume that each sentence describes just one event. Future work will focus on developing methods to automatically recognise such sentences and techniques to incorporate them into the clustering process.

Acknowledgements. This research was supported by the Irish Research Council for Science, Engineering & Technology (IRCSET) and IBM under grant RS/2004/IBM/1. The author also wishes to thank Dr. Joe Carthy and Dr. Nicholas Kushmerick for their helpful discussions.

References

W. Barlow, N. Lai, and S. Azen. 1991. A comparison of methods for calculating a stratified kappa. *Statistics in Medicine*, 10:1465–1472.

Daniel Bikel, Richard Schwartz, and Ralph Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231.

Jean Carletta. 1996. Assessing agreement on classification tasks: the kappa statistic. *Computational Linguistics*, 22:249–254.

Jacob Cohen. 1960. A coeficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Jacob Cohen. 1968. Weighted kappa: Nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70.

B.S. Everitt. 1968. Moments of the statistics kappa and the weighted kappa. *The British Journal of Mathematical and Statistical Psychology*, 21:97–103.

J.L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76.

J.L. Fleiss, 1981. *Statistical methods for rates and proportions*, pages 212–36. John Wiley & Sons.

E. Mark Gold. 1967. Grammar identification in the limit. *Information and Control*, 10(5):447–474.

Ralph Grishman. 1997. Information extraction: Techniques and challenges. In *Proceedings of the seventh International Message Understanding Conference*, pages 10–27.

Vasileios Hatzivassiloglou, Judith Klavans, Melissa Holcombe, Regina Barzilay, Min-Yen Kan, and Kathleen McKeown. 2001. SIMFINDER: A flexible clustering tool for summarisation. In *Proceedings of the NAACL Workshop on Automatic Summarisation, Association for Computational Linguistics*, pages 41–49.

Andreas Hess and Nicholas Kushmerick. 2003. Learning to attach semantic metadata to web services. In *Proceedings of the International Semantic Web Conference (ISWC 2003)*, pages 258–273. Springer.

Christopher Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Martina Naughton, Nicholas Kushmerick, and Joseph Carthy. 2006. Event extraction from heterogeneous news sources. In *Proceedings of the AAAI Workshop Event Extraction and Synthesis*, pages 1–6, Boston.

Martin Porter. 1997. An algorithm for suffix stripping. *Readings in Information Retrieval*, pages 313–316.

Franck Thollard, Pierre Dupont, and Colin de la Higuera. 2000. Probabilistic DFA inference using Kullback-Leibler divergence and minimality. In *Proceedings of the 17th International Conference on Machine Learning*, pages 975–982. Morgan Kaufmann, San Francisco.

Feiyu Xu, Hans Uszkoreit, and Hong Li. 2006. Automatic event and relation detection with seeds of varying complexity. In *Proceedings of the AAAI Workshop Event Extraction and Synthesis*, pages 12–17, Boston.

Hongyuan Zha. 2002. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in Information Retrieval*, pages 113–120, New York, NY. ACM Press.

Kinds of Features for Chinese Opinionated Information Retrieval

Taras Zagibalov
Department of Informatics
University of Sussex
United Kingdom
T.Zagibalov@sussex.ac.uk

Abstract

This paper presents the results of experiments in which we tested different kinds of features for retrieval of Chinese opinionated texts. We assume that the task of retrieval of opinionated texts (OIR) can be regarded as a subtask of general IR, but with some distinct features. The experiments showed that the best results were obtained from the combination of character-based processing, dictionary look up (maximum matching) and a negation check.

1 Introduction

The extraction of opinionated information has recently become an important research topic. Business and governmental institutions often need to have information about how their products or actions are perceived by people. Individuals may be interested in other people's opinions on various topics ranging from political events to consumer products.

At the same time globalization has made the whole world smaller, and a notion of the world as a 'global village' does not surprise people nowadays. In this context we assume information in Chinese to be of particular interest as the Chinese world (the mainland China, Taiwan, Hong Kong, Singapore and numerous Chinese communities all over the world) is getting more and more influential over the world economy and politics.

We therefore believe that a system capable of providing access to opinionated information in other languages (especially in Chinese) might be of great use for individuals as well as for institutions involved in international trade or international relations.

The sentiment classification experiments presented in this paper were done in the context of Opinionated Information Retrieval which is planned to be a module in a Cross-Language Opinion Extraction system (CLOE). The main goal of this system is to provide access to opinionated information on any topic ad-hoc in a language different to the language of a query.

To implement the idea the CLOE system which is the context for the experiments described in the paper will consist of four main modules:

1. Query translation
2. Opinionated Information Retrieval
3. Opinionated Information Extraction
4. Results presentation

The OIR module will process complex queries consisting of a word sequence indicating a topic and sentiment information. An example of such a query is: "Asus laptop + OPINIONS", another, more detailed query, might be "Asus laptop + POSITIVE OPINIONS".

Another possible approach to the architecture of the CLOE system would be to implement the processing as a pipeline consisting, first, of using IR to retrieve certain articles relevant to the topic followed by second stage of classifying them according to sentiment polarity. But such an approach probably would be too inefficient, as the search will produce a lot of irrelevant results (containing no opinionated information).

Proceedings of the ACL 2007 Student Research Workshop, pages 37–42,
Prague, June 2007.

2 Chinese NLP and Feature Selection Problem

One of the central problems in Chinese NLP is what the basic unit[1] of processing should be. The problem is caused by a distinctive feature of the Chinese language - absence of explicit word boundaries, while it is widely assumed that a word is of extreme importance for any NLP task. This problem is also crucial for the present study as the basic unit definition affects the kinds of features to be used.

In this study we use a mixed approached, based both on words (tokens consisting of more than one character) and characters as basic units. It is also important to note, that we use notion of words in the sense of Vocabulary Word as it was stated by Li (2000). This means that we use only tokens that are listed in a dictionary, and do not look for all words (including grammar words).

3 Related Work

Processing of subjective texts and opinions has received a lot of interest recently. Most of the authors traditionally use a classification-based approach for sentiment extraction and sentiment polarity detection (for example, Pang et al. (2002), Turney (2002), Kim and Hovy (2004) and others), however, the research described in this paper uses the information retrieval (IR) paradigm which has also been used by some researchers.

Several sentiment information retrieval models were proposed in the framework of probabilistic language models by Eguchi and Lavrenko (2006). The setting for the study was a situation when a user's query specifies not only terms expressing a certain topic and also specifies a sentiment polarity of interest in some manner, which makes this research very similar to the present one. However, we use sentiment scores (not probabilistic language models) for sentiment retrieval (see Section 4.1). Dave et al. (Dave et al., 2003) described a tool for sifting through and synthesizing product reviews, automating the sort of work done by aggregation sites or clipping services. The authors of this paper used probability scores of arbitrary-length substrings that provide optimal classification. Unlike this approach we use a combination of sentiment weights of characters and words (see Section 4).

Recently several works on sentiment extraction from Chinese texts were published. In a paper by Ku et al. (2006a) a dictionary-based approach was used in the context of sentiment extraction and summarization. The same authors describe a corpus of opinionated texts in another paper (2006b). This paper also defines the annotations for opinionated materials. Although we use the same dictionary in our research, we do not use only word-based approach to sentiment detection, but we also use scores for characters obtained by processing the dictionary as a training corpus (see Section 4).

4 Experiments

In this paper we present the results of sentiment classification experiments in which we tested different kinds of features for retrieval of Chinese opinionated information.

As stated earlier (see Section 1), we assume that the task of retrieval of opinionated texts (OIR) can be regarded as a subtask of general IR with a query consisting of two parts: (1) words indicating topic and (2) a semantic class indicating sentiment (OPINIONS). The latter part of the query cannot be specified in terms that can be instantly used in the process of retrieval.

The sentiment part of the query can be further detailed into subcategories such as POSITIVE OPINIONS, NEGATIVE OPINIONS, NEUTRAL OPINIONS each of which can be split according to sentiment intensity (HIGHLY POSITIVE OPINIONS, SLIGHTLY NEGATIVE OPINIONS etc.). But whatever level of categorisation we use, the query is still too abstract and cannot be used in practice. It therefore needs to be put into words and most probably expanded. The texts should also be indexed with appropriate sentiment tags which in the context of sentiment processing implies classification of the texts according to presence / absence of a sentiment and, if the texts are opinionated, according to their sentiment polarity.

To test the proposed approach we designed two experiments.

The purpose of the first experiment was to find the most effective kind of features for sentiment polar-

[1] In the context of this study terms "feature" and "basic unit" are used interchangeably.

ity discrimination (detection) which can be used for OIR [2]. Nie et al. (2000) found that for Chinese IR the most effective kinds of features were a combination of dictionary look up (longest-match algorithm) together with unigrams (single characters). The approach was tested in the first experiment.

The second experiment was designed to test the found set of features for text classification (indexing) for an OIR query of the first level (finds opinionated information) and for an OIR query of the second level (finds opinionated information with sentiment direction detection), thus the classifier should 1) detect opinionated texts and 2) classify the found items either as positive or as negative.

As training corpus for the second experiment we use the NTU sentiment dictionary (NTUSD) (by Ku et al. (2006a))[3] as well as a list of sentiment scores of Chinese characters obtained from processing of the same dictionary. Dictionary look up used the longest-match algorithm. The dictionary has 2809 items in the "positive" part and 8273 items in the "negative". The same dictionary was also used as a corpus for calculating the sentiment scores of Chinese characters. The use of the dictionary as a training corpus for obtaining the sentiment scores of characters is justified by two reasons: 1) it is domain-independent and 2) it contains only relevant (sentiment-related) information. The above mentioned parts of the dictionary used as the corpus comprised 24308 characters in the "negative" part and 7898 characters in the "positive" part.

4.1 Experiment 1

A corpus of E-Bay[4] customers' reviews of products and services was used as a test corpus. The total number of reviews is 128, of which 37 are negative (average length 64 characters) and 91 are positive (average length 18 characters), all of the reviews were tagged as 'positive' or 'negative' by the reviewers[5].

We computed two scores for each item (a review): one for positive sentiment, another for negative sentiment. The decision about an item's sentiment polarity was made every time by finding the biggest score of the two.

For every phrase (a chunk of characters between punctuation marks) a score was calculated as:

$$Sc_{phrase} = \sum (Sc_{dictionary}) + \sum (Sc_{character})$$

where $Sc_{dictionary}$ is a dictionary based score calculated using following formula:

$$Sc_{dictionary} = \frac{L_d}{L_s} * 100$$

where L_d - length of a dictionary item, L_s - length of a phrase. The constant value 100 is used to weight the score, obtained by a series of preliminary tests as a value that most significantly improved the accuracy.

The sentiment scores for characters were obtained by the formula:

$$Sc_i = F_i / F_{(i+j)}$$

where Sc_i is the sentiment score for a character for a given class *i*, F_i - the character's relative frequency in a class *i*, $F_{(i+j)}$ - the character's relative frequency in both classes *i* and *j* taken as one unit. The relative frequency of character *c* is calculated as

$$F_c = \frac{\sum N_c}{\sum N_{(1...n)}}$$

where $\sum N_c$ is a number of the character's occurrences in the corpus, and $\sum N_{(1...n)}$ is the number of all characters in the same corpus.

Preliminary tests showed that inverting all the characters for which $Sc_i \leq 1$ improves accuracy. The inverting is calculated as follows:

$$Sc_{inverted} = Sc_i - 1$$

We compute scores rather than probabilities since we are combining information from two distinct sources (characters and words).

[2] For simplicity we used only binary polarity in both experiments: positive or negative. Thus terms "sentiment polarity" and "sentiment direction" are used interchangeably in this paper.

[3] Ku et al. (2006a) automatically generated the dictionary by enlarging an initial manually created seed vocabulary by consulting two thesauri, including tong2yi4ci2ci2lin2 and the Academia Sinica Bilingual Ontological Wordnet 3.

[4] http://www.ebay.com.cn/

[5] The corpus is available at http://www.informatics.sussex.ac.uk/users/tz21/corpSmall.zip.

In addition to the features specified (characters and dictionary items) we also used a simple negation check. The system checked two most widely used negations in Chinese: *bu* and *mei*. Every phrase was compared with the following pattern: *negation+ 0-2 characters+ phrase*. The scores of all the unigrams in the phrase that matched the pattern were multiplied by -1.

Finally, the score was calculated for an item as the sum of the phrases' scores modified by the negation check:

$$Sc_{item} = \sum (Sc_{phrase} * NegCheck)$$

For sentiment polarity detection the item scores for each of the two polarities were compared to each other: the polarity with bigger score was assigned to the item.

$$SentimentPolarity = argmax(Sc_i|Sc_j)$$

where Sc_i is an item score for one polarity and Sc_j is an item score for the other.

The main evaluation measure was accuracy of sentiment identification, expressed in percent.

4.1.1 Results of Experiment 1

To find out which kinds of features perform best for sentiment polarity detection the system was run several times with different settings.

Running without character scores (with dictionary longest-match only) gave the following results: almost 64% of positive and near 65% for negative reviews were detected correctly, which is 64% accuracy for the whole corpus (note that a baseline classifier tagging all items as positive achieves an accuracy of 71.1%). Characters with sentiment scores alone performed much better on negative reviews (84% accuracy) rather than on positive (65%), but overall performance was still better: 70%. Both methods combined gave a significant increase on positive reviews (73%) and no improvement on negative (84%), giving 77% overall. The last run was with the dictionary look up, the characters and the negation check. The results were: 77% for positive and 89% for negative, 80% corpus-wide (see Table 1).

Judging from the results it is possible to suggest that both the word-based dictionary look up method and character-based method contributed to the final result. It also corresponds to the results obtained by Nie et al. (2000) for Chinese information retrieval, where the same combination of features (characters and words) also performed best.

Method	*Positive*	*Negative*	*All*
Dictionary	63.7	64.8	64.0
Characters	64.8	83.7	70.3
Characters+Dictionary	73.6	83.7	76.5
Char's+Dictionary+negation	76.9	89.1	80.4

Table 1: Results of Experiment 1 (accuracy in percent).

The negation check increased the performance by 3% overall, up to 80%. Although the performance gain is not very high, the computational cost of this feature is very low.

As we used a non-balanced corpus (71% of the reviews are positive), it is quite difficult to compare the results with the results obtained by other authors. But the proposed classifier outperformed some standart classifiers on the same data set: a Naive Bayes (multinomial) classifier gained only 49.6 % of accuracy (63 items tagged correctly) while a Support vector machine classifier got 64.5 % of accuracy (82 items).[6]

4.2 Experiment 2

The second experiment included two parts: determining whether texts are opinionated which is a precondition for the processing of the OPINION part of the query; and tagging found texts with relevant sentiment for processing a more detailed form of this query POSITIVE/NEGATIVE OPINION.

For this experiment we used the features that showed the best performance as described in section 4.1: the dictionary items and the characters with the sentiment scores.

The test corpus for this experiment consisted of 282 items, where every item is a paragraph. We used paragraphs as basic items in this experiment because of two reasons: 1. opinionated texts (reviews) are usually quite short (in our corpus all of them are one paragraph), while texts of other genres are usually much longer; and 2. for IR tasks it is more usual to retrieve units longer then a sentence.

[6]We used WEKA 3.4.10 (http://www.cs.waikato.ac.nz/ ml/weka)

The test corpus has following structure: 128 items are opinionated, of which 91 are positive and 37 are negative (all the items are the reviews used in the first experiment, see 4.1). 154 items are not opinionated, of which 97 are paragraphs taken from a scientific book on Chinese linguistics and 57 items are from articles taken form a Chinese on-line encyclopedia Baidu Baike[7].

For the first task we used the following technique: every item was assigned a score (a sum of the characters' scores and dictionary scores described in 4.1). The score was divided by the number of characters in the item to obtain the average score:

$$averSc_{item} = \frac{Sc_{item}}{L_{item}}$$

where Sc_{item} is the item score, and L_{item} is the length of an item (number of characters in it).

A positive and a negative average score is computed for each item.

4.2.1 Results of Experiment 2

To determine whether an item is opinionated (for OPINION query), the maximum of the two scores was compared to a threshold value. The best performance was achieved with the threshold value of 1.6 - more than 85% of accuracy[8] (see Table 2).

Next task (NEGATIVE/POSITIVE OPINIONS) was processed by comparing the negative and positive scores for each found item (see Table 2).

Query	*Recall*	*Precision*	*F-measure*
OPINION	71.8	85.1	77.9
POS/NEG OPINION	64.0	75.9	69.4

Table 2: Results of Experiment 2 (in percent).

Although the unopinionated texts are very different from the opinionated ones in terms of genre and topic, the standard classifiers (Naive Bayes (multinomial) and SVM) failed to identify any non-opinionated texts. The most probable explanation for this is that there were no items tagged 'unopinionated' in the training corpus (the sentiment dictionary) and there were only words and phrases with predominant sentiment meaning rather then topic-related.

[7] http://baike.baidu.com/

[8] A random choice could have approximately 55% of accuracy if tagged all items as negative.

It is worth noting that we observed the same relation between subjectivity detection and polarity classification accuracy as described by Pang and Lee (2004) and Eriksson (2006). The accuracy of the sentiment detection of opinionated texts (excluding erroneously detected unopinionated texts) in Experiment 2 has increased by 13% for positive reviews and by 6% for negative reviews (see Table 3).

Query	*Positive*	*Negative*
Experiment 1	76.9	89.1
Experiment 2	89.9	95.6

Table 3: Accuracy of sentiment polarity detection of opinionated texts (in percent).

5 Conclusion and Future Work

These preliminary experiments showed that using single characters and dictionary items modified by the negation check can produce reasonable results: about 78% F-measure for sentiment detection (see 4.1.1) and almost 70% F-measure for sentiment polarity identification (see 4.2.1) in the context of domain-independent opinionated information retrieval. However, since the test corpus is very small the results obtained need further validation on bigger corpora.

The use of the dictionary as a training corpus helped to avoid domain-dependency, however, using a dictionary as a training corpus makes it impossible to obtain grammar information by means of analysis of punctuation marks and grammar word frequencies.

More intensive use of context information could improve the accuracy. The dictionary-based processing may benefit from the use of word relations information: some words have sentiment information only when used with others. For example, a noun *dongxi* ('a thing') does not seem to have any sentiment information on its own, although it is tagged as 'negative' in the dictionary.

Some manual filtering of the dictionary may improve the output. It might also be promising to test the influence on performance of the different classes of words in the dictionary, for example, to use only adjectives or adjectives and nouns together (excluding adverbials).

Another technique to be tested is computing the

positive and negative scores for the characters used only in one class, but absent in another. In the current system, characters are assigned only one score (for the class they are present in). It might improve accuracy if such characters have an appropriate negative score for the other class.

Finally, the average sentiment score may be used for sentiment scaling. For example, if in our experiments items with a score less than 1.6 were considered not to be opinionated, then ones with score more than 1.6 can be put on a scale where higher scores are interpreted as evidence for higher sentiment intensity (the highest score was 52). The "scaling" approach could help to avoid the problem of assigning documents to more than one sentiment category as the approach uses a continuous scale rather than a predefined number of rigid classes. The scale (or the scores directly) may be used as a means of indexing for a search engine comprising OIR functionality.

References

Kushal Dave, Steve Lawrence, and David M. Pennock. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the International World Wide Web Conference*, pages 519 – 528, Budapest, Hungary. ACM Press.

Koji Eguchi and Victor Lavrenko. 2006. Sentiment retrieval using generative models. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 345–354, Sydney, July.

Brian Eriksson. 2006. Sentiment classification of movie reviews using linguistic parsing. http://www.cs.wisc.edu/∼apirak/cs/cs838/eriksson_final.pdf.

Soo-Min Kim and Eduard H. Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of COLING-04*, pages 1367–1373, Geneva, Switzerland, August 23-27.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006a. Opinion extraction, summarization and tracking in news and blog corpora. In *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*, volume AAAI Technical Report, pages 100–107, March.

Lun-Wei Ku, Yu-Ting Liang, and Hsin-Hsi Chen. 2006b. Tagging heterogeneous evaluation corpora for opinionated tasks. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 667–670, Genoa, Italy, May.

Wei Li. 2000. On Chinese parsing without using a separate word segmenter. *Communication of COLIPS*, 10:17–67.

Jian-Yun Nie, Jiangfeng Gao, Jian Zhang, and Ming Zhou. 2000. On the use of words and n-grams for Chinese information retrieval. In *Proceedings of the 5th International Workshop Information Retrieval with Asian Languages*, pages 141–148. ACM Press, November.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 271–278, Barcelona, Spain.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86, University of Pennsylvania.

Peter D. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 417–424, Philadelphia, Pennsylvania.

Limitations of Current Grammar Induction Algorithms

Bart Cramer
School of Behavioral and Cognitive Neurosciences
University of Groningen
Groningen, the Netherlands
bart.cramer@gmail.com

Abstract

I review a number of grammar induction algorithms (ABL, Emile, Adios), and test them on the Eindhoven corpus, resulting in disappointing results, compared to the usually tested corpora (ATIS, OVIS). Also, I show that using neither POS-tags induced from Biemann's unsupervised POS-tagging algorithm nor hand-corrected POS-tags as input improves this situation. Last, I argue for the development of entirely incremental grammar induction algorithms instead of the approaches of the systems discussed before.

1 Introduction

Grammar induction is a task within the field of natural language processing that attempts to construct a grammar of a given language solely on the basis of positive examples of this language. If a successful method is found, this will have both practical applications and considerable theoretical implications.

Concerning the practical side, this will make the engineering of NLP systems easier, especially for less widely studied languages. One can conceive successful GI algorithms as an inspiration for statistical machine translation systems.

Theoretically, grammar induction is important as well. One of the main assertions in the nativist's position is the Poverty of the Stimulus argument, which means that the child does not perceive enough positive examples of language throughout his early youth to have learned the grammar from his parents, without the help of innate knowledge (or: Universal Grammar), that severely constrains the number of hypotheses (i.e. grammars) that he can learn. Proved more strictly for formal grammars, Gold's (1967) work showed that one cannot learn any type of superfinite grammar (e.g. regular languages, context-free languages), if one only perceives (an unlimited amount of) positive examples. After, say, n examples, there is always more than 1 grammar that would be able to explain the seen examples, thus these grammar might give different judgments on an $n + 1^{th}$ example, of which it is impossible to say in advance which judgment is the correct one.

But, given this is true, isn't the grammar induction pursuit deemed to fail? Not really. First, there are hints that children do receive negative information, and that they use it for grammar acquisition. Also, the strictness required by Gold is not needed, and an approximation in the framework of PAC (Probably Approximately Correct) or VC (Vapnis and Chervonenkis) could then suffice. This, and other arguments favouring the use of machine learning techniques in linguistic theory testing, are very well reviewed in Lappin and Shieber (2007).

Several attempts have been made to create such systems. The authors of these systems reported promising results on the ATIS and OVIS treebanks. I tried to replicate these findings on the more complicated Eindhoven treebank, which turned out to yield disappointing results, even inferior to very simple baselines. As an attempt to ameliorate this, and as an attempt to confirm Klein and Manning's (2002) and Bod's (2006) thesis that good enough unsupervised POS-taggers exist to justify using POS-tags instead of words in evaluating GI systems, I pre-

Proceedings of the ACL 2007 Student Research Workshop, pages 43–48,
Prague, June 2007.

sented the algorithms with both POS-tags that were induced from Biemann's unsupervised POS-tagging algorithm and hand-corrected POS-tags. This did not lead to improvement.

2 Current Grammar Induction Models

2.1 Algorithms

Grammar induction models can be split up into two types: tag-based and word-based grammar induction. The key feature that distinguishes between these two is the type of input. Tag-based systems receive part-of-speech tags as their input (i.e. the words are already labelled), and only induce rules using the given tags. This kind of work is done by, for instance, Klein and Manning (2005). On the other hand, word-based models accept plain text as its input, and have to extract both the categories and the syntactic rules from given input.

Recently, several word-based grammar induction algorithms have been developed: Alignment-Based Learning (van Zaanen, 2002), Adios (Solan et al., 2005), Emile (Adriaans, 1992; Adriaans and Vervoort, 2002) and GraSp[1] (Henrichsen, 2002). Although the means of computation and underlying aims differ, they all rely to a certain extent on Harris' principle (1951): if two word groups constitute the same category, then they can be interchanged in any sentence, without damaging the grammaticality of that sentence. Hence, these GI system depend on the inverse: if two word groups appear to occur in the same contexts, they probably possess the same syntactic characteristics.

The most prominent example of this principle is Alignment-Based Learning, or ABL, (van Zaanen, 2002). This algorithm consists of two stages. First, all sentences are aligned such that it finds a shared and a distinct part of all pairs of sentences, suggesting that the distinct parts have the same type. For example, consider the pair 'I saw the man' and 'I saw John'. Here, 'John' and 'the man' are correctly identified as examples of the same type (NP's in this case). The second step, that takes the same corpus as input, tries to identify the constituents in that sentence. Because the generated constituents found in the previous step might overlap, the correct ones have to be selected. Simple heuristics are used to achieve this, for example to take the constituent that was generated first (ABL-first) or to take the constituent with the highest score on some probabilistic function (ABL-leaf). For details, I refer to van Zaanen (2000). Because ABL compares all sentences in the corpus with all other sentences, the algorithm is quadratic in the number of sentences, but has low memory demands. Interestingly, ABL does not come up with an explicit grammar, but generates just a bracketed version of the corpus instead.

	John (.)	Pat (.)	Jim (.)
walks	x	x	
talks		x	x
smiles	x	x	

Table 1: An example of some context/expression pairs to show the workings of EMILE. Note that, under standard settings, a rule covering this entire table will be inferred, causing a phrase like 'John talks' to be accepted, although there was no such input sentence.

Adios (Solan et al., 2005) uses Harris' principle as well, although it attempts to create a grammar (either context-free or context-sensitive) more explicitly. The algorithm represents language as a directed pseudograph[2], with *equivalence classes* (initially single words) as nodes. Input sentences can be regarded as 'snakes' over the nodes in the graph. If enough support is found, words are merged into equivalence classes, or frequently occurring edges are put in a *path* (a *rule* in usual grammatical terms). This generalisation process is done iteratively, until convergence is reached.

Emile (Adriaans, 1992; Adriaans and Vervoort, 2002) is the system that to a greater extent tries to pinpoint its reasons to accept a linguistic hypothesis. Each rule is divided into *expressions* and *types*, where types should be the interchangeable part of two sentences. Instead of explicitly comparing each sentence with all other sentences, it incrementally builds up a table of type/expression pairs, and on the basis of this table rules are extracted. An example is given in table 1. This incrementality has two major

[1] As there was no current working version of this system, I did not include it in this project.

[2] This is a graph that allows for loops and multiple edges.

consequences: it makes the system vastly more efficient in terms of time, at the cost of rising memory demands, and it models time linearly, in contrast to ABL and Adios.

2.2 Evaluation

Different methods of evaluation are used in GI. One of them is visual inspection (Henrichsen, 2002). This is not a reproducible and independent evaluation measure, and it does certainly not suffice as an assessment of the quality of the results. However, Roberts and Atwell (2003) argue that this evaluation should still be included in GI discussions.

A second evaluation method is shown by Solan et al. (2005), in which Adios had to carry out a test that is available on the Internet: English as a Second Language (ESL). This test shows three sentences, of which the examinee has to say which sentence is the grammatical one. Adios answers around 60% correct on these questions, which is considered as intermediate for a person who has had 6 years of English lessons. Although this sounds impressive, no examples of test sentences are given, and the website is not available anymore, so we are not able to assess this result.

A third option is to have sentences generated by the induced grammar judged on their naturalness, and compare this average with the average of the sentences of the original corpus. Solan et al. (2005) showed that the judgments of Adios generated sentences were comparable to the sentences in their corpus. However, the algorithm might just generates overly simple utterances, and will receive relatively high scores that it doesn't deserve.

The last option for evaluation is to compare the parses with hand-annotated treebanks. This gives the most quantifiable and detailed view on the performance of a GI system. An interesting comparative study between Emile and ABL using this evaluation method is available in van Zaanen and Adriaans (2001) where F-scores of 41.4% (Emile) and 61.7% (ABL) are reported on the OVIS (Openbaar Vervoer Informatie Systeem[3]; Dutch) corpus, and 25.4% and 39.2% on the ATIS (Air Traffic Information System; English) corpus.

[3]This acronym means Public Transport Information System.

3 Experiment 1

3.1 Motivation

A major choice in evaluating GI systems is to decide which corpus to train the algorithm on. The creators of ABL and Emile chose to test on the ATIS and OVIS corpus, which is, I believe, an unfortunate choice. These corpora contain sentences that are spoken to a computer, and represent a very limited subset of language. Deep recursion, one of the aspects that is hard to catch in grammar induction, does not occur often. The average sentence lengths are 7.5 (ATIS) and 4.4 (OVIS). If we want to know whether a system is truly capable of bootstrapping knowledge about language, there is only one way to test it: by using natural language that is unlimited in its expressive power. Therefore, I will test ABL, Adios and Emile on the Eindhoven corpus, that contains 7K sentences, with an average length of approximately 20 tokens. This is, as far as I know, the first attempt to train and test word-based GI algorithms on such a complicated corpus.

3.2 Method

The Eindhoven corpus has been automatically annotated by Alpino (Bouma et al., 2000; van der Beek et al., 2002), a wide-coverage hand-written parser for Dutch, with around 90% dependency triple accuracy. Afterwards, this treebank has been manually corrected. The treebank does not literally contain trees, but graphs: some nodes can be copied, so that linguistic structure can be analyzed in more detail. However, by removing all double nodes it is still possible to retrieve a list of bracket-tuples from these graphs. The graphs are also non-concatenative, meaning that a constituent can span word groups that are not contiguous. Therefore, if a sentence contains a constituent $w_i...w_jw_k...w_l$, with $k - j > 1$, three bracket-tuples are generated: (i, j), (k, l) and (i, l).

Evaluation of the algorithm is done according to PARSEVAL, except for a few changes that are also proposed by Klein and Manning (2002). The set of bracket-pairs that is found in the Alpino treebank are called *facts*, and those from a grammar induction algorithm *predictions*. The intersection of the facts and predictions are called *hits*. From these we can compute the unlabeled precision, recall and F-score. The subtleties adopted from Klein and Man-

ning are the following: constituents of length 0 or 1, constituents that span the whole sentence and constituents just excluding punctuation are not taken into account, as these are obvious predictions.

Three baselines were created: an algorithm that always branches left[4], idem for right-branching and an algorithm that performs binary branching on random points in the sentence. Note that left-branching and right-branching yield the maximum number of predictions.

3.3 Results

From the results in table 2, it can be seen that ABL scores best: it is the only one that is able to slightly outperform the random baseline. This is surprising, because it is the least complicated system of the three. Adios and Emile performed poorly. It appears that, with larger sentences, the search space become too sparse to actually induce any meaningful structure. This is expressed in the low number of predictions per sentence that Adios (1.5) and Emile (0.7) make. Adjusting support parameters, to make the algorithm accept more hypotheses, did not have the intended effect. Still, notice that Emile has a relatively high precision.

In sum, none of the systems is convincingly able to outperform the very simple baselines. Neither did visual inspection give the impression that meaningful information was derived. Therefore, it can be concluded that current word-based GI algorithms are not equipped to derive syntactic structure from corpora as complicated as the Eindhoven corpus.

4 Experiment 2

4.1 Motivation

The second experiment deals with the difference between tag-based and word-based systems. Intuitively, the latter task seems to be more challenging. Still, Klein and Manning (2002) and Bod (2006) stick to tag-based models. Their argumentation is twofold.

First, Bod assumes that unsupervised POS-tagging can be done successfully, without explicitly showing results that can confirm this. Klein and Manning did tag their text using a simple unsupervised POS-tagging algorithm, and this moderately harmed their performance: their Context-Constituent Model's F-score on Wall Street Journal text fell from 71.1% to 63.2%.

Second, Klein and Manning created context vectors for a number of non-terminals (NP, VP, PP), and extracted the two principal components from these vectors. They did the same with contexts of constituents and distituents. The distribution of these vectors suggest that the non-terminals were easier to distinguish from each other than the constituents from the distituents, suggesting that POS-tagging is easier than finding syntactic rules. However, this result would be more convincing if this is true for POS-tags as well.

4.2 Method

In order to test the argument above, and as an attempt to improve the results from the previous experiment, POS-tags were induced using Biemann's unsupervised POS-tagger (Biemann, 2006). Because that algorithm needs at least 50M words to work reliably, it was trained on the concatenation of the Eindhoven corpus and the CLEF corpus (70M words, also newspaper text). The tags of the Eindhoven corpus are then used as input for the GI algorithms, both under same settings as experiment 1. The evaluation was done the same way as in experiment 1.

The same method was carried out using hand-corrected tags. Large and equal improvements will imply the justification for tag-based grammar induction. If the models only improve on the hand-corrected tags, this will suggest the opposite.

4.3 Results

The results can be found in table 3. Generally, more predictions were made with respect to experiment 1, due to the denser search space. Only a convergence to the baseline was achieved, especially by Adios and Emile, that were very low in predictions in the first experiment. Again, none of the tested systems was able to clearly outperform the baselines.

Because using neither induced nor hand-corrected made the systems work more reliably, there seems to be no strong evidence in favor or against Bod's and Klein and Manning's conjecture. Therefore, there is no sound justification for tag-based grammar induction yet.

[4]For example: [[[I saw] the] large] house.

Method	Hits/Predictions	Precision	Recall	F-score
Left	5.8K / 119K	4.9%	9.2%	6.4%
Right	4.4K / 119K	3.6%	6.9%	4.8%
Random	11K / 93K	11.7%	17.3%	14.0%
ABL-leaf	4.0K / 24K	16.9%	6.4%	9.3%
ABL-first	13K / 113K	11.6%	20.8%	14.9%
Adios	319 / 11K	2.8%	0.5%	0.9%
Emile	912 / 5.2K	17.3%	1.5%	2.7%

Table 2: This table shows the results of experiment 1. Left, Right and Random are baseline scores. The two variants of ABL differ in the selection phase. 62.9K facts were found in the Alpino treebank.

	Induced tags				Hand-corrected tags			
Method	Hits/Pred.'s	Precision	Recall	F-score	Hits/Pred.'s	Precision	Recall	F-score
ABL-leaf	5K / 30K	16.8%	8.1%	10.9%	7.0K / 34K	21.0%	11.2%	14.6%
ABL-first	11K / 125K	9.2%	18.2%	12.2%	12.6K / 123K	10.3%	20.0%	13.6%
Adios	2.7K / 24K	11.2%	4.3%	6.3%	2.2K / 20K	11.0%	3.5%	5.3%
Emile	1.8K / 16K	11.2%	2.9%	4.6%	1.7K / 19K	8.9%	2.7%	4.1%

Table 3: This table shows the results of experiment 2. The baseline scores are identical to the ones in experiment 1.

5 Discussion

The results from experiment 1 and 2 clearly show that ABL, Adios and Emile have severe shortcomings, and that they cannot derive meaningful structure from language as complicated as the Eindhoven corpus. An important reason for this is that a corpus with only 7K sentences is not able to sufficiently cover the search space. This can be seen from the very low number of predictions made by Adios and Emile: there was not enough support to accept hypotheses.

But how should we proceed? Any algorithm based solely on Harris' principle can be either incremental (Emile) or non-incremental (ABL, Adios). The previous experiments show that very large corpora are needed to mitigate the very sparse search space, leading me to conclude that non-incremental systems are not suitable for the problem of grammar induction. Also, incremental systems have the advantage of an intuitive notion of time: it is always clear which working hypothesis of a grammar is maintained.

Emile retains a Boolean table with all combinations of types and expressions it has encountered up until a given moment. This means that very infrequent words demand a disproportionally large part of the memory. Therefore, all found words and rules should be divided into three groups: pivotal, normal and infrequent. Initially, all encountered words are infrequent. Transitions to the normal and pivotal stage occur when an estimator of the relative frequency is high enough, for example by taking the lower bound of the confidence interval (Mikheev, 1997). Ultimately, the number of words in the normal and pivotal stage will converge to a constant. For example, if the relative frequency of a word should be larger than 0.01 to become pivotal, there can only be 100 of these words. Because one can define upper limits for pivotal and normal words, the size of the bookkeeping table is limited as well. Also, when the system starts inducing syntactic categories of words, very infrequent words should not be parsed as a separate category initially, but as a member of another open-class category. This connects to the cross-linguistic tendency that infrequent words generally have simple complementation patterns.

One very important question remains: what intuitions should this imaginary system use to induce rules? First, all sentences should be sorted by length. Then, for each sentence, the following steps are taken:

- Update the bookkeeping tables.
- Parse the sentence as deeply as possible.
- If the sentence cannot be parsed completely, induce all possible rules that would make the parse complete. Add all these rules to the bookkeeping tables.

The last step deserves some extra attention. If the algorithm encounters the sentence 'he is such a (.)', we can safely infer that the unknown word at (.) is a noun. Inducing complementation patterns should be possible as well. Imagine that the algorithm understands NP's and transitive verbs. Then consider the following: 'John gave Tim a book'. It will parse 'John gave Tim' as a sentence, and 'a book' as a noun phrase. Because these two should be connected, a number of hypotheses are generated, for example: 'a book' is a complement of 'Tim'; 'a book' is a complement of 'John gave Tim'; 'a book' is a second complement of 'gave'. Naturally, only the last hypothesis is correct. All three inductions are included, but only the last is likely to be reproduced in later sentences in the corpus, because sentences of the form '(.) gave (.) (.)' are more likely than 'John gave Tim (.)' and 'Tim (.)'.

6 Acknowledgments

I would like to thank Jennifer Spenader, Gertjan van Noord and the anonymous reviewers for providing me their invaluable comments.

References

Pieter W. Adriaans and Mark R. Vervoort. 2002. The EMILE 4.1 grammar induction toolbox. In *Proceedings of the 6th International Colloquium on Grammar Induction (ICGI)*, pages 293–295, Amsterdam, the Netherlands.

Pieter W. Adriaans. 1992. *Language learning from a categorial perspective*. Ph.D. thesis, University of Amsterdam, NL.

Chris Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of ACL/COLING-2006 Students Research Workshop*, pages 7–12, Sydney, Australia.

Rens Bod. 2006. An all-subtrees approach to unsupervised parsing. In *Proceedings of ACL/COLING-2006*, pages 865–872, Sydney, Australia.

Gosse Bouma, Gertjan van Noord, and Robert Malouf. 2000. Alpino: wide-coverage computational analysis of Dutch. In *Proceedings of Computational Linguistics in the Netherlands (CLIN)*, pages 45–59, Tilburg, the Netherlands.

E. Mark Gold. 1967. Language identification in the limit. *Information and Control*, 16:447–474.

Zellig S. Harris. 1951. *Methods in Structural Linguistics*. University of Chicago Press, Chicago.

Peter J. Henrichsen. 2002. GraSp: Grammar learning from unlabelled speech corpora. In *Proceedings of CoNLL-2002*, pages 22–28, Pennsylvania, PA, USA.

Dan Klein and Christopher D. Manning. 2002. A generative Constituent-Context Model for improved grammar induction. In *Proceedings of ACL-2001*, pages 128–135, Toulouse, France.

Dan Klein and Christopher D. Manning. 2005. Natural language grammar induction with a generative constituent-context model. *Pattern Recognition*, 9(38):1407–1419.

Shalom Lappin and Stuart M. Shieber. 2007. Machine learning theory and practice as a source of insight into universal grammar. *Computational Linguistics*, 43:1–34.

Andrei Mikheev. 1997. Automatic rule induction for unknown-word guessing. *Computational Linguistics*, 23(3):405–423.

Andrew Roberts and Eric Atwell. 2003. The use of corpora for automatic evaluation of grammar inference systems. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 657–661, Lancaster, United Kingdom.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences*, 102(33):11629–11634.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino dependency treebank. In *Proceedings of Computational Linguistics in the Netherlands (CLIN) 2001*, pages 8–22, Enschede, the Netherlands.

Menno van Zaanen and Pieter W. Adriaans. 2001. Alignment-Based Learning versus EMILE: A comparison. In *Proceedings of the 13th Dutch-Belgian Artificial Intelligence Conference (BNAIC)*, pages 315–322, Amsterdam, the Netherlands.

Menno van Zaanen. 2002. Implementing Alignment-Based Learning. In *Proceedings of the 6th International Colloquium on Grammatical Inference (ICGI)*, pages 312–314, Amsterdam, the Netherlands.

Logistic Online Learning Methods and Their Application to Incremental Dependency Parsing

Richard Johansson
Department of Computer Science
Lund University
Lund, Sweden
richard@cs.lth.se

Abstract

We investigate a family of update methods for online machine learning algorithms for cost-sensitive multiclass and structured classification problems. The update rules are based on multinomial logistic models. The most interesting question for such an approach is how to integrate the cost function into the learning paradigm. We propose a number of solutions to this problem.

To demonstrate the applicability of the algorithms, we evaluated them on a number of classification tasks related to incremental dependency parsing. These tasks were conventional multiclass classification, hiearchical classification, and a structured classification task: complete labeled dependency tree prediction. The performance figures of the logistic algorithms range from slightly lower to slightly higher than margin-based online algorithms.

1 Introduction

Natural language consists of complex structures, such as sequences of phonemes, parse trees, and discourse or temporal graphs. Researchers in NLP have started to realize that this complexity should be reflected in their statistical models. This intuition has spurred a growing interest of related research in the machine learning community, which in turn has led to improved results in a wide range of applications in NLP, including sequence labeling (Lafferty et al., 2001; Taskar et al., 2006), constituent and dependency parsing (Collins and Duffy, 2002; McDonald et al., 2005), and logical form extraction (Zettlemoyer and Collins, 2005).

Machine learning research for structured problems have generally used margin-based formulations. These include global batch methods such as Max-margin Markov Networks (M^3N) (Taskar et al., 2006) and SVM^{struct} (Tsochantaridis et al., 2005) as well as online methods such as Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003) and the Online Passive-Aggressive Algorithm (OPA) (Crammer et al., 2006). Although the batch methods are formulated very elegantly, they do not seem to scale well to the large training sets prevalent in NLP contexts. The online methods on the other hand, although less theoretically appealing, can handle realistically sized data sets.

In this work, we investigate whether logistic online learning performs as well as margin-based methods. Logistic models are easily extended to using kernels; that this is theoretically well-justified was shown by Zhu and Hastie (2005), who also made an elegant argument that margin-based methods are in fact related to regularized logistic models. For batch learning, there exist several learning algorithms in a logistic framework for conventional multiclass classification but few for structured problems.

Prediction of complex structures is conventionally treated as a cost-sensitive multiclass classification problem, although special care has to be taken to handle the large space of possible outputs. The integration of the cost function into the logistic framework leads to two distinct (although related) update methods: the Scaled Prior Variance (SPV) and the Minimum Expected Cost (MEC) updates.

Apart from its use in structured prediction, cost-sensitive classification is useful for *hierachical* classification, which we briefly consider here in an experiment. This type of classification has useful ap-

Proceedings of the ACL 2007 Student Research Workshop, pages 49–54,
Prague, June 2007.

plications in NLP. Apart from the obvious use in classification of concepts in an ontology, it is also useful for prediction of complex morphological or named-entity tags. Cost-sensitive learning is also required in the SEARN algorithm (Daumé III et al., 2006), which is a method to decompose the prediction problem of a complex structure into a sequence of actions, and train the search in the space of action sequences to maximize global performance.

2 Algorithm

We model the learning problem as finding a discriminant function F that assigns a score to each possible output y given an input $\boldsymbol{x}$. Classification in this setting is done by finding the $\hat{y}$ that maximizes $F(\boldsymbol{x}, y)$. In this work, we consider linear discriminants of the following form:

$$F(\boldsymbol{x}, y) = \langle \boldsymbol{w}, \boldsymbol{\Psi}(\boldsymbol{x}, y) \rangle$$

Here, $\boldsymbol{\Psi}(\boldsymbol{x}, y)$ is a numeric feature representation of the pair $(\boldsymbol{x}, y)$ and $\boldsymbol{w}$ a vector of feature weights. Learning in this case is equivalent to assigning appropriate weights in the vector $\boldsymbol{w}$.

In the online learning framework, the weight vector is constructed incrementally. Algorithm 1 shows the general form of the algorithm. It proceeds a number of times through the training set. In each step, it computes an update to the weight vector based on the current example. The resulting weight vector tends to be overfit to the last few examples; one way to reduce overfitting is to use the average of all successive weight vectors as the result of the training (Freund and Schapire, 1999).

Algorithm 1 General form of online algorithms

input Training set $\mathcal{T} = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^{T}$
Number of iterations N
for n in 1..N
  for $(\boldsymbol{x}_t, y_t)$ in $\mathcal{T}$
    Compute update vector $\delta\boldsymbol{w}$ for $(\boldsymbol{x}_t, y_t)$
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \delta\boldsymbol{w}$
return $\boldsymbol{w}_{\text{average}}$

Following earlier online learning methods such as the Perceptron, we assume that in each update step, we adjust the weight vector by incrementally adding feature vectors. For stability, we impose the constraint that the sum of the updates in each step should be zero. We assume that the possible output values are $\{y_i\}_{i=0}^{m}$ and, for convenience, that y_0 is the correct value. This leads to the following *ansatz*:

$$\delta\boldsymbol{w} = \sum_{j=1}^{m} \alpha_j (\boldsymbol{\Psi}(\boldsymbol{x}, y_0) - \boldsymbol{\Psi}(\boldsymbol{x}, y_j))$$

Here, α_j defines how much F is shifted to favor y_0 instead of y_j. This is also the approach (implicitly) used by other algorithms such as MIRA and OPA.

The following two subsections present two ways of creating the weight update $\delta\boldsymbol{w}$, differing in how the cost function is integrated into the model. Both are based on a multinomial logistic framework, where we model the probability of the class y being assigned to an input $\boldsymbol{x}$ using a "soft-max" function as follows:

$$P(y|\boldsymbol{x}) = \frac{e^{F(\boldsymbol{x},y)}}{\sum_{j=0}^{m} e^{F(\boldsymbol{x},y_j)}}$$

2.1 Scaled Prior Variance Approach

The first update method, Scaled Prior Variance (SPV), directly uses the probability of the correct output. It uses a maximum a posteriori approach, where the cost function is used by the prior.

Naïvely, the update could be done by maximizing the likelihood with respect to $\boldsymbol{\alpha}$ in each step. However, this would lead to overfitting – in the case of separability, a maximum does not even exist. We thus introduce a regularizing prior that penalizes large values of $\boldsymbol{\alpha}$. We introduce variance-controlling hyperparameters s_j for each α_j, and with a Gaussian prior we obtain (disregarding constants) the following log posterior:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\alpha}) &= \sum_{j=1}^{m} \alpha_j (K_{00} - K_{j0}) - \sum_{j=1}^{m} s_j \alpha_j^2 \\ &- \log \sum_{k=0}^{m} e^{f_k + \sum_{j=1}^{m} \alpha_j (K_{0k} - K_{jk})} \end{aligned}$$

where $K_{ij} = \langle \boldsymbol{\Psi}(\boldsymbol{x}, y_i), \boldsymbol{\Psi}(\boldsymbol{x}, y_j) \rangle$ and $f_k = F(\boldsymbol{x}, y_k)$ (i.e. the output before $\boldsymbol{w}$ is updated). As usual, the feature vectors occur only in inner products, allowing us to use kernels if appropriate.

We could have used any prior; however, in practice we will require it to be log-concave to avoid suboptimal local maxima. A Laplacian prior (i.e. $-\sum_{j=1}^{m} s_j|\alpha_j|$) will also be considered in this work – the discontinuity of its gradient at the origin seems to pose no problem in practice.

Costs are incorporated into the model by associating them to the prior variances. We tried two variants of variance scaling. In the first case, we let the variance be directly proportional to the cost (C-SPV):

$$s_j = \frac{\gamma}{c(y_j)}$$

where γ is a tradeoff parameter controlling the relative weight of the prior with respect to the likelihood. Intuitively, this model allows the algorithm more freedom to adjust an α_j associated with a y_j with a high cost.

In the second case, inspired by margin-based learning we instead scaled the variance by the *loss*, i.e. the scoring error plus the cost (L-SPV):

$$s_j = \frac{\gamma}{\max(0, f_j - f_0) + c(y_j)}$$

Here, the intuition is instead that the algorithm is allowed more freedom for "dangerous" outputs that are ranked high but have high costs.

2.2 Minimum Expected Cost Approach

In the second approach to integrating the cost function, the Minimum Expected Cost (MEC) update, the method seeks to minimize the expected cost in each step. Once again using the soft-max probability, we get the following expectation of the cost:

$$\begin{aligned}\mathbf{E}(c(y)|\boldsymbol{x}) &= \sum_{k=0}^{m} c(y_k) P(y_k|\boldsymbol{x}) \\ &= \frac{\sum_{k=0}^{m} c(y_k) e^{f_k + \sum_{j=1}^{m} \alpha_j (K_{0k} - K_{jk})}}{\sum_{k=0}^{m} e^{f_k + \sum_{j=1}^{m} \alpha_j (K_{0k} - K_{jk})}}\end{aligned}$$

This quantity is easily minimized in the same way as the SPV posterior was maximized, although we had to add a constant 1 to the expectation to avoid numerical instability. To avoid overfitting, we added a quadratic regularizer $\gamma \sum_{j=1}^{m} \alpha_j^2$ to $\log(1 + \mathbf{E}(c(y)|\boldsymbol{x}))$ just like the prior in the SPV method, although this regularizer does not have an interpretation as a prior.

The MEC update is closely related to SPV: for cost-insensitive classification (i.e. the cost of every misclassified instance is 1), the expectation is equal to one minus the likelihood in the SPV model.

2.3 Handling Complex Prediction Problems

The algorithm can thus be used for any cost-sensitive classification problem. This class of problems includes prediction of complex structures such as trees or graphs. However, for those problems the set of possible outputs is typically very large. Two broad categories of solutions to this problem have been common in literature, both of which rely on the structure of the domain:

- *Subset selection*: instead of working with the complete range of outputs, only an "interesting" subset is used, for instance by repeatedly finding the most violated constraints (Tsochantaridis et al., 2005) or by using N-best search (McDonald et al., 2005).
- *Decomposition*: the inherent structure of the problem is used to factorize the optimization problem. Examples include Markov decompositions in M^{3}N (Taskar et al., 2006) and dependency-based factorization for MIRA (McDonald et al., 2005).

In principle, both methods could be used in our framework. In this work, we use subset selection since it is easy to implement for many domains (in the form of an N-best search) and allows a looser coupling between the domain and the learning algorithm.

2.4 Implementation Issues

Since we typically work with only a few variables in each iteration, maximizing the log posterior or minimizing the expectation is easy (assuming, of course, that we chose a log-concave prior). We used gradient ascent and did not try to use more sophisticated optimization procedures like BFGS or Newton's method. Typically, only a few iterations were needed to reach the optimum. The running time of the update step is almost identical to that of MIRA, which solves a small quadratic program in each step, but longer than for the Perceptron algorithm or OPA.

Actions	Parser actions	Conditions
Initialize	$(nil, W, \emptyset)$	
Terminate	(S, nil, A)	
Left-arc	$(n\|S, n'\|I, A) \rightarrow (S, n'\|I, A \cup \{(n', n)\})$	$\neg\exists n''(n'', n) \in A$
Right-arc	$(n\|S, n'\|I, A) \rightarrow (n'\|n\|S, I, A \cup \{(n, n')\})$	$\neg\exists n''(n'', n') \in A$
Reduce	$(n\|S, I, A) \rightarrow (S, I, A)$	$\exists n'(n', n) \in A$
Shift	$(S, n\|I, A) \rightarrow (n\|S, I, A)$	

Table 1: Nivre's parser transitions where W is the initial word list; I, the current input word list; A, the graph of dependencies; and S, the stack. (n', n) denotes a dependency relations between n' and n, where n' is the head and n the dependent.

3 Experiments

To compare the logistic online algorithms against other learning algorithms, we performed a set of experiments in incremental dependency parsing using the Nivre algorithm (Nivre, 2003).

The algorithm is a variant of the shift–reduce algorithm and creates a projective and acyclic graph. As with the regular shift–reduce, it uses a stack S and a list of input words W, and builds the parse tree incrementally using a set of parsing actions (see Table 1). However, instead of finding constituents, it builds a set of arcs representing the graph of dependencies. It can be shown that every projective dependency graph can be produced by a sequence of parser actions, and that the worst-case number of actions is linear with respect to the number of words in the sentence.

3.1 Multiclass Classification

In the first experiment, we trained multiclass classifiers to choose an action in a given parser state (see (Nivre, 2003) for a description of the feature set). We stress that this is true multiclass classification rather than a decomposed method (such as one-versus-all or pairwise binarization).

As a training set, we randomly selected 50,000 instances of state–action pairs generated for a dependency-converted version of Penn Treebank. This training set contained 22 types of actions (such as SHIFT, REDUCE, LEFT-ARC(SUBJECT), and RIGHT-ARC(OBJECT). The test set was also randomly selected and contained 10,000 instances.

We trained classifiers using the logistic updates (C-SPV, L-SPV, and MEC) with Gaussian and Laplacian priors. Additionally, we trained OPA and MIRA classifiers, as well as an Additive Ultraconservative (AU) classifier (Crammer and Singer, 2003), a variant of the Perceptron.

For all algorithms, we tried to find the best values of the respective regularization parameter using cross-validation. All training algorithms iterated five times through the training set and used an expanded quadratic kernel.

Table 2 shows the classification error for all algorithms. As can be seen, the performance was lower for the logistic algorithms, although the difference was slight. Both the logistic (MEC and SPV) and the margin-based classifiers (OPA and MIRA) outperformed the AU classifier.

Method	Test error
MIRA	6.05%
OPA	6.17%
C-SPV, Laplace	6.20%
MEC, Laplace	6.21%
C-SPV, Gauss	6.22%
MEC, Gauss	6.23%
L-SPV, Laplace	6.25%
L-SPV, Gauss	6.26%
AU	6.39%

Table 2: Multiclass classification results.

3.2 Hierarchical Classification

In the second experiment, we used the same training and test set, but considered the selection of the parsing action as a hierarchical classficiation task, i.e. the predicted value has a main type (SHIFT, REDUCE, LEFT-ARC, and RIGHT-ARC) and possibly also a subtype (such as LEFT-ARC(SUBJECT) or

RIGHT-ARC(OBJECT)).

To predict the class in this experiment, we used the same feature function but a new cost function: the cost of misclassification was 1 for an incorrect parsing action, and 0.5 if the action was correct but the arc label incorrect.

We used the same experimental setup as in the multiclass experiment. Table 3 shows the average cost on the test set for all algorithms. Here, the MEC update outperformed the margin-based ones by a negligible difference. We did not use AU in this experiment since it does not optimize for cost.

Method	Average cost
MEC, Gauss	0.0573
MEC, Laplace	0.0576
OPA	0.0577
C-SPV, Gauss	0.0582
C-SPV, Laplace	0.0587
MIRA	0.0590
L-SPV, Gauss	0.0590
L-SPV, Laplace	0.0632

Table 3: Hierarchical classification results.

3.3 Prediction of Complex Structures

Finally, we made an experiment in prediction of dependency trees. We created a global model where the discriminant function was trained to assign high scores to the correct parse tree. A similar model was previously used by McDonald et al. (2005), with the difference that we here represent the parse tree as a sequence of actions in the incremental algorithm rather than using the dependency links directly.

For a sentence $\boldsymbol{x}$ and a parse tree y, we defined the feature representation by finding the sequence $((S_1, I_1), a_1), ((S_2, I_2), a_2) \ldots$ of states and their corresponding actions, and creating a feature vector for each state/action pair. The discriminant function was thus written

$$\langle \boldsymbol{\Psi}(\boldsymbol{x}, y), \boldsymbol{w} \rangle = \sum_i \langle \psi((S_i, I_i), a_i), \boldsymbol{w} \rangle$$

where ψ is the feature function from the previous two experiments, which assigns a feature vector to a state (S_i, I_i) and the action a_i taken in that state.

The cost function was defined as the sum of link costs, where the link cost was 0 for a correct dependency link with a correct label, 0.5 for a correct link with an incorrect label, and 1 for an incorrect link.

Since the history-based feature set used in the parsing algorithm makes it impossible to use independence to factorize the scoring function, an exact search to find the best-scoring action sequence is not possible. We used a beam search of width 2 in this experiment.

We trained models on a 5000-word subset of the Basque Treebank (Aduriz et al., 2003) and evaluated them on a 8000-word subset of the same corpus. As before, we used an expanded quadratic kernel, and all algorithms iterated five times through the training set.

Table 4 shows the results of this experiment. We show labeled accuracy instead of cost for ease of interpretation. Here, the loss-based SPV outperformed

Method	Labeled Accuracy
L-SPV, Gauss	66.24
MIRA	66.19
MEC, Gauss	65.99
C-SPV, Gauss	65.84
OPA	65.45
MEC, Laplace	64.81
C-SPV, Laplace	64.73
L-SPV, Laplace	64.50

Table 4: Results for dependency tree prediction.

MIRA, and two other logistic updates also outperformed OPA. The differences between the first four scores are however not statistically significant. Interestingly, all updates with Laplacian prior resulted in low performance. The reason for this may be that Laplacian priors tend to promote sparse solutions (see Krishnapuram et al. (2005), *inter alia*), and that this sparsity is detrimental for this highly lexicalized feature set.

4 Conclusion and Future Work

This paper presented new update methods for online machine learning algorithms. The update methods are based on a multinomial logistic model. Their performance is on par with other state-of-the-art online learning algorithms for cost-sensitive problems.

We investigated two main approaches to integrating the cost function into the logistic model. In the first method, the cost was linked to the prior variances, while in the second method, the update rule sets the weights to minimize the expected cost. We tried a few different priors. Which update method and which prior was the best varied between experiments. For instance, the update where the prior variances were scaled by the costs was the best-performing in the multiclass experiment but the worst-performing in the dependency tree prediction experiment.

In the SPV update, the cost was incorporated into the MAP model in a rather ad-hoc fashion. Although this seems to work well, we would like to investigate this further and possibly devise a cost-based prior that is both theoretically well-grounded and performs well in practice.

To achieve a good classification performance using the updates presented in this article, there is a considerable need for cross-validation to find the best value for the regularization parameter. This is true for most other classification methods as well, including SVM, MIRA, and OPA. There has been some work on machine learning methods where this parameter is tuned automatically (Tipping, 2001), and a possible extension to our work could be to adapt those models to the multinomial and cost-sensitive setting.

We applied the learning models to three problems in incremental dependency parsing, the last of which being prediction of full labeled dependency trees. Our system can be seen as a unification of the two best-performing parsers presented at the CoNLL-X Shared Task (Buchholz and Marsi, 2006).

References

Itzair Aduriz, Maria Jesus Aranzabe, Jose Mari Arriola, Aitziber Atutxa, Arantza Diaz de Ilarraza, Aitzpea Garmendia, and Maite Oronoz. 2003. Construction of a Basque dependency treebank. In *Proceedings of the TLT*, pages 201–204.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the CoNLL-X*.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the ACL*.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 2003(3):951–991.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Schwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 2006(7):551–585.

Hal Daumé III, John Langford, and Daniel Marcu. 2006. Search-based structured prediction. *Submitted*.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Balaji Krishnapuram, Lawrence Carin, Mário A. T. Figueiredo, and Alexander J. Hartemink. 2005. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(6).

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP-2005*.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160, Nancy, France, 23-25 April.

Ben Taskar, Carlos Guestrin, Vassil Chatalbashev, and Daphne Koller. 2006. Max-margin Markov networks. *Journal of Machine Learning Research*, to appear.

Michael E. Tipping. 2001. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211 – 244.

Iannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6(Sep):1453–1484.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI 2005*.

Ji Zhu and Trevor Hastie. 2005. Kernel logistic regression and the import vector machine. *Journal of Computational and Graphical Statistics*, 14(1):185–205.

Adaptive String Distance Measures for Bilingual Dialect Lexicon Induction

Yves Scherrer
Language Technology Laboratory (LATL)
University of Geneva
1211 Geneva 4, Switzerland
yves.scherrer@lettres.unige.ch

Abstract

This paper compares different measures of graphemic similarity applied to the task of bilingual lexicon induction between a Swiss German dialect and Standard German. The measures have been adapted to this particular language pair by training stochastic transducers with the Expectation-Maximisation algorithm or by using handmade transduction rules. These adaptive metrics show up to 11% F-measure improvement over a static metric like Levenshtein distance.

1 Introduction

Building lexical resources is a very important step in the development of any natural language processing system. However, it is a time-consuming and repetitive task, which makes research on automatic induction of lexicons particularly appealing. In this paper, we will discuss different ways of finding lexical mappings for a translation lexicon between a Swiss German dialect and Standard German. The choice of this language pair has important consequences on the methodology. On the one hand, given the sociolinguistic conditions of dialect use (diglossia), it is difficult to find written data of high quality; parallel corpora are virtually non-existent. These data constraints place our work in the context of scarce-resource language processing. On the other hand, as the two languages are closely related, the lexical relations to be induced are less complex. We argue that this point alleviates the restrictions imposed by the scarcity of the resources. In particular, we claim that if two languages are close, even if one of them is scarcely documented, we can successfully use techniques that require training.

Finding lexical mappings amounts to finding word pairs that are maximally similar, with respect to a particular definition of similarity. Similarity measures can be based on any level of linguistic analysis: semantic similarity relies on context vectors (Rapp, 1999), while syntactic similarity is based on the alignment of parallel corpora (Brown et al., 1993). Our work is based on the assumption that phonetic (or rather graphemic, as we use written data) similarity measures are the most appropriate in the given language context because they require less sophisticated training data than semantic or syntactic similarity models. However, phonetic similarity measures can only be used for *cognate language pairs*, i.e. language pairs that can be traced back to a common historical origin and that possess highly similar linguistic (in particular, phonological and morphological) characteristics. Moreover, we can only expect phonetic similarity measures to induce *cognate word pairs*, i.e. word pairs whose forms and significations are similar, as a result of a historical relationship.

We will present different models of phonetic similarity that are adapted to the given language pair. In particular, attention has been paid to develop techniques requiring little manually annotated data.

2 Related Work

Our work is inspired by Mann and Yarowsky (2001). They induce translation lexicons between a resource-rich language (typically English) and a scarce resource language of another language family (for example, Portuguese) by using a resource-

Proceedings of the ACL 2007 Student Research Workshop, pages 55–60,
Prague, June 2007.

rich bridge language of the same family (for example, Spanish). While they rely on existing translation lexicons for the source-to-bridge step (English-Spanish), they use string distance models (called *cognate models*) for the bridge-to-target step (Spanish-Portuguese). Mann and Yarowsky (2001) distinguish between *static metrics*, which are sufficiently general to be applied to any language pair, and *adaptive metrics*, which are adapted to a specific language pair. The latter allow for much finer-grained results, but require more work for the adaptation. Mann and Yarowsky (2001) use variants of Levenshtein distance as a static metric, and a Hidden Markov Model (HMM) and a stochastic transducer trained with the Expectation-Maximisation (EM) algorithm as adaptive metrics. We will also use Levenshtein distance as well as the stochastic transducer, but not the HMM, which performed worst in Mann and Yarowsky's study.

The originality of their approach is that they apply models used for speech processing to cognate word pair induction. In particular, they refer to a previous study by Ristad and Yianilos (1998). Ristad and Yianilos showed how a stochastic transducer can be trained in a non-supervised manner using the EM algorithm and successfully applied their model to the problem of pronunciation recognition (sound-to-letter conversion). Jansche (2003) reviews their work in some detail, correcting thereby some errors in the presentation of the algorithms.

Heeringa et al. (2006) present several modifications of the Levenshtein distance that approximate linguistic intuitions better. These models are presented in the framework of dialectometry, i.e. they provide numerical measures for the classification of dialects. However, some of their models can be adapted to be used in a lexicon induction task. Kondrak and Sherif (2006) use phonetic similarity models for cognate word identification.

Other studies deal with lexicon induction for cognate language pairs and for scarce resource languages. Rapp (1999) extends an existing bilingual lexicon with the help of non-parallel corpora, assuming that corresponding words share co-occurrence patterns. His method has been used by Hwa et al. (2006) to induce a dictionary between Modern Standard Arabic and the Levantine Arabic dialect. Although this work involves two closely related language varieties, graphemic similarity measures are not used at all. Nevertheless, Schafer and Yarowsky (2002) have shown that these two techniques can be combined efficiently. They use Rapp's co-occurrence vectors in combination with Mann and Yarowsky's EM-trained transducer.

3 Two-Stage Models of Lexical Induction

Following the standard statistical machine translation architecture, we represent the lexicon induction task as a two-stage model. In the first stage, we use the source word to generate a fixed number of candidate translation strings, according to a transducer which represents a particular similarity measure. In the second stage, these candidate strings are filtered through a lexicon of the target language. Candidates that are not words of the target language are thus eliminated.

This article is, like previous work, mostly concerned with the comparison of different similarity measures. However, we extend previous work by introducing two original measures (3.3 and 3.4) and by embedding the measures into the proposed two-stage framework of lexicon induction.

3.1 Levenshtein Distance

One of the simplest string distance measures is the Levenshtein distance. According to it, the distance between two words is defined as the least-cost sequence of edit and identity operations. All edit operations (insertion of one character, substitution of one character by another, and deletion of one character) have a fixed cost of 1. The identity operation (keeping one character from the source word in the target word) has a fixed cost of 0. Levenshtein distance operates on single letters without taking into account contextual features. It can thus be implemented in a memoryless (one-state) transducer. This distance measure is static – it remains the same for all language pairs. We will use Levenshtein distance as a baseline for our experiments.

3.2 Stochastic Transducers Trained with EM

The algorithm presented by Ristad and Yianilos (1998) enables one to train a memoryless stochastic transducer with the Expectation-Maximisation (EM) algorithm. In a stochastic transducer, all transitions represent probabilities (rather than costs or weights).

The transduction probability of a given word pair is the sum of the probabilities of all paths that generate it. The goal of using the EM algorithm is to find the transition probabilities of a stochastic transducer which maximise the likelihood of generating the word pairs given in the training stage. This goal is achieved iteratively by using a training lexicon consisting of correct word pairs. The initial transducer contains uniform probabilities. It is used to transduce the word pairs of the training lexicon, thereby counting all transitions used in this process. Then, the transition probabilities of the transducer are reestimated according to the frequency of usage of the transitions counted before. This new transducer is then used in the next iteration.

This adaptive model is likely to perform better than the static Levenshtein model. For example, to transduce Swiss German dialects to Standard German, inserting *n* or *e* is much more likely than inserting *m* or *i*. Language-independent models cannot predict such specific facts, but stochastic transducers learn them easily. However, these improvements come at a cost: a training bilingual lexicon of sufficient size must be available. For scarce resource languages, such lexicons often need to be built manually.

3.3 Training without a Bilingual Corpus

In order to further reduce the data requirements, we developed another strategy that avoided using a training bilingual lexicon altogether and used other resources for the training step instead. The main idea is to use a simple list of dialect words, and the Standard German lexicon. In doing this, we assume that the structure of the lexicon informs us about which transitions are most frequent. For example, the dialect word *chue 'cow'* does not appear in the Standard German lexicon, but similar words like *Kuh 'cow'*, *Schuh 'shoe'*, *Schule 'school'*, *Sache 'thing'*, *Kühe 'cows'* do. Just by inspecting these most similar existing words, we can conclude that *c* may transform to *k* (*Kuh, Kühe*), that *s* is likely to be inserted (*Schuh, Schule, Sache*), and that *e* may transform to *h* (*Kuh, Schuh*). But we also conclude that none of the letters *c, h, u, e* is likely to transform to *ö* or *f*, just because such words do not exist in the target lexicon. While such statements are coincidental for one single word, they may be sufficiently reliable when induced over a large corpus.

In this model, we use an iterative training algorithm alternating two tasks. The first task is to build a list of hypothesized word pairs by using the dialect word list, the Standard German lexicon, and a transducer[1]: for each dialect word, candidate strings are generated, filtered by the lexicon, and the best candidate is selected. The second task is to train a stochastic transducer with EM, as explained above, on the previously constructed list of word pairs. In the next iteration, this new transducer is used in the first task to obtain a more accurate list of word pairs, which in turn allows us to build a new transducer in the second task. This process is iterated several times to gradually eliminate erroneous word pairs.

The most crucial step is the selection of the best candidate from the list returned by the lexicon filter. We could simply use the word which obtained the highest transduction probability. However, preliminary experiments have shown that the iterative algorithm tends to prefer deletion operations, so that it will converge to generating single-letter words only (which turn out to be present in our lexicon). To avoid this scenario, the length of the suggested candidate words must be taken into account. We therefore simply selected the longest candidate word.[2]

3.4 A Rule-based Model

This last model does not use learning algorithms. It consists of a simple set of transformation rules that are known to be important for the chosen language pair. Marti (1985, 45-64) presents a precise overview of the phonetic correspondences between the Bern dialect and Standard German. Contrary to the learning models, this model is implemented in a weighted transducer with more than one state. Therefore, it allows contextual rules too. For example, we can state that the Swiss German sequence *üech* should be translated to *euch*. Each rule is given a weight of 1, no matter how many characters it concerns. The rule set contains about 50 rules. These rules are then superposed with a Levenshtein transducer, i.e. with context-free edit and identity opera-

[1] In the initialization step, we use a Levenshtein transducer.

[2] In fact, we should select the word with the lowest absolute value of the length difference. The suggested simplification prevents us from being trapped in the single-letter problem and reflects the linguistic reality that Standard German words tend to be longer than dialect words.

tions for each letter. These additional transitions assure that every word *can* be transduced to its target, even if it does not use any of the language-specific rules. The identity transformations of the Levenshtein part weigh 2, and its edit operations weigh 3. With these values, the rules are always preferred to the Levenshtein edit operations. These weights are set somewhat arbitrarily, and further adjustments could slightly improve the results.

4 Experiments and Results

4.1 Data and Training

Written data is difficult to obtain for Swiss German dialects. Most available data is in colloquial style and does not reliably follow orthographic rules. In order to avoid tackling these additional difficulties, we chose a dialect literature book written in the Bern dialect. From this text, a word list was extracted; each word was manually translated to Standard German. Ambiguities were resolved by looking at the word context, and by preferring the alternatives perceived as most frequent.[3] No morphological analysis was performed, so that different inflected forms of the same lemma may occur in the word list. The only preprocessing step concerned the elimination of morpho-phonological variants (*sandhi* phenomena). The whole list contains 5124 entries. For the experiments, 393 entries were excluded because they were foreign language words, proper nouns or Standard German words.[4] From the remaining word pairs, about 92% were annotated as cognate pairs.[5] One half of the corpus was reserved for training the EM-based models, and the other half was used for testing.

The Standard German lexicon is a word list consisting of 202'000 word forms. While the lexicon provides more morphological, syntactic and semantic information, we do not use it in this work.

[3]Further quality improvements could be obtained by including the results of a second annotator, and by allowing multiple translations.

[4]This last category was introduced because the dialect text contained some quotations in Standard German.

[5]This annotation was done by the author, a native speaker of both German varieties. Mann and Yarowsky (2001) consider a word pair as cognate if the Levenshtein distance between the two words is less than 3. Their heuristics is very conservative: it detects 84% of the manually annotated cognate pairs of our corpus.

The test corpus contains 2366 word pairs. 407 pairs (17.2 %) consist of identical words (lower bound). 1801 pairs (76.1%) contain a Standard German word present in the lexicon, and 1687 pairs (71.3%) are cognate pairs, with the Standard German word present in the lexicon (upper bound). It may surprise that many Standard German words of the test corpus do not exist in the lexicon. This concerns mostly *ad-hoc* compound nouns, which cannot be expected to be found in a Standard German lexicon of a reasonable size. Additionally, some Bern dialect words are expressed by two words in Standard German, such as the sequence *ir 'in the (fem.)'* that corresponds to Standard German *in der*. For reasons of computational complexity, our model only looks for single words and will not find such correspondences.

The basic EM model (3.2) was trained in 50 iterations, using a training corpus of 200 word pairs. Interestingly, training on 2000 word pairs did not improve the results. The larger training corpus did not even lead the algorithm to converge faster.[6] The monolingual EM model (3.3) was trained in 10 iterations, each of which involved a basic EM training with 50 iterations on a training corpus of 2000 dialect words.

4.2 Results

As explained above, the first stage of the model takes the dialect words given in the test corpus and generates, for each dialect word, the 500 most similar strings according to the transducer used. This list is then filtered by the lexicon. Between 0 and 20 candidate words remain, depending on how effective the lexicon filter has been. Thus, each source word is associated to a candidate list, which is ordered with respect to the costs or probabilities attributed to the candidates by the transducer. Experiments with 1000 candidate strings yielded comparable results.

Table 1 shows some results for the four models. The table reports the number of times the expected Standard German words appeared anywhere in the corresponding candidate lists (*List*), and the number

[6]This is probably due to the fact that the percentage of identical words is quite high, which facilitates the training. Another reason could be that the orthographical conventions used in the dialect text are quite close to the Standard German ones, so that they conceal some phonetic differences.

		N	L	P	R	F
Levenshtein	List	840	3.1	18.5	35.5	24.3
	Top	671	1.1	32.7	28.4	30.4
EM bilingual	List	1210	4.5	21.4	51.1	30.2
	Top	794	0.7	52.5	33.6	**41.0**
EM mono-	List	1070	5.0	16.6	45.2	24.3
lingual	Top	700	0.7	47.9	29.6	36.6
Rules	List	987	3.2	22.8	41.7	29.5
	Top	909	1.0	45.6	38.4	**41.7**

Table 1: Results. The table shows the absolute numbers of correct target words induced (*N*) and the average lengths of the candidate lists (*L*). The three rightmost columns represent percentage values of precision (*P*), recall (*R*), and F-measure (*F*).

of times they appeared at the best-ranked position of the candidate lists (*Top*). Precision and recall measures are computed as follows:[7]

$$precision = \frac{|correct\ target\ words|}{|unique\ candidate\ words|}$$

$$recall = \frac{|correct\ target\ words|}{|tested\ words|}$$

As Table 1 shows, the three adaptive models perform better than the static Levenshtein distance model. This finding is consistent with the results of Mann and Yarowsky (2001), although our experiments show more clear-cut differences. The stochastic transducer trained on the bilingual corpus obtained similar results to the rule-based system, while the transducer trained on a monolingual corpus performed only slightly better than the baseline. Nevertheless, its performance can be considered to be satisfactory if we take into account that virtually no information on the exact graphemic correspondences has been given. The structure of the lexicon and of the source word list suffice to make some generalisations about graphemic correspondences between two languages. However, it remains to be shown if this method can be extended to more distant language pairs.

In contrast to Levenshtein distance, the bilingual EM model improves the *List* statistics a lot, at the expense of longer candidate lists. However, when comparing the *Top* statistics, the difference between the models is less marked. The rule-based model generates rather short candidate lists, but it still outperforms all other models with respect to the words proposed in first position. The rule-based model obtains high F-measure values, which means that its precision and recall values are better balanced than in the other models.

[7]The words that occur in several candidate lists (i.e., for different source words) are counted only once, hence the term *unique candidate words*.

4.3 Discussion

All models require only a small amount of training or development data. Such data should be available for most language pairs that relate a scarce resource language to a resource-rich language. However, the performances of the rule-based model and the bilingual EM model show that building a training corpus with manually translated word pairs, or alternatively implementing a small rule set, may be worthwhile.

The overall performances of the presented systems may seem poor. Looking at the recall values of the *Top* statistics, our models only induce about one third of the test corpus, or only about half of the test words that *can* be induced by phonetic similarity models – we cannot expect our models to induce non-cognate words or words that are not in the lexicon (see the upper bound values in 4.1). Using the same models, Mann and Yarowsky (2001) induced over 90% of the Spanish-Portuguese cognate vocabulary. One reason for their excellent results lies in their testing procedure. They use a small test corpus of 100 word pairs. For each given word, they compute the transduction costs to each of the 100 possible target words, and select the best-ranked candidate as hypothesized solution. The list of possible target words can thus be explored exhaustively. We tested our models with Mann and Yarowsky's testing procedure and obtained very competitive results (see Table 2). Interestingly, the monolingual EM model performed much worse in this evaluation, a result which could not be expected in light of the results in Table 1.

While Mann and Yarowsky's procedure is very useful to evaluate the performance of different similarity measures and the impact of different language pairs, we believe that it is not representative for the task of lexicon induction. Typically, the list of possible target words (the target lexicon) does not contain 100 words only, but is much larger (202'000 words in our case). This difference has several implications. First, the lexicon is more likely to present very

	Mann and Yarowsky		Our work	
	cognate	full	cognate	full
Levenshtein	92.3	67.9	90.5	85.2
EM bilingual	92.3	67.1	92.2	86.5
EM monolingual			81.9	76.7
Rules			94.1	88.7

Table 2: Comparison between Mann and Yarowsky's results on Spanish-Portuguese (68% of the full vocabulary are cognate pairs), and our results on Swiss German-Standard German (83% cognate pairs). The tests were performed on 10 corpora of 100 word pairs each. The numbers represent the percentage of correctly induced word pairs.

similar words (for example, different inflected forms of the same lexeme), increasing the probability of "near misses". Second, our lexicon is too large to be searched exhaustively. Therefore, we introduced our two-stage approach, whose first stage is completely independent of the lexicon. The drawback of this approach is that for many dialect words, it yields no result at all, because the 500 generated candidates were all non-words. The recall rates could be increased by generating more candidates, but this would lead to longer execution times and lower precision rates.

5 Conclusion and Perspectives

The experiments conducted with various adaptive metrics of graphemic similarity show that in the case of closely related language pairs, lexical induction performances can be increased compared to a static measure like Levenshtein distance. They also show that requirements for training data can be kept rather small. However, these models also show their limits. They only use single word information for training and testing, which means that the rich contextual information encoded in texts, as well as the morphologic and syntactic information available in the target lexicon, cannot be exploited. Future research will focus on integrating contextual information about the syntactic and semantic properties of the words into our models, still keeping in mind the data restrictions for dialects and other scarce resource languages. Such additional information could be implemented by adding a third step to our two-stage model.

Acknowledgements

We thank Paola Merlo for her precious and useful comments on this work. We also thank Eric Wehrli for allowing us to use the LATL Standard German lexicon.

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Wilbert Heeringa, Peter Kleiweg, Charlotte Gooskens, and John Nerbonne. 2006. Evaluation of string distance algorithms for dialectology. In *Proceedings of the ACL Workshop on Linguistic Distances*, pages 51–62, Sydney, Australia.

Rebecca Hwa, Carol Nichols, and Khalil Sima'an. 2006. Corpus variations for translation lexicon induction. In *Proceedings of AMTA'06*, pages 74–81, Cambridge, MA, USA.

Martin Jansche. 2003. *Inference of String Mappings for Language Technology*. Ph.D. thesis, Ohio State University.

Grzegorz Kondrak and Tarek Sherif. 2006. Evaluation of several phonetic similarity algorithms on the task of cognate identification. In *Proceedings of the ACL Workshop on Linguistic Distances*, pages 43–50, Sydney, Australia.

Gideon S. Mann and David Yarowsky. 2001. Multipath translation lexicon induction via bridge languages. In *Proceedings of NAACL'01*, Pittsburgh, PA, USA.

Werner Marti. 1985. *Berndeutsch-Grammatik*. Francke Verlag, Bern, Switzerland.

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of ACL'99*, pages 519–526, Maryland, USA.

Eric Sven Ristad and Peter N. Yianilos. 1998. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532.

Charles Schafer and David Yarowsky. 2002. Inducing translation lexicons via diverse similarity measures and bridge languages. In *Proceedings of CoNLL'02*, pages 146–152, Taipei, Taiwan.

Identifying Linguistic Structure in a Quantitative Analysis of Dialect Pronunciation

Jelena Prokić
Alfa-Informatica
University of Groningen
The Netherlands
j.prokic@rug.nl

Abstract

The aim of this paper is to present a new method for identifying linguistic structure in the aggregate analysis of the language variation. The method consists of extracting the most frequent sound correspondences from the aligned transcriptions of words. Based on the extracted correspondences every site is compared to all other sites, and a correspondence index is calculated for each site. This method enables us to identify sound alternations responsible for dialect divisions and to measure the extent to which each alternation is responsible for the divisions obtained by the aggregate analysis.

1 Introduction

Computational dialectometry is a multidisciplinary field that uses quantitative methods in order to measure linguistic differences between the dialects. The distances between the dialects are measured at different levels (phonetic, lexical, syntactic) by aggregating over entire data sets. The aggregate analyses do not expose the underlying linguistic structure, i.e. the specific linguistic elements that contributed to the differences between the dialects. This is very often seen as one of the main drawbacks of the dialectometry techniques and dialectometry itself. Two attempts to overcome this drawback are presented in Nerbonne (2005) and Nerbonne (2006). In both of these papers the identification of linguistic structure in the aggregate analysis is based on the analysis of the pronunciation of the vowels found in the data set.

In work presented in this paper the identification of linguistic structure in the aggregate analysis is based on the automatic extraction of regular sound correspondences which are further quantified in order to characterize each site based on the frequency of a certain sound extracted from the pool of the site's pronunciation. The results show that identification of regular sound correspondences can be successfully applied to the task of identifying linguistic structure in the aggregate analysis of dialects based on word pronunciations.

The rest of the paper is structured as follows. Section 2 gives an overview of the work previously done in the areas covered in this paper. In Section 3 more information on the aggregate analysis of Bulgarian dialects is given. Work done on the identification of regular sound correspondences and their quantification is presented in Section 4. Conclusion and suggestions for future work are given in Section 5.

2 Previous Work

The work presented in this paper can be divided in two parts: the aggregate analysis of Bulgarian dialects on one hand, and the identification of linguistic structure in the aggregate analysis on the other. In this section the work closely related to the one presented in this paper will be described in more detail.

2.1 Aggregate Analysis of Bulgarian

Dialectometry produces aggregate analyses of the dialect variations and has been done for different languages. For several languages aggregate analyses have been successfully developed which distinguish various dialect areas within the language area. The

Proceedings of the ACL 2007 Student Research Workshop, pages 61–66,
Prague, June 2007.

most closely related to the work presented in this paper is quantitative analysis of Bulgarian dialect pronunciation reported in Osenova et al. (2007).

In work done by Osenova et al. (2007) aggregate analysis of pronunciation differences for Bulgarian was done on the data set that comprised 36 word pronunciations from 490 sites. The data was digitalized from the four-volume set of Atlases of Bulgarian Dialects (Stojkov and Bernstein, 1964; Stojkov, 1966; Stojkov et al., 1974; Stojkov et al., 1981). Pronunciations of the same words were aligned and compared using L04.[1] Results were analyzed using cluster analysis, composite clustering, and multidimensional scaling. The analyses showed that results obtained using aggregate analysis of word pronunciations mostly conform with the traditional phonetic classification of Bulgarian dialects as presented in Stojkov (2002).

2.2 Extraction of Linguistic Structure

Although techniques in dialectometry have shown to be successful in the analysis of the dialect variation, all of them aggregate over the entire available data, failing to extract linguistic structure from the aggregate analysis. Two attempts to overcome this withdraw are presented in Nerbonne (2005) and Nerbonne (2006).

Nerbonne (2005) suggests aggregating over a linguistically interesting subset of the data. Nerbonne compares aggregate analysis restricted to vowel differences to those using the complete data set. Results have shown that vowels are probably responsible for a great deal of aggregate differences, since there was high correlation between differences obtained only by using vowels and by using complete transcriptions (r = 0.936). Two ways of aggregate analysis also resulted in comparable maps. However, no other subset has been analyzed in this paper, making it impossible to conclude how successful other subsets would be if similar analysis was done.

The second paper (Nerbonne, 2006) applies factor analysis to the result of the dialectometric analysis in order to extract linguistic structure. The study focuses on the pronunciation of vowels found in the data. Out of 1132 different vowels found in the data 204 vowel positions are investigated, where a vowel position is, e.g., the first vowel in the word 'Washington' or the second vowel in the word 'thirty'. Factor analysis has shown that 3 factors are most important, explaining 35% of the total amount of variance. The main drawback of applying this technique in dialectometry is that it is not directly related to the aggregate analysis, but is rather an independent step. Just as in Nerbonne (2005), only vowels were examined.

2.3 Sound Correspondences

In his PhD thesis Kondrak (Kondrak, 2002) presents techniques and algorithms for the reconstruction of the proto-languages from cognates. In Chapter 6 the focus is on the automatic determination of sound correspondences in bilingual word lists and the identification of cognates on the basis of extracted correspondences. Kondrak (2002) adopted Melamed's parameter estimation models (Melamed, 2000) used in statistical machine translation and successfully applied them to determination of sound correspondences, i.e. diachronic phonology. Kondrak induced a model of sound correspondence in bilingual word lists, where phoneme pairs with the highest scores represent the most likely correspondences. The more regular sound correspondences the two words share, the more likely it is that they are cognates and not borrowings.

In this paper the identification of sound correspondences will be used to extract linguistic elements (i.e. phones) responsible for the dialect divisions. The method presented in this study differs greatly from Kondrak's in that he uses regular sound correspondences to directly compare two words and determine if they are cognates. In this study extracted sound correspondences are further quantified in order to characterize each site in the data set by assigning it a unique index. This is the first time that this method has been applied in dialectometry.

3 Aggregate Analysis

In the first phase of this project L04 toolkit was used in order to make an aggregate analysis of Bulgarian dialects. In this section more information on the data set used in the project, as well as on the process of the aggregate analysis will be given.

[1] L04 is a freely available software used for dialectometry and cartography. It can be found at http://www.let.rug.nl/kleiweg/L04/

3.1 Data Set

The data used in this research, as well as the research itself, are part of the project Buldialect—Measuring linguistic unity and diversity in Europe.[2] The data set consisted of pronunciations of 117 words collected from 84 sites equally distributed all over Bulgaria. It comprises nouns, pronouns, adjectives, verbs, adverbs and prepositions which can be found in different word forms (singular and plural, 1st, 2nd, and 3rd person verb forms, etc.).

3.2 Measuring of Dialect Distances

Aggregate analysis of Bulgarian dialects done in this project was based on the phonetic distances between the various pronunciations of a set of words. No morphological, lexical, or syntactic variation was taken into account.

First, all word pronunciations were aligned based on the following principles: a) a vowel can match only with the vowel b) a consonant can match only with the consonant c) [j] can match both vowels and consonants.

An example of the alignment of two pronunciations is given in Figure 1.[3]

g	l	ˈɑ	v	ɑ
g	l	ə	v	ˈɤ
		1		1

Figure 1: Alignment of word pronunciation pair

The alignments were carried out using the Levensthein algorithm,[4] which also results in the calculation of a distance between each pair of words. The distance is the smallest number of insertions, deletions, and substitutions needed to transform one string to the other. In this work all three operations were assigned the same value—1. All words are represented as series of phones which are not further defined. The result of comparing two phones can be 1 or 0; they either match or they don't. In Figure 1 the cheapest way to transform one pronunciation to the other would be by making two substitutions: [ˈɑ] should be replaced by [ə], and [ɑ] by [ˈɤ], meaning that the distance between these two pronunciations is 2. The distance between each pair of pronunciations was further normalized by the length of the longest alignment that gives the minimal cost.[5] After normalization, we get the final distance between two strings, which is 0.4 (2/5) in the example shown in Figure 1. If there are more plausible alignments with the minimal cost, the longest is preferred. Word pronunciations collected from all sites are aligned and compared in this fashion, allowing us to calculate the distance between each pair of sites. The difference between two locations is the mean of all differences between words collected from these two sites.

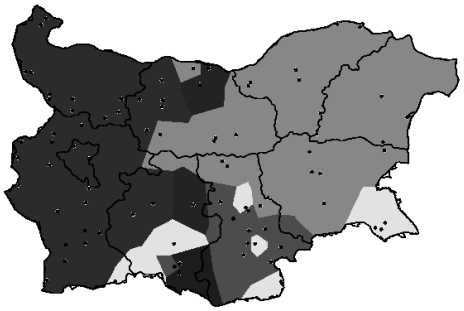


Figure 2: Classification map

The results were analyzed using clustering (Figure 2) and multidimensional scaling (Figure 3). Clustering is a common technique in a statistical data analysis based on a partition of a set of objects into groups or clusters (Manning and Schütze, 1999). Multidimensional scaling is data analysis technique that provides a spatial display of the data revealing relationships between the instances in the data set (Davison, 1992). On both the maps the biggest division is between East and West. The border between these two areas goes around Pleven and Teteven, and it is the border of "yat" realization as presented in the traditional dialectological atlases (Stojkov, 2002). The most incoherent area is the

[2]The project is sponsored by Volkswagen Stiftung. More information can be found at http://www.sfs.uni-tuebingen.de/dialectometry

[3]For technical reasons primary stress is indicated by a high vertical line before the syllable's vowel.

[4]Detailed explanation of Levensthein algorithm can be found in Heeringa (2004).

[5]An interesting discussion on the normalization by length can be found in Heeringa et al. (2006). In this paper the authors report that contrary to results from previous work (Heeringa, 2004) non-normalized string distance measures are superior to normalized ones.

area of Rodopi mountain, and the dialects present in this area show the greatest similarity with the dialects found in the Southeastern part around Malko Tyrnovo. On the map in Figure 3 it is also possible to distinguish the area around Golica and Kozichino on the East, which conforms to the maps found in Stojkov (2002). Results of the aggregate analysis conform both to the traditional maps presented in Stojkov (2002), and to the work reported in Osenova et al. (2007).

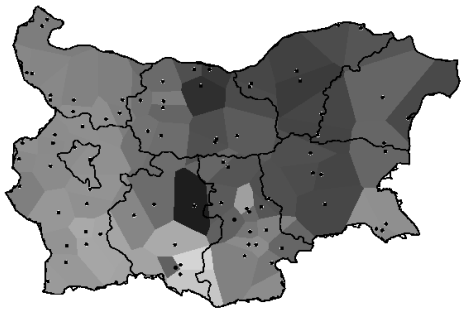


Figure 3: MDS map

4 Regular Sound Correspondences

The same data used for the aggregate analysis was reused to extract sound correspondences and to identify underlying linguistic structure in the aggregate analysis. The method and the obtained results will be presented in more detail.

4.1 Method

From the aligned pairs of word pronunciations all non-matching segments were extracted and sorted according to their frequency. In the entire data set there were 683 different pairs of sound correspondences that appeared 955199 times.

e	i	36565	j	-	21361
ə	ɤ	26398	ɑ	ə	20515
o	u	26108	e	ˈe	19934
ˈɒ	ˈe	23689	r	rʲ	19787
v	-	22100	ˈɤ	-	18867

Table 1: Most frequent sound correspondences

The most frequent correspondences were taken to be the most important sound alternations responsible for dialect variation. The method was tested on the 10 most frequent correspondences which were responsible for the 25% of sound alternations in the whole data set.

In order to determine which of the extracted sound correspondences is responsible for which of the divisions present in the aggregate analysis, each site was compared to all other sites with respect to the 10 most frequent sound correspondences. For each pair of sites all sound correspondences were extracted, including both matching and non-matching segments. For further analysis it was important to distinguish which sound comes from which place.

For each pair of the sound correspondences from Table 1 a correspondence index is calculated for each site using the following formula:

$$\frac{1}{n-1}\sum_{i=1,j\neq i}^{n} s_i \longrightarrow s'_j \qquad (1)$$

where n represents the number of sites, and $s_i \longrightarrow s'_j$ the comparison of each two sites (i, j) with respect to the sound correspondence s/s'. $s_i \longrightarrow s'_j$ is calculated applying the following formula:

$$\frac{|s_i, s'_j|}{|s_i, s'_j| + |s_i, s_j|} \qquad (2)$$

In the above formula s_i and s'_j stand for the pair of sounds involved in one of the most frequent sound correspondences from Table 1. $|s_i, s'_j|$ represents the number of times s is seen in the word pronunciations collected at site i, aligned with the s' in word pronunciations collected at site j. $|s_i, s_j|$ is the number of times s stayed unchanged. For each pair of sound correspondences a correspondence index was calculated for the s, s' correspondence, as well as for the s', s correspondence. For example, for the pair of correspondences [e] and [i], the relation of [e] corresponding to [i] is separated from the relation of [i] corresponding to [e].[6]

For example, the indices for the sites Aldomirovci and Borisovo with respect to the sound correspondence [e]-[i] were calculated in the following way. In the file with the sound correspondences extracted from all aligned word pronunciations collected at

[6] It would also be possible to modify this formula and calculate the ratio of s to s corresponding to any other sound. In this case the result would be a very small number of sites with the very high correspondence index.

these two sites, the algorithm searches for pairs represented in Table 2:

Aldomirovci	e	i	e
Borisovo	i	e	e
no. of correspondences	24	0	3

Table 2: How often [e] corresponds to [i] and [e]

For each of the sites the indices were calculated using the above formula. The index for site i (Aldomirovci) was:

$$\frac{|e,i|}{|e,i|+|e,e|} = \frac{24}{24+3} = 0.89 \qquad (3)$$

The index for site j (Borisovo) was calculated in the similar fashion from the Table 2:

$$\frac{|e,i|}{|e,i|+|e,e|} = \frac{0}{0+3} = 0.00 \qquad (4)$$

Each of these two sites was compared to all other sites with respect to the [e]-[i] correspondence resulting in 83 indices for each site. The general correspondence index for each site represents the mean of all 83 indices. For the site i (Aldomirovci) general index was 0.40, and for the site j (Borisovo) 0.21. Sites with the higher values of the general correspondence index represent the sites where sound [e] tends to be present, with respect to the [e]-[i] correspondence (see Figure 4). In the same fashion general correspondence indices were calculated for every site with respect to each pair of the most frequent correspondences (Table 1).

4.2 Results

The methods described in the previous section were applied to all phone pairs from the Table 1, resulting in 17 different divisions of the sites.[7]

Data obtained by the analysis of sound correspondences, i.e. indices of correspondences for sites was used to draw maps in which every site is set off by Voronoi tessellation from all other sites, and shaded based on the value of the general correspondence index. Light polygons on the map represent areas with the higher values of the correspondence index, i.e. areas where the first sound in the examined alternation tends to be present. This technique enables us to visualize the geographical distribution of the examined sounds. For example, map in Figure 4 represents geographical distribution of sound [e] with respect to the [e]-[i] correspondence, while map in Figure 5 reveals the presence of the sound [i] with respect to the [i]-[e] correspondence.

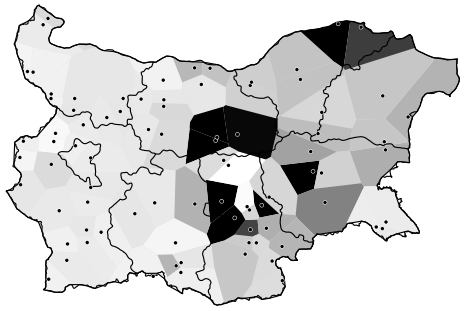


Figure 4: Distribution of [e] sound

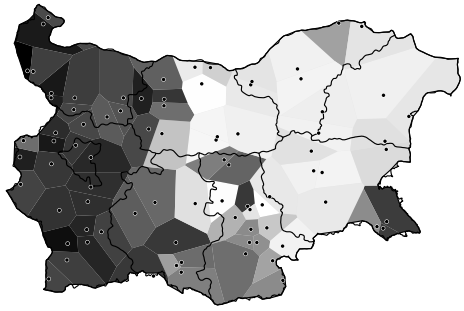


Figure 5: Distribution of [i] sound

In order to compare the dialect divisions obtained by the aggregate analysis, and those based on the general correspondence index for a certain phone pair, correlation coefficient was calculated for these 2 sets of distances. The results are shown in Table 3. Dialect divisions based on the [r]-[r^j] and [i]-[e] alternations have the highest correlation with the distances obtained by the aggregate analysis. The square of the Pearson correlation coefficient presented in column 3 enables us to see that 39.0% and 30.7% of the variance in the aggregate analysis can be explained by these two sound alternations.

[7] For three pairs where one sound doesn't have a corresponding one (when there was an insertion or deletion) it is not possible to calculate an index. Formulas for comparing two sites from the previous section would always give value 1 for the index.

Correspondence	Correlation	r^2x100(%)
[e]-[i]	0.19	3.7
[i]-[e]	0.55	30.7
[ə]-[ɤ]	0.26	6.7
[ɤ]-[ə]	0.23	5.3
[o]-[u]	0.49	24.4
[u]-[o]	0.43	18.9
[ˈɑ]-[ˈe]	0.49	24.3
[ˈe]-[ˈɑ]	0.38	14.2
[v]- -	0.14	2.0
[j]- -	0.20	4.0
[ɑ]-[ə]	0.51	26.5
[ə]-[ɑ]	0.26	7.0
[e]-[ˈe]	0.18	3.2
[ˈe]-[e]	0.23	5.2
[r]-[rʲ]	0.62	39.0
[rʲ]-[r]	0.53	28.1
[ˈɤ]- -	0.17	2.9

Table 3: Correlation coefficient

5 Conclusion and Future Work

The dialect division of Bulgaria based on the aggregate analysis presented in this paper conforms both to traditional maps (Stojkov, 2002) and to the work reported in Osenova et al. (2007), suggesting that the novel data used in this project is representative. The method of quantification of regular sound correspondences described in the second part of the paper was successful in the identification of the underlying linguistic structure of the dialect divisions. It is an important step towards more general investigation of the role of the regular sound changes in the language dialect variation. The main drawback of the method is that it analyzes one sound alternation at the time, while in the real data it is often the case that one sound corresponds to several other sounds and that sound correspondences involve series of segments.

In future work some kind of a feature representation of segments should be included in the analysis in order to deal with the drawbacks noted. It would also be very important to analyze the context in which examined sounds appear, since we can talk about regular sound changes only with respect to the certain phonological environments.

References

Mark L. Davison. 1992. *Multidimensional scaling*. Melbourne, Fl. CA: Krieger Publishing Company.

Wilbert Heeringa, Peter Kleiweg, Charlotte Gooskens, and John Nerbonne. 2006. Evaluation of String Distance Algorithms for Dialectology. In John Nerbonne and Erhard Hinrichs, editors, *Linguistic Distances*. Workshop at the joint conference of International Committee on Computational Linguistics and the Association for Computational Linguistics, Sydney.

Wilbert Heeringa. 2004. *Measuring Dialect Pronunciation Differences using Levensthein Distance*. PhD Thesis, University of Groningen.

Grzegorz Kondrak. 2002. *Algorithms for Language Reconstruction*. PhD Thesis, University of Toronto.

Chris Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press. Cambridge, MA.

I. Dan Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

John Nerbonne. 2005. Various Variation Aggregates in the LAMSAS South. In Catherine Davis and Michael Picone, editors, *Language Variety in the South III*. University of Alabama Press, Tuscaloosa.

John Nerbonne. 2006. Identifying Linguistic Structure in Aggregate Comparison. *Literary and Linguistic Computing*, 21(4).

Petya Osenova, Wilbert Heeringa, and John Nerbonne. 2007. A Quantitive Analysis of Bulgarian Dialect Pronunciation. Accepted to appear in *Zeitschrift für slavische Philologie*.

Stojko Stojkov and Samuil B. Bernstein. 1964. *Atlas of Bulgarian Dialects: Southeastern Bulgaria*. Publishing House of Bulgarian Academy of Science, volume I, Sofia, Bulgaria.

Stojko Stojkov, Kiril Mirchev, Ivan Kochev, and Maksim Mladenov. 1974. *Atlas of Bulgarian Dialects: Southwestern Bulgaria*. Publishing House of Bulgarian Academy of Science, volume III, Sofia, Bulgaria.

Stojko Stojkov, Ivan Kochev, and Maksim Mladenov. 1981. *Atlas of Bulgarian Dialects: Northwestern Bulgaria*. Publishing House of Bulgarian Academy of Science, volume IV, Sofia, Bulgaria.

Stojko Stojkov. 1966. *Atlas of Bulgarian Dialects: Northeastern Bulgaria*. Publishing House of Bulgarian Academy of Science, volume II, Sofia, Bulgaria.

Stojko Stojkov. 2002. *Bulgarska dialektologiya*. Sofia, 4th ed.

Towards a Computational Treatment of Superlatives

Silke Scheible
Institute for Communicating and Collaborative Systems (ICCS)
School of Informatics
University of Edinburgh
S.Scheible@sms.ed.ac.uk

Abstract

I propose a computational treatment of superlatives, starting with superlative constructions and the main challenges in automatically recognising and extracting their components. Initial experimental evidence is provided for the value of the proposed work for Question Answering. I also briefly discuss its potential value for Sentiment Detection and Opinion Extraction.

1 Introduction

Although superlatives are frequently found in natural language, with the exception of recent work by Bos and Nissim (2006) and Jindal and Liu (2006), they have not yet been investigated within a computational framework. And within the framework of theoretical linguistics, studies of superlatives have mainly focused on particular semantic properties that may only rarely occur in natural language (Szabolcsi, 1986; Heim, 1999).

My goal is a comprehensive computational treatment of superlatives. The initial question I address is how useful information can be automatically extracted from superlative constructions. Due to the great semantic complexity and the variety of syntactic structures in which superlatives occur, this is a major challenge. However, meeting it will benefit NLP applications such as Question Answering, Sentiment Detection and Opinion Extraction, and Ontology Learning.

2 What are Superlatives?

In linguistics, the term "superlative" describes a well-defined class of word forms which (in English) are derived from adjectives or adverbs in two different ways: Inflectionally, where the suffix *-est* is appended to the base form of the adjective or adverb (e.g. *lowest*, *nicest*, *smartest*), or analytically, where the base adjective/adverb is preceded by the markers *most*/*least* (e.g. *most interesting*, *least beautiful*). Certain adjectives and adverbs have irregular superlative forms: *good* (*best*), *bad* (*worst*), *far* (*furthest*/*farthest*), *well* (*best*), *badly* (*worst*), *much* (*most*), and *little* (*least*).

In order to be able to form superlatives, adjectives and adverbs must be *gradable*, which means that it must be possible to place them on a scale of comparison, at a position higher or lower than the one indicated by the adjective/adverb alone. In English, this can be done by using the comparative and superlative forms of the adjective or adverb:

[1] (a) Maths is ***more difficult*** than Physics.
(b) Chemistry is ***less difficult*** than Physics.
[2] (a) Maths is the ***most difficult*** subject at school.
(b) History is the ***least difficult*** subject at school.

The ***comparative*** form of an adjective or adverb is commonly used to compare two entities to one another with respect to a certain quality. For example, in [1], Maths is located at a higher point on the difficulty scale than Physics, and Chemistry at a lower point. The ***superlative*** form of an adjective is usually used to compare one entity to a set of other entities, and expresses the end spectrum of the scale: In [2], Maths and History are located at the highest and lowest points of the difficulty scale, respectively, while all the other subjects at school range somewhere in between.

3 Why are Superlatives Interesting?

From a computational perspective, superlatives are of interest because they express a comparison

Proceedings of the ACL 2007 Student Research Workshop, pages 67–72,
Prague, June 2007.

between a target entity (indicated in bold) and its comparison set (underlined), as in:

[3] **The blue whale** is the *largest* mammal.

Here, the target *blue whale* is compared to the comparison set of *mammals*. Milosavljevic (1999) has investigated the discourse purpose of different types of comparisons. She classifies superlatives as a type of *set complement comparison*, whose purpose is to highlight the uniqueness of the target entity compared to its contrast set.

My initial investigation of superlative forms showed that there are two types of relation that hold between a target and its comparison set:

Relation 1: *Superlative relation*
Relation 2: *IS-A relation*

The *superlative relation* specifies a property which all members of the set share, but which the target has the highest (or lowest) degree or value of. The *IS-A* (or *hypernymy*) *relation* expresses the membership of the target in the comparison class (e.g. its parent class in a generalisation hierarchy). Both of these relations are of great interest from a relation extraction point of view, and in Section 6, I discuss their use in applications such as Question Answering (QA) and Sentiment Detection and Opinion Extraction. That a computational treatment of superlatives is a worthwhile undertaking is also supported by the frequency of superlative forms in ordinary text: In a 250,000 word subcorpus of the WSJ corpus [1] I found 602 instances (which amounts to roughly one superlative form in every 17 sentences), while in the corpus of animal encyclopaedia entries used by Milosavljevic (1999), there were 1059 superlative forms in 250,000 words (about one superlative form in every 11 sentences).[2] These results show significant variation in the distribution of superlatives across different text genres.

4 Elements of a Computational Treatment of Superlatives

For an interpretation of comparisons, two things are generally of interest: What is being compared, and with respect to what this comparison is made. Given that superlatives express set comparisons, a computational treatment should therefore help to identify:

a) The **target** and comparison set
b) The type of ***superlative relation*** that holds between them (cf. Relation 1 in Section 3)

However, this task is far from straightforward, firstly because superlatives occur in a variety of different constructions. Consider for example:

[4] The pipe organ is the ***largest*** instrument.
[5] Of all the musicians in the brass band, Peter plays the ***largest*** instrument.
[6] The human foot is ***narrowest*** at the heel.
[7] First Class mail usually arrives the ***fastest***.
[8] This year, Jodie Foster was voted ***best*** actress.
[9] I will get there at 8 at the ***earliest***.
[10] I am ***most tired*** of your constant moaning.
[11] ***Most*** successful bands are from the U.S.

All these examples contain a superlative form (bold italics). However, they differ not only in their syntactic structure, but also in the way in which they express a comparison. Example [4] contains a clear-cut comparison between a target item and its comparison set: The pipe organ is compared to all other instruments with respect to its size. However, although the superlative form in [4] occurs in the same noun phrase as in [5], the comparisons differ: What is being compared in [5] is not just the instruments, but the musicians in the brass band with respect to the size of the instrument that they play. In example [6], the target and comparison set are even less easy to identify. What is being compared here is not the human foot and a set of other entities, but rather different *parts* of the human foot. In contrast to the first two examples, this superlative form is not incorporated in a noun phrase, but occurs freely in the sentence. The same applies to *fastest* in example [7], which is an adverbial superlative. The comparison here is between First Class mail and other mail delivery services. Finally, examples [8] to [11] are not proper comparisons: *best actress* in [8] is an idiomatic expression, *earliest* in [9] is part of a so-called PP superlative construction (Corver and Matushansky, 2006), and [10] and [11] describe two non-comparative uses of *most*, as an intensifier and a proportional quantifier, respectively (Huddleston and Pullum, 2002).

Initially, I will focus on cases like [4], which I call IS-A superlatives because they make explicit the IS-A relation that holds between target and comparison set (cf. Relation 2 in Section 3). They

[1] www.ldc.upenn.edu/Catalog/LDC2000T43.html
[2] In the following, these 250,000 word subcorpora will be referred to as SubWSJ and SubAC.

are a good initial focus for a computational approach because both their target and comparison set are explicitly realised in the text (usually, though not necessarily, in the same sentence). Common surface forms of IS-A superlatives involve the verb "to be" ([12]-[14]), appositive position [15], and other copula verbs or expressions ([16] and [17]):

[12] **The blue whale** is the ***largest*** mammal.
[13] **The blue whale** is the ***largest*** of all mammals.
[14] Of all mammals, **the blue whale** is the ***largest***.
[15] The ***largest*** mammal, the blue whale**,** weighs...
[16] **The ostrich** is considered the ***largest*** bird.
[17] **Mexico** claimed to be the ***most peaceful*** country in the Americas.

IS-A superlatives are also the most frequent type of superlative comparison, with 176 instances in SubWSJ (ca. 30% of all superlative forms), and 350 instances in SubAC (ca. 33% of all superlative forms).

The second major problem in a computational treatment of superlatives is to correctly identify and interpret the comparison set. The challenge lies in the fact that it can be restricted in a variety of ways, for example by preceding possessives and premodifiers, or by postmodifiers such as PPs and various kinds of clauses. Consider for example:

[18] **VW** is [Europe's ***largest*** **maker** of cars].
[19] **VW** is [the ***largest*** European car **maker** with this product range].
[20] **VW** is [the ***largest*** car **maker** in Europe] with an impressive product range.
[21] In China, **VW** is by far [the ***largest*** car **maker**].

The phrases *of cars* and *car* in [18] and [19] both have the role of specifying the type of *maker* that constitutes the comparison set. The phrases *Europe's*, *European* and *in Europe* occur in determinative, premodifying, and postmodifying position, respectively, but all have the role of restricting the set of car makers to the ones in Europe. And finally, the "with" PP phrases in [19] and [20] both occur in postmodifying position, but differ in that the one in [19] is involved in the comparison, while the one in [20] is non-restrictive. In addition, restrictors of the comparison can also occur elsewhere in the sentence, as shown by the PP phrase and adverbial in [21]. It is evident that in order to extract useful and reliable information, a thorough syntactic and semantic analysis of superlative constructions is required.

5 Previous Approaches

5.1 Jindal and Liu (2006)

Jindal and Liu (2006) propose the study of comparative sentence mining, by which they mean the study of sentences that express "an ordering relation between two sets of entities with respect to some common features" (2006). They consider three kinds of relations*: non-equal gradable* (e.g. *better*), *equative* (e.g. *as good as*) and *superlative* (e.g. *best*). Having identified comparative sentences in a given text, the task is to extract comparative relations from them, in form of a vector like *(relationWord, features, entityS1, entityS2)*, where *relationWord* represents the keyword used to express a comparative relation, *features* are a set of features being compared, and *entityS1* and *entityS2* are the sets of entities being compared, where *entityS1* appears to the left of the relation word and *entityS2* to the right. Thus, for a sentence like "*Canon's optics is better than those of Sony and Nikon*", the system is expected to extract the vector (*better*, {*optics*}, {*Canon*}, {*Sony, Nikon*}).

For extracting the comparative relations, Jindal and Liu use what they call *label sequential rules* (LSR), mainly based on POS tags. Their overall F-score for this extraction task is 72%, a big improvement to the 58% achieved by their baseline system. Although this result suggests that their system represents a powerful way of dealing with superlatives computationally, a closer inspection of their approach, and in particular of the gold standard data set, reveals some serious problems.

Jindal and Liu claim that for superlatives, the *entityS2* slot is "normally empty" (2006). Assuming that the members of *entityS2* usually represent the comparison set, this is somewhat counter-intuitive. A look at the data shows that even in cases where the comparison set is explicitly mentioned in the sentence, the *entityS2* slot remains empty. For example, although the comparison set in [22] is represented by the string *these 2nd generation jukeboxes (ipod , archos , dell , samsung)*, it is not annotated as *entityS2* in the gold standard:

[22] `all reviews i 've seen seem to indicate that the creative mp3 jukeboxes have the best sound quality of these 2nd generation jukeboxes ( ipod , archos , dell , samsung ) .`

(*best*, {*sound quality*}, {*creative mp3 jukeboxes*}, {--})

Jindal and Liu (2006)

Furthermore, Jindal and Liu do not distinguish between different types of superlatives. In constructions where the superlative form is incorporated into an NP, Jindal and Liu consistently interpret the string following the superlative form as a "feature", which is appropriate for cases like [22], but does not apply to superlative sentences involving the copula verb "to be" (as e.g. in [4]), where the NP head denotes the comparison set rather than a feature. A further major problem is that restrictions on the comparison set as the ones discussed in Section 4 and negation are not considered at all. Therefore, the reliability of the output produced by the system is questionable.

5.2 Bos and Nissim (2006)

In contrast to Jindal and Liu (2006), Bos and Nissim's (2006) approach to superlatives is explicitly semantic. They describe an implementation of a system that can automatically detect superlatives, and determine the correct comparison set for attributive cases, where the superlative form is incorporated into an NP. For example in [23], the comparison set of the superlative *oldest* spans from word 3 to word 7:

[23] wsj00 1690 [...] Scope: 3-7
The ***oldest*** bell-ringing group in the country , the Ancient Society of College Youths , founded in 1637 , remains male-only , [...] .
(Bos and Nissim 2006)

Bos and Nissim's system, called DLA (Deep Linguistic Analysis), uses a wide-coverage parser to produce semantic representations of superlative sentences, which are then exploited to select the comparison set among attributive cases. Compared with a baseline result, the results for this are very good, with an accuracy of 69%-83%.

The results are clearly very promising and show that comparison sets can be identified with high accuracy. However, this only represents a first step towards the goal of the present work. Apart from the superlative keyword *oldest*, the only information example [23] provides is that the comparison set spans from word 3 to word 7. However, what would be interesting to know is that the target of the comparison appears in the same sentence and spans from word 9 to word 14 (*the Ancient Society of College Youths*). Furthermore, no analysis of the semantic roles of the constituents of the resulting string is carried out: We lose the information that *the Ancient Society of College Youths* **IS-A** kind of *bell-ringing group*, and that the set of bell-ringing groups is restricted in location (*in the country*).

6 Applications

The proposed work will be beneficial for a variety of areas in NLP, for example Question Answering (QA), Sentiment Detection/Opinion Extraction, Ontology Learning, or Natural Language Generation. In this section I will discuss applications in the first two areas.

6.1 Question Answering

In open-domain QA, the proposed work will be useful for answering two question types. A superlative sentence like [24], found in a corpus, can be used to answer both a factoid question [25] and a definition question [26]:

[24] A: **The Nile** is the ***longest*** river in the world.
[25] Q: **What** is the world's ***longest*** river?
[26] Q: **What** is **the Nile**?

Here I will focus on the latter. The common assumption that superlatives are useful with respect to answering definition questions is based on the observation that superlatives like the one in [24] both place an entity in a generalisation hierarchy, and distinguish it from its contrast set.

To investigate this assumption, I carried out a study involving the TREC QA "other" question nuggets[3], which are snippets of text that contain relevant information for the definition of a specific topic. In a recent study of judgement consistency (Lin and Demner-Fushman, 2006), relevant nuggets were judged as either 'vital' or 'okay' by 10 different judges rather than the single assessor standardly used in TREC. For example, the first three nuggets for the topic "Merck & Co." are:

[27] Qid 75.8: 'other' question for target Merck & Co.
75.8 1 vital World's ***largest*** drug company.
75.8 2 okay Spent $1.68 billion on RandD in 1997.
75.8 3 okay Has experience finding new uses for established drugs.
(taken from TREC 2005; 'vital' and 'okay' reflect the opinion of the TREC evaluator.)

My investigation of the nugget judgements in Lin and Demner-Fushman's study yielded two in-

[3] http://trec.nist.gov/data/qa.html

teresting results: First of all, a relatively high proportion of relevant nuggets contains superlatives: On average, there is one superlative nugget for at least half of the TREC topics. Secondly, of 69 superlative nuggets altogether, 32 (i.e. almost half) are judged "vital" by more than 9 assessors.

Furthermore, I found that the nuggets can be distinguished by how the question target (i.e. the TREC topic, referred to as T1) relates to the superlative target (T2): In the first case, T1 and T2 coincide (referred to as class S1). In the second one, T2 is part of or closely related to T1, or T2 is part of the comparison set (class S2). In the third case, T1 is unrelated or only distantly related to T2 (S3). Table 1 shows examples of each class:

	T1	**nugget** (T2 in bold)
S1	Merck & Co.	World's ***largest*** drug company
S2	Florence Nightingale	**Nightingale Medal** *highest* international nurses award
S3	Kurds	**Irbil** *largest* city controlled by Kurds

Table 1. Examples of superlative nuggets.

Of the 69 nuggets containing superlatives, 46 fall into subclass S1, 15 into subclass S2 and 8 into subclass S3. While I noted earlier that 32/69 (46%) of superlative-containing nuggets were judged vital by more than 9 assessors, these judgements are not equally distributed over the subclasses: Table 2 shows that 87% of S1 judgements are 'vital', while only 38% of S3 judgements are.

	number of instances	% of "vital" judgements	% of "okay" judgements
S1	46	87%	13%
S2	15	59%	40%
S3	8	38%	60%

Table 2. Ratings of the classes S1, S2, and S3.

These results strongly suggest that the presence of superlatives, and in particular S1 membership, is a good indicator of the importance of nuggets, and thus for answering definition questions. Some experiments carried out in the framework of TREC 2006 (Kaisser et al., 2006), however, showed that superlatives alone are not a winning indicator of nugget importance, but S1 membership may be. A similar simple technique was used by Ahn et al. (2005) and by Razmara and Kosseim (2007). All just looked for the presence of a superlative and raised the score without further analysing the type of superlative or its role in the sentence. This calls for a more sophisticated approach, where class S1 superlatives can be distinguished.

6.2 Sentiment Detection/Opinion Extraction

Like adjectives and adverbs, superlatives can be objective or subjective. Compare for example:

[28] **The Black Forest** is the ***largest*** forest in Germany. [objective]
[29] **The Black Forest** is the ***most beautiful*** area in Germany. [subjective]

So far, none of the studies in sentiment detection (e.g. Wilson et al., 2005; Pang et al., 2002) or opinion extraction (e.g. Hu and Liu, 2004; Popescu and Etzioni, 2005) have specifically looked at the role of superlatives in these areas.

Like subjective adjectives, subjective superlatives can either express positive or negative opinions. This polarity depends strongly on the adjective or adverb that the superlative is derived from.[4] As superlatives place the adjective or adverb at the highest or lowest point of the comparison scale (cf. Section 2), the question of interest is how this affects the polarity of the adjective/adverb. If the intensity of the polarity increases in a likewise manner, then subjective superlatives are bound to express the strongest or weakest opinions possible. If this hypothesis holds true, an "extreme opinion" extraction system could be created by combining the proposed superlative extraction system with a subjectivity recognition system that can identify subjective superlatives. This would clearly be of interest to many companies and market researchers.

Initial searches in Hu and Liu's annotated corpus of customer reviews (2004) look promising. Sentences in this corpus are annotated with information about positive and negative opinions, which are located on a six-point scale, where [+/-3] stand for the strongest positive/negative opinions, and [+/-1] stand for the weakest positive/negative opinions. A search for annotated sentences containing superlatives shows that an overwhelming majority are marked with strongest opinion labels.

7 Summary and Future Work

This paper proposed the task of automatically extracting useful information from superlatives oc-

[4] It may, however, also depend on whether the superlative expresses the highest ('most') or the lowest ('least') point in the scale.

curring in free text. It provided an overview of superlative constructions and the main challenges that have to be faced, described previous computational approaches and their limitations, and discussed applications in two areas in NLP: QA and Sentiment Detection/Opinion Extraction.

The proposed task can be seen as consisting of three subtasks:

TASK 1: Decide whether a given sentence contains a superlative form
TASK 2: Given a sentence containing a superlative form, identify what type of superlative it is (initially: IS-A superlative or not?)
TASK 3: For set comparisons, identify the target and the comparison set, as well as the superlative relation

Task 1 can be tackled by a simple approach relying on POS tags (e.g. JJS and RBS in the Penn Treebank tagset). For Task 2, I have carried out a thorough analysis of the different types of superlative forms and postulated a new classification for them. My present efforts are on the creation of a gold standard data set for the extraction task. As superlatives are particularly frequent in encyclopaedic language (cf. Section 3), I am considering using the Wikipedia[5] as a knowledge base. The main challenge is to devise a suitable annotation scheme which can account for all syntactic structures in which IS-A superlatives occur and which incorporates their semantic properties in an adequate way (semantic role labelling). Finally, for Task 3, I plan to use both manually created rules and machine learning techniques.

Acknowledgements

I would like to thank Bonnie Webber and Maria Milosavljevic for their helpful comments and suggestions on this paper. Many thanks also go to Nitin Jindal and Bing Liu, Johan Bos and Malvina Nissim, and Jimmy Lin and Dina Demner-Fushman for making their data available.

References

Kisuh Ahn, Johan Bos, James R. Curran, Dave Kor, Malvina Nissim and Bonnie Webber. 2005. *Question Answering with QED*. In Voorhees and Buckland (eds.): The 14th Text REtrieval Conference, TREC 2005.

Johan Bos and Malvina Nissim. 2006. An Empirical Approach to the Interpretation of Superlatives. In *Proceedings of EMNLP 2006*, pages 9-17, Sydney, Australia.

Norbert Corver and Ora Matushansky. 2006. At our best when at our boldest. Handout. TIN-dag, Feb. 4, 2006.

Irene Heim. 1999. *Notes on superlatives.* Ms., MIT.

Minqing Hu and Bing Liu. 2004. Mining Opinion Features in Customer Reviews. In *Proceedings of AAAI*, pages 755-760, San Jose, California, USA.

Rodney Huddleston and Geoffrey K. Pullum (eds.). 2002. *The Cambridge grammar of the English language*. Cambridge: Cambridge University Press.

Michael Kaisser, Silke Scheible and Bonnie Webber. 2006. Experiments at the University of Edinburgh for the TREC 2006 QA track. In *Proceedings of TREC* 2006, Gaithersburg, MD, USA.

Nitin Jindal and Bing Liu. 2006. Mining Comparative Sentences and Relations. In *Proceedings of AAAI*, Boston, MA, USA.

Jimmy Lin and Dina Demner-Fushman. 2006. Will pyramids built of nuggets topple over? In *Proceedings of the HLT/NAACL*, pages 383-390, New York, NY, USA.

Maria Milosavljevic. 1999. *The Automatic Generation of Comparisons in Descriptions of Entities*. PhD Thesis. Microsoft Research Institute, Macquarie University, Sydney, Australia.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79-86, Philadelphia, PA, USA.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proceedings of HLT/EMNLP-2005*, pages 339-346, Vancouver, British Columbia, Canada.

Majid Razmara and Leila Kosseim. 2007. A little known fact is... Answering Other questions using interest-markers. In *Proceedings of CICLing-2007*, Mexico City, Mexico.

Anna Szabolcsi. 1986. Comparative superlatives. In *MIT Working Papers in Linguistics* (8). ed. by Naoki Fukui, Tova R. Rapoport and Elisabeth Sagey. 245-265.

Theresa Wilson, Janyce Wiebe and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *Proceedings of HLT/EMNLP* 2005, pages 347-354, Vancouver, British Columbia, Canada.

[5] www.wikipedia.org

Annotating and Learning Compound Noun Semantics

Diarmuid Ó Séaghdha
University of Cambridge Computer Laboratory
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
do242@cl.cam.ac.uk

Abstract

There is little consensus on a standard experimental design for the compound interpretation task. This paper introduces well-motivated general desiderata for semantic annotation schemes, and describes such a scheme for in-context compound annotation accompanied by detailed publicly available guidelines. Classification experiments on an open-text dataset compare favourably with previously reported results and provide a solid baseline for future research.

1 Introduction

There are a number of reasons why the interpretation of noun-noun compounds has long been a topic of interest for NLP researchers. Compounds occur very frequently in English and many other languages, so they cannot be avoided by a robust semantic processing system. Compounding is a very productive process with a highly skewed type frequency spectrum, and corpus information may be very sparse. Compounds are often highly ambiguous and a large degree of "world knowledge" seems necessary to understand them. For example, knowing that a *cheese knife* is (probably) a knife for cutting cheese and (probably) not a knife made of cheese (cf. *plastic knife*) does not just require an ability to identify the senses of *cheese* and *knife* but also knowledge about what one usually does with cheese and knives. These factors combine to yield a difficult problem that exhibits many of the challenges characteristic of lexical semantic processing in general. Recent research has made significant progress on solving the problem with statistical methods and often without the need for manually created lexical resources (Lauer, 1995; Lapata and Keller, 2004; Girju, 2006; Turney, 2006). The work presented here is part of an ongoing project that treats compound interpretation as a classification problem to be solved using machine learning.

2 Selecting an Annotation Scheme

For many classification tasks, such as part-of-speech tagging or word sense disambiguation, there is general agreement on a standard set of categories that is used by most researchers. For the compound interpretation task, on the other hand, there is little agreement and numerous classification schemes have been proposed. This hinders meaningful comparison of different methods and results. One must therefore consider how an appropriate annotation scheme should be chosen.

One of the problems is that it is not immediately clear what level of granularity is desirable, or even what kind of units the categories should be. Lauer (1995) proposes a set of 8 prepositions that can be used to paraphrase compounds: a *cheese knife* is a *knife FOR cheese* but a *kitchen knife* is a *knife (used) IN a kitchen*. An advantage of this approach is that preposition-noun co-occurrences can efficiently be mined from large corpora using shallow techniques. On the other hand, interpreting a paraphrase requires further disambiguation as one preposition can map onto many semantic relations.[1] Girju et al. (2005) and Nastase and Szpakowicz (2003) both present large inventories of seman-

[1] The interpretation of prepositions is itself the focus of a Semeval task in 2007.

Proceedings of the ACL 2007 Student Research Workshop, pages 73–78,
Prague, June 2007.

tic relations that describe noun-noun dependencies. Such relations provide richer semantic information, but it is harder for both humans and machines to identify their occurrence in text. Larger inventories can also suffer from class sparsity; for example, 14 of Girju et al.'s 35 relations do not occur in their dataset and 7 more occur in less than 1% of the data. Nastase and Szpakowicz' scheme mitigates this problem by the presence of 5 supercategories.

Each of these proposals has its own advantages and drawbacks, and there is a need for principled criteria for choosing one. As the literature on semantic annotation "best practice" is rather small,[2] I devised a novel set of design principles based on empirical and theoretical considerations:

1. The inventory of informative categories should account for as many compounds as possible
2. The category boundaries should be clear and categories should describe a coherent concept
3. The class distribution should not be overly skewed or sparse
4. The concepts underlying the categories should generalise to other linguistic phenomena
5. The guidelines should make the annotation process as simple as possible
6. The categories should provide useful semantic information

These intuitively appear to be desirable principles for any semantic annotation scheme. The requirement of class distribution balance is motivated by the classification task. Where one category dominates, the most-frequent-class baseline can be difficult to exceed and care must be taken in evaluation to consider macro-averaged performance as well as raw accuracy. It has been suggested that classifiers trained on skewed data may perform poorly on minority classes (Zhang and Oles, 2001). Of course, this is not a justification for conflating concepts with little in common, and it may well be that the natural distribution of data is inherently skewed.

There is clearly a tension between these criteria, and only a best-fit solution is possible. However, it was felt that a new scheme might satisfy them more optimally than existing schemes. Such a proposal necessitates a method of evaluation. Not all the criteria are easily evaluable. It is difficult to prove generalisability and usefulness conclusively, but it can be maximised by building on more general work on semantic representation; for example, the guidelines introduced here use a conception of events and participants compatible with that of FrameNet (Baker et al., 1998). Good results on agreement and baseline classification will provide positive evidence for the coherence and balance of the classes; agreement measures can confirm ease of annotation.

Relation	Distribution	Example
BE	191 (9.55%)	*steel knife*
HAVE	199 (9.95%)	*street name*
IN	308 (15.40%)	*forest hut*
INST	266 (13.30%)	*rice cooker*
ACTOR	236 (11.80%)	*honey bee*
ABOUT	243 (12.15%)	*fairy tale*
REL	81 (4.05%)	*camera gear*
LEX	35 (1.75%)	*home secretary*
UNKNOWN	9 (0.45%)	*simularity crystal*
MISTAG	220 (11.00%)	*blazing fire*
NONCOMP	212 (10.60%)	*[real tennis] club*

Table 1: Sample class frequencies

In choosing an appropriate level of granularity, I wished to avoid positing a large number of detailed but rare categories. Levi's (1978) set of nine semantic relations was used as a starting point. The development process involved a series of revisions over six months, aimed at satisfying the six criteria above and maximising interannotator agreement in annotation trials. The nature of the decisions which had to be made is exemplified by the compound *car factory*, whose standard referent seems to qualify as FOR, CAUSE, FROM and IN in Levi's scheme (and causes similar problems for the other schemes I am aware of). Likewise there seems to be no principled way to choose between a locative or purposive label for *dining room*. Such examples led to both redefinition of category boundaries and changes in the category set; for example, FOR was replaced by INST and AGENT, which are independent of purposivity. This resulted in the class inventory shown in Table 1 and a detailed set of annotation guidelines.[3]

[2] One relevant work is Wilson and Thomas (1997).

[3] The guidelines are publicly available at `http://www.cl.cam.ac.uk/~do242/guidelines.pdf`.

The scheme's development is described at length in Ó Séaghdha (2007b).

Many of the labels are self-explanatory. AGENT and INST(rument) apply to sentient and non-sentient participants in an event respectively, with ties (e.g., *stamp collector*) being broken by a hierarchy of coarse semantic roles. REL is an OTHER-style category for compounds encoding non-specific association. LEX(icalised) applies to compounds which are semantically opaque without prior knowledge of their meanings. MISTAG and NONCOMP(ound) labels are required to deal with sequences that are not valid two-noun compounds but have been identified as such due to tagging errors and the simple data extraction heuristic described in Section 3.1. Coverage is good, as 92% of valid compounds in the dataset described below were assigned one of the six main semantic relations.

3 Annotation Experiment

3.1 Data

A simple heuristic was used to extract noun sequences from the 90 million word written part of the British National Corpus.[4] The corpus was parsed using the RASP parser[5] and all sequences of two common nouns were extracted except those adjacent to another noun and those containing non-alphabetic characters. This yielded almost 1.6 million tokens with 430,555 types. 2,000 unique tokens were randomly drawn for use in annotation and classification experiments.

3.2 Method

Two annotators were used: the current author and an annotator experienced in lexicography but without any special knowledge of compounds or any role in the development of the annotation scheme. In all the trials described here, each compound was presented alongside the sentence in which it was found in the BNC. The annotators had to assign one of the labels in Table 1 and the rule that licensed that label in the annotation guidelines. For example, the compound *forest hut* in its usual sense would be annotated `IN,2,2.1.3.1` to indicate the semantic relation, the direction of the relation (it is a *hut in a forest*, not a *forest in a hut*) and that the label is licensed by rule 2.1.3.1 in the guidelines (*N1/N2 is an object spatially located in or near N2/N1*).[6] Two trial batches of 100 compounds were annotated to familiarise the second annotator with the guidelines and to confirm that the guidelines were indeed usable for others. The first trial resulted in agreement of 52% and the second in agreement of 73%. The result of the second trial, corresponding to a Kappa beyond-chance agreement estimate (Cohen, 1960) of $\hat{\kappa} = 0.693$, was very impressive and it was decided to proceed to a larger-scale task. 500 compounds not used in the trial runs were drawn from the 2,000-item set and annotated.

3.3 Results and Analysis

Agreement on the test set was 66.2% with $\hat{\kappa} = 0.62$. This is less than the score achieved in the second trial run, but may be a more accurate estimator of the true population κ due to the larger sample size. On the other hand, the larger dataset may have caused annotator fatigue. Pearson standardised residuals (Haberman, 1973) were calculated to identify the main sources of disagreement.[7] In the context of inter-annotator agreement one expects these residuals to have large positive values on the agreement diagonal and negative values in all other cells. Among the six main relations listed at the top of Table 1, a small positive association was observed between INST and ABOUT, indicating that borderline topics such as *assessment task* and *gas alarm* were likely to be annotated as INST by the first annotator and ABOUT by the second. It seems that the guidelines might need to clarify this category boundary.

It is clear from analysis of the data that the REL, LEX and UNKNOWN categories show very low agreement. They all have low residuals on the agreement diagonal (that for UNKNOWN is negative) and numerous positive entries off it. REL and LEX are also the categories for which it is most difficult to

[4] `http://www.natcorp.ox.ac.uk/`

[5] `http://www.informatics.susx.ac.uk/research/nlp/rasp/`

[6] The additional information provided by the direction and rule annotations could be used to give a richer classification scheme but has not yet been used in this way in my experiments.

[7] The standardised residual of cell ij is calculated as

$$e_{ij} = \frac{n_{ij} - \hat{p}_{i+}\hat{p}_{+j}}{\sqrt{\hat{p}_{i+}\hat{p}_{+j}(1-\hat{p}_{i+})(1-\hat{p}_{+j})}}$$

where n_{ij} is the observed value of cell ij and $\hat{p}_{i+}$, $\hat{p}_{+j}$ are row and column marginal probabilities estimated from the data.

provide clear guidelines. On the other hand, the MISTAG and NONCOMP categories showed good agreement, with slightly higher agreement residuals than the other categories. To get a rough idea of agreement on the six categories used in the classification experiments described below, agreement was calculated for all items which neither annotator annotated with any of REL, LEX, UNKNOWN, MISTAG and NONCOMP. This left 343 items with agreement of 73.6% and $\hat{\kappa} = 0.683$.

3.4 Discussion

This is the first work I am aware of where compounds were annotated in their sentential context. This aspect is significant, as compound meaning is often context dependent (compare *school management decided...* and *principles of school management*) and in-context interpretation is closer to the dynamic of real-world language use. Context can both help and hinder agreement, and it is not clear whether in- or out-of-context annotation is easier.

Previous work has given out-of-context agreement figures for corpus data. Kim and Baldwin (2005) report an experiment using 2,169 compounds taken from newspaper text and the categories of Nastase and Szpakowicz (2003). Their annotators could assign multiple labels in case of doubt and were judged to agree on an item if their annotations had any label in common. This less stringent measure yielded agreement of 52.31%. Girju et al. (2005) report agreement for annotation using both Lauer's 8 prepositional labels ($\hat{\kappa} = 0.8$) and their own 35 semantic relations ($\hat{\kappa} = 0.58$). These figures are difficult to interpret as annotators were again allowed assign multiple labels (for the prepositions this occurred in "almost all" cases) and the multiply-labelled items were excluded from the calculation of Kappa. This entails discarding the items which are hardest to classify and thus most likely to cause disagreement.

Girju (2006) has recently published impressive agreement results on a related task. This involved annotating 2,200 compounds extracted from an online dictionary, each presented in five languages, and resulted in a Kappa score of 0.67. This task may have been facilitated by the data source and its multilingual nature. It seems plausible that dictionary entries are more likely to refer to familiar concepts than compounds extracted from a balanced corpus, which are frequently context-dependent coinages or rare specialist terms. Furthermore, the translations of compounds in Romance languages often provide information that disambiguates the compound meaning (this aspect was the main motivation for the work) and translations from a dictionary are likely to correspond to an item's most frequent meaning. A qualitative analysis of the experiment described above suggests that about 30% of the disagreements can confidently be attributed to disagreement about the semantics of a given compound (as opposed to how a given meaning should be annotated).[8]

4 SVM Learning with Co-occurrence Data

4.1 Method

The data used for classification was taken from the 2,000 items used for the annotation experiment, annotated by a single annotator. Due to time constraints, this annotation was done before the second annotator had been used and was not changed afterwards. All compounds annotated as BE, HAVE, IN, INST, AGENT and ABOUT were used, giving a dataset of 1,443 items. All experiments were run using Support Vector Machine classifiers implemented in LIBSVM.[9] Performance was measured via 5-fold cross-validation. Best performance was achieved with a linear kernel and one-against-all classification. The single SVM parameter C was estimated for each fold by cross-validating on the training set. Due to the efficiency of the linear kernel the optimisation, training and testing steps for each fold could be performed in under an hour.

I investigated what level of performance could be achieved using only corpus information. Feature vectors were extracted from the written BNC for each modifier and head in the dataset under the following conditions:

w5, w10: Each word within a window of 5 or 10 words on either side of the item is a feature.
Rbasic, Rmod, Rverb, Rconj: These feature sets

[8] For example, one annotator thought *peat boy* referred to a *boy who sells peat* (AGENT) while the other thought it referred to a *boy buried in peat* (IN).

[9] `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

use the grammatical relation output of the RASP parser run over the written BNC. The *Rbasic* feature set conflates information about 25 grammatical relations; *Rmod* counts only prepositional, nominal and adjectival noun modification; *Rverb* counts only relations among subjects, objects and verbs; *Rconj* counts only conjunctions of nouns. In each case, each word entering into one of the target relations with the item is a feature and only the target relations contribute to the feature values.

Each feature vector counts the target word's co-occurrences with the 10,000 words that most frequently appear in the context of interest over the entire corpus. Each compound in the dataset is represented by the concatenation of the feature vectors for its head and modifier. To model aspects of co-occurrence association that might be obscured by raw frequency, the log-likelihood ratio G^2 was used to transform the feature space.[10]

4.2 Results and Analysis

Results for these feature sets are given in Table 2. The simple word-counting conditions *w5* and *w10* perform relatively well, but the highest accuracy is achieved by *Rconj*. The general effect of the log-likelihood transformation cannot be stated categorically, as it causes some conditions to improve and others to worsen, but the G^2-transformed *Rconj* features give the best results of all with 54.95% accuracy (53.42% macro-average). Analysis of performance across categories shows that in all cases accuracy is lower (usually below 30%) on the BE and HAVE relations than on the others (often above 50%). These two relations are least common in the dataset, which is why the macro-averaged figures are slightly lower than the micro-averaged accuracy.

4.3 Discussion

It is interesting that the conjunction-based features give the best performance, as these features are also the most sparse. This may be explained by the fact that words appearing in conjunctions are often taxonomically similar (Roark and Charniak, 1998) and that taxonomic information is particularly useful for compound interpretation, as evidenced by the success of WordNet-based methods (see Section 5).

[10]This measure is relatively robust where frequency counts are low and consistently outperformed other association measures in the empirical evaluation of Evert (2004).

	Raw		G^2	
	Accuracy	Macro	Accuracy	Macro
w5	52.60%	51.07%	51.35%	49.93%
w10	51.84%	50.32%	50.10%	48.60%
Rbasic	51.28%	49.92%	51.83%	50.26%
Rmod	51.35%	50.06%	48.51%	47.03%
Rverb	48.79%	47.13%	48.58%	47.07%
Rconj	**54.12%**	**52.44%**	**54.95%**	**53.42%**

Table 2: Performance of BNC co-occurrence data

In comparing reported classification results, it is difficult to disentangle the effects of different data, annotation schemes and classification methods. The results described here should above all be taken to demonstrate the feasibility of learning using a well-motivated annotation scheme and to provide a baseline for future work on the same data. In terms of methodology, Turney's (2006) Vector Space Model experiments are most similar. Using feature vectors derived from lexical patterns and frequencies returned by a Web search engine, a nearest-neighbour classifier achieves 45.7% accuracy on compounds annotated with 5 semantic classes. Turney improves accuracy to 58% with a combination of query expansion and linear dimensionality reduction. This method trades off efficiency for accuracy, requiring many times more resources in terms of time, storage and corpus size than that described here. Lapata and Keller (2004) obtain accuracy of 55.71% on Lauer's (1995) prepositionally annotated data using simple search engine queries. Their method has the advantage of not requiring supervision, but it cannot be used with deep semantic relations.

5 SVM Classification with WordNet

5.1 Method

The experiments reported in this section make a basic use of the WordNet[11] hierarchy. Binary feature vectors are used whereby a vector entry is 1 if the item belongs to or is a hyponym of the synset corresponding to that feature, and 0 otherwise. Each compound is represented by the concatenation of two such vectors, for the head and modifier. The same

[11]`http://wordnet.princeton.edu/`

classification method is used as in Section 4.

5.2 Results and Discussion

This method achieves accuracy of 56.76% and macro-averaged accuracy of 54.59%, slightly higher than that achieved by the co-occurrence features. Combining WordNet and co-occurrence vectors by simply concatenating the G^2-transformed *Rconj* vector and WordNet feature vector for each compound gives a further boost to 58.35% accuracy (56.70% macro-average).

These results are higher than those reported for similar approaches on open-text data (Kim and Baldwin, 2005; Girju et al., 2005), though the same caveat applies about comparison. The best results (over 70%) reported so far for compound interpretation use a combination of multiple lexical resources and detailed additional annotation (Girju et al., 2005; Girju, 2006).

6 Conclusion and Future Directions

The annotation scheme described above has been tested on a rigorous multiple-annotator task and achieved superior agreement to comparable results in the literature. Further refinement should be possible but would most likely yield diminishing returns. In the classification experiments, my goal was to see what level of performance could be gained by using straightforward techniques so as to provide a meaningful baseline for future research. Good results were achieved with methods that rely neither on massive corpora or broad-coverage lexical resources, though slightly better performance was achieved using WordNet. An advantage of resource-poor methods is that they can be used for the many languages where compounding is common but such resources are limited.

The learning approach described here only captures the lexical semantics of the individual consituents. It seems intuitive that other kinds of corpus information would be useful; in particular, contexts in which the head and modifier of a compound both occur may make explicit the relations that typically hold between their referents. Kernel methods for using such relational information are investigated in Ó Séaghdha (2007a) with promising results, and I am continuing my research in this area.

References

Collin Baker, Charles Fillmore, and John Lowe. 1998. The Berkeley FrameNet project. In *Proc. ACL-COLING-98*, pages 86–90, Montreal, Canada.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20:37–46.

Stefan Evert. 2004. *The Statistics of Word Cooccurrences: Word Pairs and Collocations*. Ph.D. thesis, Universität Stuttgart.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19(4):479–496.

Roxana Girju. 2006. Out-of-context noun phrase semantic interpretation with cross-linguistic evidence. In *Proc. CIKM-06*, pages 268–276, Arlington, VA.

Shelby J. Haberman. 1973. The analysis of residuals in cross-classified tables. *Biometrics*, 29(1):205–220.

Su Nam Kim and Timothy Baldwin. 2005. Automatic interpretation of noun compounds using WordNet similarity. In *Proc. IJCNLP-05*, pages 945–956, Jeju Island, Korea.

Mirella Lapata and Frank Keller. 2004. The Web as a baseline: Evaluating the performance of unsupervised Web-based models for a range of NLP tasks. In *Proc. HLT-NAACL-04*, pages 121–128, Boston, MA.

Mark Lauer. 1995. *Designing Statistical Language Learners: Experiments on Compound Nouns*. Ph.D. thesis, Macquarie University.

Judith N. Levi. 1978. *The Syntax and Semantics of Complex Nominals*. Academic Press, New York.

Vivi Nastase and Stan Szpakowicz. 2003. Exploring noun-modifier semantic relations. In *Proc. IWCS-5*, Tilburg, Netherlands.

Brian Roark and Eugene Charniak. 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proc. ACL-COLING-98*, pages 1110–1106, Montreal, Canada.

Diarmuid Ó Séaghdha. 2007a. Co-occurrence contexts for corpus-based noun compound interpretation. In *Proc. of the ACL Workshop A Broader Perspective on Multiword Expressions*, Prague, Czech Republic.

Diarmuid Ó Séaghdha. 2007b. Designing and evaluating a semantic annotation scheme for compound nouns. In *Proc. Corpus Linguistics 2007*, Birmingham, UK.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

Andrew Wilson and Jenny Thomas. 1997. Semantic annotation. In R. Garside, G. Leech, and A. McEnery, editors, *Corpus Annotation*. Longman, London.

Tong Zhang and Frank J. Oles. 2001. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4(1):5–31.

Semantic Classification of Noun Phrases Using Web Counts and Learning Algorithms

Paul Nulty
School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
paul.nulty@ucd.ie

Abstract

This paper investigates the use of machine learning algorithms to label modifier-noun compounds with a semantic relation. The attributes used as input to the learning algorithms are the web frequencies for phrases containing the modifier, noun, and a prepositional joining term. We compare and evaluate different algorithms and different joining phrases on Nastase and Szpakowicz's (2003) dataset of 600 modifier-noun compounds. We find that by using a Support Vector Machine classifier we can obtain better performance on this dataset than a current state-of-the-art system; even with a relatively small set of prepositional joining terms.

1 Introduction

Noun-modifier word pairs occur frequently in many languages, and the problem of semantic disambiguation of these phrases has many potential applications in areas such as question-answering and machine translation. One very common approach to this problem is to define a set of semantic relations which capture the interaction between the modifier and the head noun, and then attempt to assign one of these semantic relations to each noun-modifier pair. For example, the phrase "*flu virus*" could be assigned the semantic relation "*causal*" (the virus causes the flu); the relation for "*desert storm*" could be "*location*" (the storm is located in the desert).

There is no consensus as to which set of semantic relations best captures the differences in meaning of various noun phrases. Work in theoretical linguistics has suggested that noun-noun compounds may be formed by the deletion of a predicate verb or preposition (Levi 1978). However, whether the set of possible predicates numbers 5 or 50, there are likely to be some examples of noun phrases that fit into none of the categories and some that fit in multiple categories.

Modifier-noun phrases are often used interchangeably with paraphrases which contain the modifier and the noun joined by a preposition or simple verb. For example, the query "*morning exercise*" returns 133,000 results from the Yahoo search engine, and a query for the phrase "*exercise in the morning*" returns 47,500 results. Sometimes people choose to use a modifier-noun compound phrase to describe a concept, and sometimes they choose to use a paraphrase which includes a preposition or simple verb joining head noun and the modifier. One method for deducing semantic relations between words in compounds involves gathering n-gram frequencies of these paraphrases, containing a noun, a modifier and a "joining term" that links them. Some algorithm can then be used to map from joining term frequencies to semantic relations and so find the correct relation for the compound in question. This is the approach we use in our experiments. We choose two sets of joining terms, based on the frequency with which they occur in between nouns in the British National Cor-

Proceedings of the ACL 2007 Student Research Workshop, pages 79–84,
Prague, June 2007.

pus (BNC). We experiment with three different learning algorithms; Nearest Neighbor, Multi-Layer Perceptron and Support Vector Machines (SVM).

2 Motivation

The motivation for this paper is to discover which joining terms are good predictors of a semantic relation, and which learning algorithms perform best at the task of mapping from joining terms to semantic relations for modifier-noun compounds.

2.1 Joining Terms

Choosing a set of joining terms in a principled manner in the hope of capturing the semantic relation between constituents in the noun phrase is difficult, but there is certainly some correlation between a prepositional term or short linking verb and a semantic relation. For example, the preposition "*during*" indicates a temporal relation, while the preposition "*in*" indicates a locative relation, either temporal or spatial.

In this paper, we are interested in whether the frequency with which a joining term occurs between two nouns is related to how it indicates a semantic interaction. This is in part motivated by Zipf's theory which states that the more frequently a word occurs in a corpus the more meanings or senses it is likely to have (Zipf 1929). If this is true, we would expect that very frequent prepositions, such as "*of*", would have many possible meanings and therefore not reliably predict a semantic relation. However, less frequent prepositions, such as *"while"* would have a more limited set of senses and therefore accurately predict a semantic relation.

2.2 Machine Learning Algorithms

We are also interested in comparing the performance of machine learning algorithms on the task of mapping from n-gram frequencies of joining terms to semantic relations. For the experiments we use Weka, (Witten and Frank, 1999) a machine learning toolkit which allows for fast experimentation with many standard learning algorithms. In Section 5 we present the results obtained using the nearest-neighbor, neural network (i.e. multi-layer perceptron) and SVM. The mechanisms of these different learning approaches will be discussed briefly in Section 4.

3 Related Work

3.1 Web Mining

Much of the recent work conducted on the problem of assigning semantic relations to noun phrases has used the web as a corpus. The use of hit counts from web search engines to obtain lexical information was introduced by Turney (2001). The idea of searching a large corpus for specific lexico-syntactic phrases to indicate a semantic relation of interest was first described by Hearst (1992).

A lexical pattern specific enough to indicate a particular semantic relation is usually not very frequent, and using the web as a corpus alleviates the data sparseness problem. However, it also introduces some problems.

- The query language permitted by the large search engines is somewhat limited.
- Two of the major search engines (Google and Yahoo) do not provide exact frequencies, but give rounded estimates instead.
- The number of results returned is unstable as new pages are created and deleted all the time.

Nakov and Hearst (2005) examined the use of web-based n-gram frequencies for an NLP task and concluded that these issues do not greatly impact the interpretation of the results. Keller and Lapata (2003) showed that web frequencies correlate reliably with standard corpus frequencies.

Lauer (1995) tackles the problem of semantically disambiguating noun phrases by trying to find the preposition which best describes the relation between the modifier and head noun. His method involves searching a corpus for occurrences paraphrases of the form "*noun preposition modifier*". Whichever preposition is most frequent in this context is chosen. Lapata and Keller (2005) improved on Lauer's results at the same task by using the web as a corpus. Nakov and Hearst (2006) use queries of the form "*noun that * modifier*" where '*' is a wildcard operator. By retrieving the words that most commonly occurred in the place of the wildcard they were able to identify very specific predicates that are likely to represent the relation between noun and modifier.

3.2 Machine Learning Approaches

There have been two main approaches used when applying machine learning algorithms to the semantic disambiguation of modifier-noun phrases.

The first approach is to use semantic properties of the noun and modifier words as attributes, using a lexical hierarchy to extract these properties. This approach was used by Rosario and Hearst (2001) within a specific domain – medical texts. Using an ontology of medical terms they train a neural network to semantically classify nominal phrases, achieving 60% accuracy over 16 classes.

Nastase and Szpakowicz (2003) use the position of the noun and modifier words within general semantic hierarchies (Roget's Thesaurus and WordNet) as attributes for their learning algorithms. They experiment with various algorithms and conclude that a rule induction system is capable of generalizing to characterize the noun phrases.

Moldovan et al (2004) also use WordNet. They experiment with a Bayesian algorithm, decision trees, and their own algorithm; semantic scattering. There are some drawbacks to the technique of using semantic properties extracted from a lexical hierarchy. Firstly, it has been noted that the distinctions between word senses in WordNet are very fine-grained, making the task of word-sense disambiguation tricky. Secondly, it is usual to use a rule-based learning algorithm when the attributes are properties of the words rather than n-gram frequency counts. As Nastase and Szpakowicz (2003) point out, a large amount of labeled data is required to allow these rule-based learners to effectively generalize, and manually labeling thousands of modifier-noun compounds would be a time-consuming task.

causal	flu virus, onion tear
temporal	summer travel, morning class
spatial	west coast, home remedy
participant	mail sorter, blood donor
quality	rice paper, picture book

Table 1**:** Examples for each of the five relations

The second approach is to use statistical information about the occurrence of the noun and modifier in a corpus to generate attributes for a machine learning algorithm. This is the method we will describe in this paper. Turney and Littman (2005) use a set of 64 short prepositional and conjunctive phrases they call "joining terms" to generate exact queries for AltaVista of the form "*noun joining term modifier*", and "*modifier joining term noun*". These hit counts were used with a nearest neighbor algorithm to assign the noun phrases semantic relations. Over the set of 5 semantic relations defined by Nastase and Szpakowicz (2003), they achieve an accuracy of 45.7% for the task of assigning one of 5 semantic relations to each of the 600 modifier-noun phrases.

4 Method

The method described in this paper is similar to the work presented in Turney and Littman (2005). We collect web frequencies for queries of the form "*head joining term modifier*". We did not collect queries of the form "*modifier joining term head*"; in the majority of paraphrases of noun phrases the head noun occurs before the modifying word. As well as trying to achieve reasonable accuracy, we were interested in discovering what kinds of joining phrases are most useful when trying to predict the semantic relation, and which machine learning algorithms perform best at the task of using vectors of web-based n-gram frequencies to predict the semantic relation.

For our experiments we used the set of 600 labeled noun-modifier pairs of Nastase and Szpakowicz (2003). This data was also used by Turney and Littman (2005). Of the 600 modifier-noun phrases, three contained hyphenated or two-word modifier terms, for example "*test-tube baby*". We omitted these three examples from our experiments, leaving a dataset of 597 examples.

The data is labeled with two different sets of semantic relations: one set of 30 relations with fairly specific meanings, and another set of 5 relations with more abstract meanings. For our experiments we focused on the set of 5 relations. One reason for this is that dividing a set of 600 instances into 30 classes results in a fairly sparse and uneven dataset. Table 1 is a list of the relations used and examples of compounds that are labeled with each relation.

4.1 Collecting Web Frequencies

In order to collect the n-gram frequencies, we used the Yahoo Search API. Collecting frequencies for

600 noun-modifier pairs, using 28 different joining terms required 16,800 calls to the search engine. We will discuss our choice of the joining terms in the next section.

When collecting web frequencies we took advantage of the OR operator provided by the search engine. For each joining term, we wanted to sum the number of hits for the term on its own, the term followed by '*a*' and the term followed by '*the*'. Instead of conducting separate queries for each of these forms, we were able to sum the results with just one search. For example, if the noun phrase was "student invention" and the joining phrase was "by"; one of the queries would be:

"invention by student" OR "invention by a student" OR "invention by the student"

This returns the sum of the number of pages matched by each of these three exact queries. The idea is that these sensible paraphrases will return more hits than nonsense ones, such as:

"invention has student" OR "invention has a student" OR "invention has the student"

It would be possible to construct a set of hand-coded rules to map from joining terms to semantic relations; for example "during" maps to temporal, "by" maps to causal and so on. However, we hope that the classifiers will be able to identify combinations of prepositions that indicate a relation.

4.2 Choosing a Set of Joining Terms

Possibly the most difficult problem with this method is deciding on a set of joining terms which is likely to provide enough information about the noun-modifier pairs to allow a learning algorithm to predict the semantic relation. Turney and Littman (2005) use a large and varied set of joining terms. They include the most common prepositions, conjunctions and simple verbs like "*has*", "*goes"* and "*is*". Also, they include the wildcard operator '*' in many of their queries; for example "*not*", "* *not*" and "*but not*" are all separate queries. In addition, they include prepositions both with and without the definite article as separate queries, for example "*for*" and "*for the*".

The joining terms used for the experiments in this paper were chosen by examining which phrases most commonly occurred between two nouns in the BNC. We counted the frequencies with which phrases occurred between two nouns and chose the 28 most frequent of these phrases as our joining terms. We excluded conjunctions and determiners from the list of the most frequent joining terms. We excluded conjunctions on the basis that in most contexts a conjunction merely links the two nouns together for syntactic purposes; there is no real sense in which one of the nouns modifies another semantically in this context. We excluded determiners on the basis that the presence of a determiner does not affect the semantic properties of the interaction between the head and modifier.

4.3 Learning Algorithms

There were three conditions experimented with using three different algorithms. For the first condition, the attributes used by the learning algorithms consisted of vectors of web hits obtained using the 14 most frequent joining terms found in the BNC. The next condition used a vector of web hits obtained using the joining terms that occurred from position 14 to 28 in the list of the most frequent terms found in the BNC. The third condition used all 28 joining terms. The joining terms are listed in Table 2. We used the log of the web counts returned, as recommended in previous work (Keller and Lapata, 2003).

1-14	**15-28**
of	against
in	within
to	during
for	through
on	over
with	towards
at	without
is	across
from	because
as	behind
by	after
between	before
about	while
has	under

Table 2: Joining terms ordered by the frequency with which they occurred between two nouns in the BNC.

The first learning algorithm we experimented with was the nearest neighbor algorithm 'IB1', as

implemented in Weka. This algorithm considers the vector of n-gram frequencies as a multi-dimensional space, and chooses the label of the nearest example in this space as the label for each new example. Testing for this algorithm was done using leave-one-out cross validation.

The next learning algorithm we used was the multi-layer perceptron, or neural network. The network was trained using the backpropagation of error technique implemented in Weka. For the first two sets of data we used a network with 14 input nodes, one hidden layer with 28 nodes, and 5 output nodes. For the final condition, which uses the frequencies for all 28 joining terms, we used 28 input nodes, one hidden layer with 56 nodes, and again 5 outputs, one for each class. We used 20-fold cross validation with this algorithm.

The final algorithm we tested was an SVM trained with the Sequential Minimal Optimization method provided by Weka. A support vector machine is a method for creating a classification function which works by trying to find a hypersurface in the space of possible inputs that splits the positive examples from the negative examples for each class. For this test we again used 20-fold cross validation.

5. Results

The accuracy of the algorithms on each of the conditions is illustrated below in Table 3. Since the largest class in the dataset accounts for 43% of the examples, the baseline accuracy for the task (guessing "participant" all the time) is 43%.

The condition containing the counts for the less frequent joining terms performed slightly better than that containing the more frequent ones, but the best accuracy resulted from using all 28 frequencies. The Multi-Layer Perceptron performed better than the nearest neighbor algorithm on all three conditions. There was almost no difference in accuracy between the first two conditions, and again using all of the joining terms produced the best results.

The SVM algorithm produced the best accuracy of all, achieving 50.1% accuracy using the combined set of joining terms. The less frequent joining terms achieve slightly better accuracy using the Nearest Neighbor and SVM algorithms, and very slightly worse accuracy using the neural network. Using all of the joining terms resulted in a significant improvement in accuracy for all algorithms. The SVM consistently outperformed the baseline; neither of the other algorithms did so.

6. Discussion and Future Work

Our motivation in this paper was twofold. Firstly, we wanted to compare the performance of different machine learning algorithms on the task of mapping from a vector of web frequencies of paraphrases containing joining terms to semantic relations. Secondly, we wanted to discover whether the frequency of joining terms was related to their effectiveness at predicting a semantic relation.

6.1 Learning Algorithms

The results suggest that the nearest neighbor approach is not the most effective algorithm for the classification task. Turney and Littman (2005) achieve an accuracy of 45.7%, where we achieve a maximum accuracy of 38.1% on this dataset using a nearest neighbor algorithm. However, their technique uses the cosine of the angle between the vectors of web counts as the similarity metric, while the nearest neighbor implementation in Weka uses the Euclidean distance.

Also, they use 64 joining terms and gather counts for both the forms "noun joining term modifier" and "modifier joining term noun" (128 frequencies in total); while we use only the former construction with 28 joining terms. By using the SVM classifier, we were able to achieve a higher accuracy than Turney and Littman (50.1% versus 45.7%) with significantly fewer joining terms (28 versus 128). However, one issue with the SVM is

	Joining Terms 1-14	Joining terms 15-28	All 28 Joining terms
Nearest Neighbor	32.6	34.7	38.1
Multi Layer Perceptron	37.6	37.4	42.2
Support Vector Machine	44.2	45.9	50.1

Table 3**:** Accuracy for each algorithm using each set of joining terms on the Nastase and Szpakowicz test set of modifier-noun compounds.

that it never predicted the class "causal" for any of the examples. The largest class in our dataset is "participant", which is the label for 43% of the examples; the smallest is "temporal", which labels 9% of the examples. "Causal" labels 14% of the data. It is difficult to explain why the algorithm fails to account for the "causal" class; a useful task for future work would be to conduct a similar experiment with a more balanced dataset.

6.2 Joining Terms

The difference in accuracy achieved by the two sets of joining terms is quite small, although for two of the algorithms the less frequent terms did achieve slightly better results. The difficulty is that the task of deducing a semantic relation from a paraphrase such as "storm in the desert" requires many different types of information. It requires knowledge about the preposition "in"; i.e. that it indicates a location. It requires knowledge about the noun "desert", i.e. that it is a location in space rather than time, and it requires the knowledge that a "storm" may refer both to an event in time and an entity in space. It may be that a combination of semantic information from an ontology and statistical information about paraphrases could be used together to achieve better performance on this task.

Another interesting avenue for future work in this area is investigation into exactly how "joining terms" relate to semantic relations. Given Zipf's observation that high frequency words are more ambiguous than low frequency words, it is possible that there is a relationship between the frequency of the preposition in a paraphrase such as *"storm in the desert"* and the ease of understanding that phrase. For example, the preposition *'of'* is very frequent and could be interpreted in many ways. Therefore, the *'of'* may be used in phrases where the semantic relation can be easily deduced from the nominals in the phrase alone. Less common (and therefore more informative) prepositions such as '*after'* or '*because*' may be used more often in phrases where the nominals alone do not contain enough information to deduce the relation, or the relation intended is not the most obvious one given the two nouns.

References

Marti A. Hearst. 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. *COLING 92:* (2) pp. 539-545*,* Nantes, France,

Frank Keller and Mirella Lapata. 2003. Using the Web to Obtain Frequencies for Unseen Bigrams. *Computational Linguistics,* 29: pp 459-484.

Mirella Lapata and Frank Keller. 2005. Web-Based Models for Natural Language Processing. *ACM Transactions on Speech and Language Processing* 2:1, pp 1-31.

Mark Lauer. 1995. *Designing Statistical Language Learners: Experiments on Noun Compounds*. PhD thesis, Macquarie University, NSW 2109, Australia.

Judith Levi. 1978. *The Syntax and Semantics of Complex Nominals,* Academic Press, New York, NY.

Dan Moldovan, Adriana Badulescu, Marta Tatu, Daniel Antohe and Roxana Girju. 2004. Models for the Semantic Classification of Noun Phrases. In *Proceedings of the HLT/NAACL Workshop on Computational Lexical Semantics.* pp 60-67 Boston , MA.

Preslav Nakov and Marti Hearst. 2006. Using Verbs to Characterize Noun-Noun Relations. In *Proceedings of AIMSA 2006,* pp 233-244, Varne, Bulgaria.

Preslav Nakov and Marti Hearst. 2005. Using the Web as an Implicit Training Set: Application to Structural Ambiguity Resolution. *In Proceedings of HLT/EMNLP'05.* pp 835-842, Vancouver, Canada.

Vivi Nastase and Stan Szpakowicz. 2003. *Exploring Noun-Modifier Semantic Relations.* In *Fifth International Workshop on Computational Semantics*, pp 285-301. Tillburg, Netherlands.

Barbara Rosario and Marti A. Hearst. 2001. Classifying the semantic relations in noun compounds via a domain-specific lexical hierarchy. In *Proceedings of EMNLP 2001*, pp 82-90, Pittsburgh, PA, USA.

Peter D. Turney. 2001. Mining the web for synonyms: PM-IR vs LSA on TOEFL. *Proceedings of ECML'01.* pp 491-502. Freiburg, Germany.

Peter D. Turney and Michael L. Littman. 2005. Corpus-based learning of analogies and semantic relations. *Machine Learning,* 60(1–3):251–278.

Ian H. Witten and Eibe Frank. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann.

George K. Zipf. 1932. *Selected Studies of the Principle of Relative Frequency in Language*. Cambridge, MA.

Computing Lexical Chains with Graph Clustering

Olena Medelyan
Computer Science Department
The University of Waikato
New Zealand
olena@cs.waikato.ac.nz

Abstract

This paper describes a new method for computing lexical chains. These are sequences of semantically related words that reflect a text's cohesive structure. In contrast to previous methods, we are able to select chains based on their cohesive strength. This is achieved by analyzing the connectivity in graphs representing the lexical chains. We show that the generated chains significantly improve performance of automatic text summarization and keyphrase indexing.

1 Introduction

Text understanding tasks such as topic detection, automatic summarization, discourse analysis and question answering require deep understanding of the text's meaning. The first step in determining this meaning is the analysis of the text's concepts and their inter-relations. Lexical chains provide a framework for such an analysis. They combine semantically related words across sentences into meaningful sequences that reflect the cohesive structure of the text.

Lexical chains, introduced by Morris and Hirst (1991), have been studied extensively in the last decade, since large lexical databases are available in digital form. Most approaches use WordNet or Roget's thesaurus for computing the chains and apply the results for text summarization.

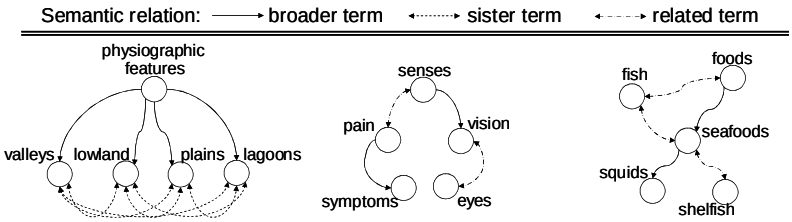
We present a new approach for computing lexical chains by treating them as graphs, where nodes are document terms and edges reflect semantic relations between them. In contrast to previous methods, we analyze the cohesive strength within a chain by computing the diameter of the chain graph. Weakly cohesive chains with a high graph diameter are decomposed by a graph clustering algorithm into several highly cohesive chains. We use WordNet and alternatively a domain-specific thesaurus for obtaining semantic relations between the terms.

We first give an overview of existing methods for computing lexical chains and related areas. Then we discuss the motivation behind the new approach and describe the algorithm in detail. Our evaluation demonstrates the advantages of using extracted lexical chains for the task of automatic text summarization and keyphrase indexing, compared to a simple baseline approach. The results are compared to annotations produced by a group of humans.

2 Related Work

Morris and Hirst (1991) provide the theoretical background behind lexical chains and demonstrate how they can be constructed manually from Roget's thesaurus. The algorithm was re-implemented as soon as digital WordNet and Roget's became available (Barzilay and Elhadad, 1997) and its complexity was improved (Silber and McCoy, 2002; Galley and McKeown, 2003). All these algorithms perform explicit word sense disambiguation while computing the chains. For each word in a document the algorithm chooses only one sense, the one that relates to members of existing lexical chains. Reeve et al. (2006)

Proceedings of the ACL 2007 Student Research Workshop, pages 85–90,
Prague, June 2007.



(a) strong $m = 1$ (b) weak $m = 4$ (c) average $m = 2$

Figure 1. Lexical chains of different cohesive strength.

compute lexical chains with a medical thesaurus and suggest an implicit disambiguation: once the chains are computed, weak ones containing irrelevant senses are eliminated. We also follow this approach.

One of the principles of building lexical chains is that each term must belong to exactly one chain. If several chains are possible, Morris and Hirst (1991) choose the chain to whose overall score the term contributes the most. This score is a sum over weights of semantic relations between chain members. This approach produces different lexical chains depending on the order of words in the document. This is not justified, as the same content can be expressed with different sequences of statements. We propose an alternative order independent approach, where a graph clustering algorithm calculates the chain to which a term should belong.

3 Lexical Chains

The following notation is used throughout the paper. A lexical chain is a graph $G = (V,E)$ with nodes $v_i \in V$ being terms and edges $(v_i, v_j, w_{ij}) \in E$ representing semantic relations between them, where w_{ij} is a weight expressing the strength of the relation.[1] A set of terms and semantic relations building a graph is a valid lexical chain if the graph is *connected*, i.e. there are no unconnected nodes and no isolated groups of nodes.

The *graph distance* $d(v_i, v_j)$ between two nodes v_i and v_j is the minimum length of the path connecting them. And the *graph diameter* is the "longest shortest distance" between any two nodes in a graph, defined as:

(1) $m = \max_{v_i, v_j} d(v_i, v_j)$.

Because semantic relations are either bi-directional or inverse, we treat lexical chains as *undirected* graphs.

3.1 The Cohesive Strength

Lexical cohesion is the property of lexical entities to "stick together" and function as a whole (Morris and Hirst, 1991). How strongly the elements of a lexical chain "stick together," that is the cohesive strength of the chain, has been defined as the sum of semantic relations between every pair of chain members (e.g. Morris and Hirst, 1991; Silber and McCoy, 2002). This number increases with the length of a chain, but longer lexical chains are not necessarily more cohesive than shorter ones.

Instead, we define the cohesive strength as the diameter of the chain graph. Depending on their diameter we propose to group lexical chains as follows:

1. **Strongly cohesive** lexical chains (Fig. 1a) build *fully connected graphs* where each term is related to all other chain members and $m = 1$.
2. **Weakly cohesive** lexical chains (Fig. 1b) connect terms without cycles and with a diameter $m = |V| - 1$.
3. **Moderately cohesive** lexical chains (Fig. 1c) are in-between the above cases with $m \in [1, |V| - 1]$.

To detect individual topics in texts it is more useful to extract strong lexical chains. For example, Figure 1a describes "physiographic features" and 1c refers to "seafood," while it is difficult to summarize the weak chain 1b with a single term. The goal is to compute lexical chains with the highest possible cohesion. Thus, the algorithm must have a way to control the selection.

[1] The initial experiments presented in this paper use an unweighted graph with $w_{i,j} = 1$ for any semantic relation.

3.2 Computing Lexical Chains

The algorithm consists of two stages. First, we compute lexical chains in a text with only one condition: to be included into a chain a term needs to be related to at least one of its members. Then, we apply graph clustering on the resulting weak chains to determine their strong subchains.

I. Determining all chains. First, the documents' n-grams are mapped onto terms in the thesaurus. To improve conflation we ignore stopwords and sort the remaining stemmed words alphabetically. Second, for each thesaurus term *t* that was found in the document we search for an appropriate lexical chain. We iterate over the list *L* containing previously created chains and check whether term *t* is related to any of the members of each chain. The following cases are possible:

1. No lexical chains were found.
   A new lexical chain with the term *t* as a single element is created and included in *L*.
2. One lexical chain was found.
   This chain is updated with the term *t*.
3. Two or more lexical chains were found.
   We merge these chains into a single new chain, and remove the old chains from *L*.

II. Clustering within the weak chains. Algorithms for graph clustering divide sparsely connected graphs into dense subgraphs with a similar diameter. We consider each lexical chain in *L* with diameter $m > 3$ as a weak chain and apply graph clustering to identify highly cohesive subchains within this chain. The list *L* is updated with the newly generated chains and the original chain is removed.

A popular graph clustering algorithm, Markov Clustering (MCL) is based on the idea that "a random walk that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited" (van Dongen, 2000). MCL is implemented as a sequence of iterative operations on a matrix representing the graph. We use ChineseWhispers (Biemann, 2006), a special case of MCL that performs the iteration in a more aggressive way, with an optimized linear complexity with the number of graph edges.

Figure 2 demonstrates how an original weakly cohesive lexical chain has been divided by ChineseWhispers into five strong chains.

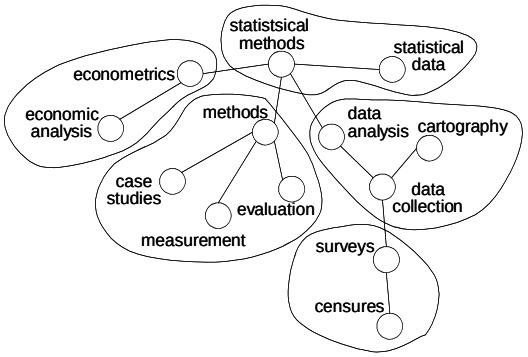


Figure 2. Clustering of a weak chain with ChineseWhispers.

4 Lexical Chains for Text Summarization

Lexical chains are usually evaluated in terms of their performance on the automatic text summarization task, where the most significant sentences are extracted from a document into a summary of a predefined length. The idea is to use the cohesive information about sentence members stored in lexical chains. We first describe the summarization approach and then compare results to manually created summaries.

4.1 Identifying the Main Sentences

The algorithm takes one document at a time and computes its lexical chains as described in Section 3.2, using the lexical database WordNet. First, we consider all semantic senses of each document term. However, after weighting the chains we eliminate senses appearing in low scored chains.

Doran et al. (2004) state that changes in weighting schemes have little effect on summaries. We have observed significant differences between reported functions on our data and achieved best results with the formula produced by Barzilay and Elhadad (1997):

$$(2) \qquad Score(LC) = (1 - \frac{|LC|}{\sum_{t \in LC} freq(t)}) \cdot \sum_{t \in LC} freq(t)$$

Here, $|LC|$ is the length of the chain and *freq(t)* is the frequency of the term *t* in the document. All lexical chains with score lower than a threshold contain irrelevant word senses and are eliminated.

Next we identify the main sentences for the final summary of the document. Different heuristics have been proposed for sentence extraction based on the information in lexical chains. For each top scored chain, Barzilay and Elhadad (1997) extract

		Rater 2	
		Positive	Negative
Rater 1	Positive	a	b
	Negative	c	d

Table 1. Possible choices for any two raters

that sentence which contains the first appearance of a chain member. Doran et al. (2004) sum up the weights all words in the sentence, which correspond to the chain weights in which these words occur. We choose the latter heuristic because it significantly outperforms the former method in our experiments.

The highest scoring sentences from the document, presented in their original order, form the automatically generated summary. How many sentences are extracted depends on the requested summary length, which is defined as the percentage of the document length.

4.2 Experimental Settings

For evaluation we used a subset of a manually annotated corpus specifically created to evaluate text summarization systems (Hasler et al. 2003). We concentrate only on documents with at least two manually produced summaries: 11 science and 29 newswire articles with two summaries each, and 7 articles additionally annotated by a third person. This data allows us to compare the consistency of the system with humans to their consistency with each other.

The results are evaluated with the Kappa statistic κ, defined for Table 1 as follows:

$$\kappa = \frac{2(ab - bc)}{(a+c)(c+9)+(b+d)(a+b)} \quad (3)$$

It takes into account the probability of chance agreement and is widely used to measure inter-rater agreement (Hripcsak and Rothshild, 2005). The ideal automatic summarization algorithm should have as high agreement with human subjects as they have with each other.

We also use a baseline approach (BL) to estimate the advantage of using the proposed lexical chaining algorithm (LCA). It extracts text summaries in exactly the manner described in Section 4.1, with the exception of the lexical chaining stage. Thus, when weighting sentences, the frequencies of *all* WordNet mappings are taken into account without the implicit word sense disambiguation provided by lexical chains.

		Humans	BL	LCA
29 newswire articles	S1	0.32	0.19	0.20
	S2		0.20	0.24
11 science articles	S1	0.34	0.08	0.13
	S2		0.13	0.22

Table 2. Kappa agreement on 40 summaries

	vs. human 2,3 and 1	vs. BL	vs. LCA
human 1	0,41	0,30	0,30
human 2	0,38	0,22	0,24
human 3	0,28	0,17	0,24
average	0,36	0,23	0,26

Table 3. Kappa agreement on 7 newswire articles

4.3 Results

Table 2 compares the agreement among the human annotators and their agreement with the baseline approach BL and the lexical chain algorithm LCA. The agreement between humans is low, which confirms that sentence extraction is a highly subjective task. The lexical chain approach LCA significantly outperforms the baseline BL, particularly on the science articles.

While the average agreement of the LCA with humans is still low, the picture changes when we look at the agreement on individual documents. Human agreement varies a lot (*stdev* = 0.24), while results produced by LCA are more consistent (*stdev* = 0.18). In fact, for over 50% of documents LCA has greater or the same agreement with one or both human annotators than they with each other. The overall superior performance of humans is due to exceptionally high agreement on a few documents, whereas on another couple of documents LCA failed to produce a consistent summary with both subjects. This finding is similar to the one mentioned by Silber and McCoy (2002).

Table 3 shows the agreement values for 7 newswire articles that were summarized by three human annotators. Again, LCA clearly outperforms the baseline BL. Interestingly, both systems have a greater agreement with the first subject than the first and the third human subjects with each other.

5 Lexical Chains for Keyphrase Indexing

Keyphrase indexing is the task of identifying the main topics in a document. The drawback of conventional indexing systems is that they analyze

	Professional Indexers						
	1	2	3	4	5	6	Avg
1		61	51	64	57	57	58
2	61		48	53	60	52	55
3	51	48		54	44	61	51
4	64	53	54		51	57	56
5	57	60	44	51		49	52
6	57	52	61	57	49		55
BL	42	39	37	39	39	35	39
LCA	43	42	40	40	39	40	41

Table 4. Topic consistency over 30 documents

document terms individually. Lexical chains enable *topical* indexing, where first highly cohesive terms are organized into larger topics and then the main topics are selected. Properties of chain members help to identify terms that represent each keyphrases. To compute lexical chains and assign keyphrases this time we use a domain-specific thesaurus instead of WordNet.

5.1 Finding Keyphrases in Lexical Chains

The ranking of lexical chains is essential for determining the main topics of a document. Unlike in summarization, it should capture the specificity of the individual chains. Also, for some topics, e.g. proper nouns, the number of terms to express it can be limited; therefore we average frequencies over all chain members. Our measure of chain specificity combines TFIDFs and term length,[2] which boosts chains containing specific terms that are particularly frequent in a given document:

$$Score(LC) = \frac{\sum_{t \in LC} TFIDF(t) \times \sum_{t \in LC} length(t)}{|LC|} \quad (4)$$

We assume that the top ranked weighted lexical chains represent the main topics in a document. To determine the keyphrases, for each lexical chain we need to choose a term that describes this chain in the best way, just as "seafood" is the best descriptor for the chain in Figure 1c.

Each member of the chain *t* is scored as follows:

$$Score(t) = TFIDF(t) \times ND(t) \times length(t) \quad (5)$$

where *ND(t)* is the node degree, or the number of edges connecting term *t* to other chain members. The top scored term is chosen as a keyphrase.

Professional indexers tend to choose more than one term for a document's most prominent topics. Thus, we extract the top two keyphrases from the top two lexical chains with $|LC| \geq 3$. If the second keyphrase is a broader or a narrower term of the first one, this rule does not apply.

5.2 Evaluation of the Extracted Keyphrases

This approach is evaluated on 30 documents indexed each by 6 professional indexers from the UN's Food and Agriculture Organization. The keyphrases are driven from the agricultural thesaurus Agrovoc[3] with around 40,000 terms and 30,000 semantic relations between them.

The effectiveness of the lexical chains is shown in comparison to a baseline approach, which given a document simply defines keyphrases as Agrovoc terms with top TFIDF values.

Indexing consistency is computed with the F-Measure *F*, which can be expressed in terms of Table 1 (Section 4.1) as following:[4]

$$F = \frac{2a}{2a + b + c} \quad (6)$$

The overlap between two keyphrase sets *a* is usually computed by exact matching of keyphrases. However, discrepancies between professional human indexers show that there are no "correct" keyphrases. Capturing main topics rather than exact term choices is more important. Lexical chains provide a way of measuring this so called *topical consistency*. Given a set of lexical chains extracted from a document, we first compute chains that are covered in its keyphrase set and then compute consistency in the usual manner.

5.3 Results

Table 4 shows topical consistency between each pair of professional human indexers, as well as between the indexers and the two automatic approaches, baseline BL and the lexical chain algorithm LCA, averaged over 30 documents.

The overall consistency between the human indexers is 55%. The baseline BL is 16 percentage points less consistent with the 6 indexers, while

[2] Term length, measured in words, gives an indirect but simple measure of its specificity. E.g., "tropical rain forests" is more specific than "forests".

[3] http://www.fao.org/agrovoc/

[4] When vocabulary is large, the consistency is the same, whether it is computed with the Kappa statistic or the F-Measure (Hripcsak and Rothshild, 2005).

LCA is 1 to 5 percentage points more consistent with each indexer than the baseline.

6 Discussion

Professional human indexers first perform conceptual analysis of a document and then translate the discovered topics into keyphrases. We show how these two indexing steps are realized with lexical chain approach that first builds an intermediate semantic representation of a document and then translates chains into keyphrases. Conceptual analysis with lexical chains in text summarization helps to identify irrelevant word senses.

The initial results show that lexical chains perform better than baseline approaches in both experiments. In automatic summarization, lexical chains produce summaries that in most cases have higher consistency with human annotators than they with each other, even using a simplified weighting technique. Integrating lexical chaining into existing keyphrase indexing systems is a promising step towards their improvement.

The lexical chaining does not require any resources other than a controlled vocabulary. We have shown that it performs well with a general lexical database and with a domain-specific thesaurus. We use the Semantic Knowledge Organization Standard [5] which allows easy inter-changeability of thesauri. Thus, this approach is domain and language independent.

7 Conclusions

We have shown a new method for computing lexical chains based on graph clustering. While previous chaining algorithms did not analyze the lexical cohesion within each chain, we force our algorithm to produce highly cohesive lexical chains based on the minimum diameter of the chain graph. The required cohesion can be controlled by increasing the diameter value and adjusting parameters of the graph clustering algorithm.

Experiments on text summarization and key-phrase indexing show that the lexical chains approach produces good results. It combines symbolic analysis with statistical features and outperforms a purely statistical baseline. The future work will be to further improve the lexical chaining technique and integrate it into a more complex topical indexing system.

8 Acknowledgements

I would like to thank my PhD supervisors Ian H. Witten and Eibe Frank, as well as Gordon Paynter and Michael Poprat and the anonymous reviewers of this paper for their valuable comments. This work is supported by a Google Scholarship.

References

Chris Biemann 2006. Chinese Whispers—an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proc of the HLT-NAACL-06 Workshop on Textgraphs*, pp. 73-80.

Regina Barzilay and Michael Elhadad. 1997. Using Lexical Chains for Text Summarization, In *Proc of the* ACL *Intelligent Scalable Text Summarization Workshop*, pp. 10-17.

Stijn M. van Dongen. 2000. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht.

William P. Doran, Nicola Stokes, Joe Carthy and John Dunnion. 2004. Assessing the Impact of Lexical Chain Scoring Methods on Summarization. In *Proc of CICLING'04*, pp. 627-635.

Laura Hasler, Constantin Orasan and Ruslan Mitkov. 2003. Building Better Corpora for Summarization. In *Proc of Corpus Linguistics CL'0*3, pp. 309-319.

George Hripcsak and Adam S. Rothschild. 2005. Agreement, the F-Measure, and Reliability in IR. *JAMIA*, (12), pp. 296-298.

Jane Morris and Graeme Hirst. 1991. Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text. *Computational Linguistics*, 17(1), pp. 21-48.

Lawrence H. Reeve, Hyoil Han and Ari D. Brooks. 2006. BioChain: Using Lexical Chaining for Biomedical Text Summarization. In *Proc of the ACM Symposium on Applied Computing*, pp. 180-184.

Gregory Silber and Kathleen McCoy, 2002. Efficiently Computed Lexical Chains as an Intermediate Representation for Automatic Text Summarization. *Computational Linguistics*, vol. 28, pp. 487-496.

[5] http://www.w3.org/2004/02/skos/

Clustering Hungarian Verbs on the Basis of Complementation Patterns

Kata Gábor
Dept. of Language Technology
Linguistics Institute, HAS
1399 Budapest, P. O. Box 701/518
Hungary
gkata@nytud.hu

Enikő Héja
Dept. of Language Technology
Linguistics Institute, HAS
1399 Budapest, P. O. Box 701/518
Hungary
eheja@nytud.hu

Abstract

Our paper reports an attempt to apply an unsupervised clustering algorithm to a Hungarian treebank in order to obtain semantic verb classes. Starting from the hypothesis that semantic metapredicates underlie verbs' syntactic realization, we investigate how one can obtain semantically motivated verb classes by automatic means. The 150 most frequent Hungarian verbs were clustered on the basis of their complementation patterns, yielding a set of basic classes and hints about the features that determine verbal subcategorization. The resulting classes serve as a basis for the subsequent analysis of their alternation behavior.

1 Introduction

For over a decade, automatic construction of wide-coverage structured lexicons has been in the center of interest in the natural language processing community. On the one hand, structured lexical databases are easier to handle and to expand because they allow making generalizations over classes of words. On the other hand, interest in the automatic acquisition of lexical information from corpora is due to the fact that manual construction of such resources is time-consuming, and the resulting database is difficult to update. Most of the work in the field of acquisition of verbal lexical properties aims at learning subcategorization frames from corpora e.g. (Pereira et al., 1993; Briscoe and Carroll, 1997; Sass, 2006). However, semantic grouping of verbs on the basis of their syntactic distribution or other quantifiable features has also gained attention (Schulte im Walde, 2000; Schulte im Walde and Brew, 2002; Merlo and Stevenson, 2001; Dorr and Jones, 1996). The goal of these investigations is either the validation of verb classes based on (Levin, 1993), or finding algorithms for the categorization of new verbs.

Unlike these projects, we report an attempt to cluster verbs on the basis of their syntactic properties with the further goal of identifying the semantic classes relevant for the description of Hungarian verbs' alternation behavior. The theoretical grounding of our clustering attempts is provided by the so-called Semantic Base Hypothesis (Levin, 1993; Koenig et al., 2003). It is founded on the observation that semantically similar verbs tend to occur in similar syntactic contexts, leading to the assumption that verbal semantics determines argument structure and the surface realization of arguments. While in English semantic argument roles are mapped to configurational positions in the tree structure, Hungarian codes complement structure in its highly rich nominal inflection system. Therefore, we start from the examination of case-marked NPs in the context of verbs.

The experiment discussed in this paper is the first stage of an ongoing project for finding the semantic verb classes which are syntactically relevant in Hungarian. As we do not have presuppositions about which classes have to be used, we chose an unsupervised clustering method described in (Schulte im Walde, 2000). The 150 most frequent Hungarian verbs were categorized according to their comp-

Proceedings of the ACL 2007 Student Research Workshop, pages 91–96,
Prague, June 2007.

lementation structures in a syntactically annotated corpus, the Szeged Treebank (Csendes et al., 2005). We are seeking the answer to two questions:

1. Are the resulting clusters semantically coherent (thus reinforcing the Semantic Base Hypothesis)?

2. If so, what are the alternations responsible for their similar behavior?

The subsequent sections present the input features [2] and the clustering methods [3], followed by the presentation of our results [4]. Problematic issues raised by the evaluation are discussed in [5]. Future work is outlined in [6]. The paper ends with the conclusions [7].

2 Feature Space

As currently available Hungarian parsers (Babarczy et al., 2005; Gábor and Héja, 2005) cannot be used satisfactorily for extracting verbal argument structures from corpora, the first experiment was carried out using a manually annotated Hungarian corpus, the Szeged Treebank. Texts of the corpus come from different topic areas such as business news, daily news, fiction, law, and compositions of students. It currently comprises 1.2 million words with POS tagging and syntactic annotation which extends to top-level sentence constituents but does not differentiate between complements and adjuncts.

When applying a classification or clustering algorithm to a corpus, a crucial question is which quantifiable features reflect the most precisely the linguistic properties underlying word classes. (Brent, 1993) uses regular patterns. (Schulte im Walde, 2000; Schulte im Walde and Brew, 2002; Briscoe and Carroll, 1997) use subcategorization frame frequencies obtained from parsed corpora, potentially completed by semantic selection information. (Merlo and Stevenson, 2001) approximates diathesis alternations by hand-selected grammatical features. While this method has the advantage of working on POS-tagged, unparsed corpora, it is costly with respect to time and linguistic expertise. To overcome this drawback, (Joanis and Stevenson, 2003) develop a general feature space for supervised verb classification. (Stevenson and Joanis, 2003) investigate the applicability of this general feature space to unsupervised verb clustering tasks. As unsupervised methods are more sensitive to noisy features, the key issue is to filter out the large number of probably irrelevant features. They propose a semi-supervised feature selection method which outperforms both hand-selection of features and usage of the full feature set.

As in our experiment we do not have a pre-defined set of semantic classes, we need to apply unsupervised methods. Neither have we manually defined grammatical cues, not knowing which alternations should be approximated. Hence, similarly to (Schulte im Walde, 2000), we represent verbs by their subcategorization frames.

In accordance with the annotation of the treebank, we included both complements and adjuncts in subcategorization patterns. It is important to note, however, that not only practical considerations lead us to this decision. First, there are no reliable syntactic tests for differentiating complements from adjuncts. This is due to the fact that Hungarian is a highly inflective, non-configurational language, where constituent order does not reveal dependency relations. Indeed, complements and adjuncts of verbs tend to mingle. In parallel, Hungarian presents a very rich nominal inflection system: there are 19 case suffixes, and most of them can correspond to more than one syntactic function, depending on the verb class they occur with. Second, we believe that adjuncts can be at least as revealing of verbal meaning as complements are: many of them are not productive (in the sense that they cannot be added to any verb), they can only appear with predicates the meaning of which is compatible with the semantic role of the adjunct. For these considerations we chose to include both complements and adjuncts in subcategorization patterns.

Subcategorization frames to be extracted from the treebank are composed of case-marked NPs and infinitives that belong to a children node of the verb's maximal projection. As Hungarian is a non-configurational language, this operation simply yields a non-ordered list of the verb's syntactic dependents. There was no upper bound on the number of syntactic dependents to be included in the frame. Frame types were obtained from individual frames by omitting lexical information as well as every piece of morphosyntactic description except

for the POS tag and the case suffix. The generalization yielded 839 frame types altogether.[1]

3 Clustering Methods

In accordance with our goal to set up a basis for a semantic classification, we chose to perform the first clustering trial on the 150 most frequent verbs in the Szeged Treebank. The representation of verbs and the clustering process were carried out based on (Schulte im Walde, 2000). The data to be compared were the maximum likelihood estimates of the probability distribution of verbs over the possible frame types:

$$p(t|v) = \frac{f(v,t)}{f(v)} \tag{1}$$

with $f(v)$ being the frequency of the verb, and $f(v,t)$ being the frequency of the verb in the frame. These values have been calculated for each of the 150 verbs and 839 frame types.

Probability distributions were compared using *relative entropy* as a distance measure:

$$D(x\|y) = \sum_{i=1}^{n} x_i \cdot \log \frac{x_i}{y_i} \tag{2}$$

Due to the large number of subcategorization frame types, verbs' representation comprise a lot of zero probability figures. Using relative entropy as a distance measure compels us to apply a smoothing technique to be able to deal with these figures. However, we do not want to lose the information coded in zero frequencies - namely, the presumable incompatibility of the verb with certain semantic roles associated with specific case suffixes. Since we work with the 150 most frequent verbs, we wish to use a method which is apt to reflect that a gap in the case of a high-frequency lemma is more likely to be an impossible event than in the case of a relatively less frequent lemma (where it might as well be accidental). That is why we have chosen the smoothing technique below:

$$\begin{aligned} f_{\mathrm{e}} &= \frac{0,001}{f(v)} \quad \text{if} \\ f_{\mathrm{c}}(t,v) &= 0 \end{aligned} \tag{3}$$

where f_e is the estimated and f_c is the observed frequency.

Two alternative bottom-up clustering algorithms were then applied to the data:

1. First we employed an agglomerative clustering method, starting from 150 singleton clusters. At every iteration we merged the two most similar clusters and re-counted the distance measures. The problem with this approach, as Schulte im Walde notes on her experiment, is that verbs tend to gather in a small number of big classes after a few iterations. To avoid this, we followed her in setting to four the maximum number of elements occuring in a cluster. This method - and the size of the corpus - allowed us to categorize 120 out of 150 verbs into 38 clusters, as going on with the process would have led us to considerably less coherent clusters. However, the results confronted us with the *chaining effect*, i.e. some of the clusters had a relatively big distance between their least similar members.

2. In the second experiment we put a restriction on the distance between each pair of verbs belonging to the same cluster. That is, in order for a new verb to be added to a cluster, its distance from all of the current cluster members had to be smaller than the maximum distance stated based on test runs. In this experiment we could categorize 71 verbs into 23 clusters. The convenience of this method over the first one is its ability to produce popular yet coherent clusters, which is a particularly valuable feature given that our goal at this stage is to establish basic verb classes for Hungarian. However, we are also planning to run a top-down clustering algorithm on the data to get a probably more precise overview of their structure.

4 Results

With both methods we describe in Section 3, a big part of the verbs showed a tendency to gather together in a few but popular clusters, while the rest of them were typically paired with their nearest synonym (e.g.: *zár* (close) with *végez* (finish) or antonym (e.g.: *ül* (*sit*) with *áll* (stand)). Naturally,

[1] The order in which syntactic dependents appear in the sentence was not taken into account.

method 1 (i.e. placing an upper limit on the number of verbs within a cluster) produced more clusters and gave more valuable results on the least frequent verbs. On the other hand, method 2 (i.e. placing an upper limit on the distance between each pair of verbs within the class) is more efficient for identifying basic verb classes with a lot of members. Given our objective to provide a Levin-type classification for Hungarian, we need to examine whether the clusters are semantically coherent, and if so, what kind of semantic properties are shared among class members. The three most popular verb clusters were investigated first, because they contain many of the most frequent verbs and yet are characterized by strong inter-cluster coherence due to the method used. The three clusters absorbed one third of the 71 categorized verbs. The clusters are the following:

C-1 VERBS OF BEING: *marad* (remain), *van* (be), *lesz* (become), *nincs* (not being)

C-2 MODALS: *megpróbál* (try out), *próbál* (try), *szokik* (used to), *szeret* (like), *akar* (want), *elkezd* (start), *fog* (will), *kíván* (wish), *kell* (must)

C-3 MOVEMENT VERBS: *indul* (leave), *jön* (come), *elindul* (depart), *megy* (go), *kimegy* (go out), *elmegy* (go away)

Verb clusters C-1 and C-3 exhibit intuitively strong semantic coherence, whereas C-2 is best defined along syntactic lines as 'modals'. A subclass of C-2 is composed of verbs which express some mental attitude towards undertaking an action, e.g. (*szeret* (like), *akar* (want), *kíván* (wish)), but for the rest of the verbs it is hard to capture shared meaning components.

It can be said in general about the clusters obtained that many of them can be anchored to general semantic metapredicates or one of the arguments' semantic role, e.g.: CHANGE OF STATE VERBS (*erősödik* (get stronger), *gyengül* (intransitive weaken), *emelkedik* (intransitive rise)), verbs with a beneficiary role (*biztosít* (guarantee), *ad* (give), *nyújt* (provide), *készít*(make)), VERBS OF ABILITY (*sikerül* (succeed), *lehet* (be possible), *tud* (be able, can)). Some clusters seem to result from a tighter semantic relation, e.g. VERBS OF APPEARANCE or VERBS OF JUDGEMENT were put together. In other cases the relation is broader as verbs belonging to the class seem to share only aspectual characteristics, e.g. AGENTIVE VERBS OF CONTINUOS ACTIVITIES (*ül* (be sitting), *áll* (be standing), *lakik* (live somewhere), *dolgozik* (work)). At the other end of the scale we find one group of verbs which 'accidentally' share the same syntactic patterns without being semantically related (*foglalkozik* (deal with sg), *találkozik* (meet sy), *rendelkezik* (dispose of sg)).

5 Evaluation and Discussion

As (Schulte im Walde, 2007) notes, there is no widely accepted practice of evaluating semantic verb classes. She divides the methods into two major classes. The first type of methods assess whether the resulting clusters are coherent enough, i. e. elements belonging to the same cluster are closer to each other than to elements outside the class, according to an independent similarity/distance measure. However, relying on such a method would not help us evaluating the semantic coherence of our classes. The second type of methods use gold standards. Widely accepted gold standards in this field are Levin's verb classes or verbal WordNets. As we do not dispose of a Hungarian equivalent of Levin's classification – that is exactly why we experiment with automatic clustering – we cannot use it directly.

We also run across difficulties when considering Hungarian verbal WordNet (Kuti et al., 2005) as the standard for evaluation. Mapping verb clusters to the net would require to state semantic relatedness in terms of WordNet-type hierarchy relations. However, if we try to capture the distance between verbal meanings by the number of intermediary nodes in the WordNet, we face the problem that the semantic distance between mother-children nodes is not uniform.

As our work is about obtaining a Levin-type verb classification, it could be an obvious choice to evaluate semantic classes by collecting alternations specific to the given class. Hungarian language hardly lends itself to this method because of its peculiar syntactic features. The large number of subcategorization frames and the optionality of most complements and adjuncts yield too much possible alterna-

	acc	ins	abl	ela
indul	-	ins/com	source	source
jön	-	ins/com	source	source
elindul	-	ins/com	source	source
megy	-	ins/com	source	source
kimegy	-	ins/com	source	source
elmegy	-	ins/com	source	source

Table 1: The semantic roles of cases beside C-3 verb cluster

tions. Hence, we decided to narrow down the scope of investigation. We start from verb clusters and the meaning components their members share. Then we attempt to discover which semantic roles can be licenced by these meaning components. If verbs in the same cluster agree both in being compatible with the same semantic roles and in the syntactic encoding of these roles, we consider that they form a correct cluster.

To put it somewhat more formally, we represent verb classes by matrices with a) nominal case suffixes in columns and b) individual verb lemmata in rows. The first step of the evaluation process is to fill in the cells with the semantic roles the given suffix can code in the context of the verb. We consider the clusters correct, if the corresponding matrices meet two requirements:

1. They have to be specific to the cluster.
2. Cells in the same column have to contain the same semantic role.

Tables 1. and 2. illustrate coherent and distinctive case matrices[2].

According to Table 1. ablative case, just as elative, codes a physical source in the environment of movement verbs. Both cases having the same semantic role, the decision between them is determined by the semantics of the corresponding NP. These cases code an other semantic role – cause – in the case of verbs of existence (Table 2).

It is important to note that we do not dispose of a preliminary list of semantic roles. To avoid arbitrary

[2] *Com* is for comitative – approximately encoding the meaning 'together with' , *ins* is for the instrument of the described event, *source* denotes a starting point in the space, *cause* refers to entity which evoked the eventuality described by the verb.

	acc	ins	abl	ela
marad	-	com	cause	material
van	-	com	cause	material
lesz	-	com	cause	material
nincs	-	com	cause	material

Table 2: The semantic roles of cases beside C-1 verb cluster

or vague role specifications, we need more than one persons to fill in the cells, based on example sentences.

6 Future Work

There are two major directions regarding our future work. With respect to the automatic clustering process, we have the intention of widening the scope of the grammatical features to be compared by enriching subcategorization frames by other morphological properties. We are also planning to test top-down clustering methods such as the one described in (Pereira et al., 1993). On the long run, it will be inevitable to make experiments on larger corpora. The obvious choice is the 180 million words Hungarian National Corpus (Váradi, 2002). It is a POS-tagged corpus but does not contain any syntactic annotation; hence its use would require at least some partial parsing such as NP analysis to be employable for our purposes. The other future direction concerns evaluation and linguistic analysis of verb clusters. We define well-founded verb classes on the basis of semantic role matrices. These semantic roles can be filled in a sentence by case-marked NPs. Therefore, evaluation of automatically obtained clusters presupposes the definition of such matrices, which is our major linguistic task in the future. When we have the supposed matrices at our disposal, we can start evaluating the clusters via example sentences which illustrate case suffix alternations or roles characteristic to specific classes.

7 Conclusions

The experiment of clustering the 150 most frequent Hungarian verbs is the first step towards finding the semantic verb classes underlying verbs' syntactic distribution. As we did not have presuppositions

about the relevant classes, neither any gold standard for automatic evaluation, the results have to serve as input for a detailed linguistic analysis to find out at what extent they are usable for the syntactic description of Hungarian. However, as demonstrated in Section 4, the verb clusters we got show surprisingly transparent semantic coherence. These results, obtained from a corpus which is by several orders of magnitude smaller than what is usual for such purposes, is a reinforcement of the usability of the Semantic Base Hypothesis for language analysis. Our further work will emphasize both the refinement of the clustering methods and the linguistic interpretation of the resulting classes.

References

Anna Babarczy, Bálint Gábor, Gábor Hamp, András Kárpáti, András Rung and István Szakadát. 2005. Hunpars: mondattani elemző alkalmazás [Hunpars: A rule-based sentence parser for Hungarian]. *Proceedings of the 3th Hungarian Conference of Computational Linguistics (MSZNY05)*, pages 20-28, Szeged, Hungary.

Michael R. Brent. 1993. From grammar to lexicon: unsupervised learning of lexical syntax. *Computational Linguistics*, 19(2):243–262, MIT Press, Cambridge, MA, USA.

Ted Briscoe and John Carroll. 1997. Automatic Extraction of Subcategorization from Corpora. *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*, pages 356–363, Washington, DC, USA.

Dóra Csendes, János Csirik, Tibor Gyimóthy and András Kocsor. 2005. The Szeged Treebank. *LNCS series Vol. 3658*, 123-131.

Bonnie J. Dorr and Doug Jones. 1996. Role of Word Sense Disambiguation in Lexical Acquisition: Predicting Semantics from Syntactic Cues. *Proceedings of the 14th International Conference on Computational Linguistics (COLING-96)*, pages 322–327, Kopenhagen, Denmark.

Kata Gábor and Enikő Héja. 2005. Vonzatok és szabad határozók szabályalapú kezelése [A Rule-based Analysis of Complements and Adjuncts]. *Proceedings of the 3th Hungarian Conference of Computational Linguistics (MSZNY05)*, pages 245-256, Szeged, Hungary.

Eric Joanis and Suzanne Stevenson. 2003. A general feature space for automatic verb classification. *Proceedings of the 10th Conference of the EACL (EACL 2003)*, pages 163–170, Budapest, Hungary.

Jean-Pierre Koenig, Gail Mauner and Breton Bienvenue. 2003. Arguments for Adjuncts. *Cognition*, 89, 67-103.

Judit Kuti, Péter Vajda and Károly Varasdi. 2005. Javaslat a magyar igei WordNet kialakítására [Proposal for Developing the Hungarian WordNet of Verbs]. *Proceedings of the 3th Hungarian Conference of Computational Linguistics (MSZNY05)*, pages 79–87, Szeged, Hungary.

Beth Levin. 1993. *English Verb Classes And Alternations: A Preliminary Investigation*. Chicago University Press.

Paola Merlo and Suzanne Stevenson. 2001. Automatic Verb Classification Based on Statistical Distributions of Argument Structure. *Computational Linguistics, 27(3)*, pages 373-408.

Fernando C. N. Pereira, Naftali Tishby and Lillan Lee. 1993. Distributional Clustering of English Words. *31st Annual Meeting of the ACL*, pages 183-190, Columbus, Ohio, USA.

Bálint Sass. 2006. Igei vonzatkeretek az MNSZ tagmondataiban [Exploring Verb Frames in the Hungarian National Corpus]. *Proceedings of the 4th Hungarian Conference of Computational Linguistics (MSZNY06)*, pages 15–22, Szeged, Hungary.

Sabine Schulte im Walde. 2000. Clustering Verbs Semantically According to their Alternation Behaviour. *Proceedings of the 18th International Conference on Computational Linguistics (COLING-00)*, pages 747–753, Saarbrücken, Germany.

Sabine Schulte im Walde and Chris Brew. 2002. Inducing German Semantic Verb Classes from Purely Syntactic Subcategorisation Information. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 223-230, Philadelphia, PA.

Sabine Schulte im Walde. to appear. The Induction of Verb Frames and Verb Classes from Corpora. *Corpus Linguistics. An International Handbook.*, Anke Lüdeling and Merja Kytö (eds). Mouton de Gruyter, Berlin.

Suzanne Stevenson and Eric Joanis. 2003. Semi-supervised Verb Class Discovery Using Noisy Features. *Proceedings of the 7th Conference on Computational Natural Language Learning (CoNLL-03)*, pages 71-78, Edmonton, Canada.

Tamás Váradi. 2002. The Hungarian National Corpus. *Proceedings of the Third International Conference on Language Resources and Evaluation*, pages 385–389, Las Palmas, Spain.

Author Index