Proceedings of the
4th Web as Corpus Workshop (WAC-4)
Can we beat Google?

Edited by Stefan Evert, Adam Kilgarriff and Serge Sharoff

Marrakech, Morocco
1 June 2008

Workshop Programme

9.15 – 9.30	Welcome & Introduction

Session 1: Can we do better than Google?

9.30 – 10.00	**Reranking Google with GReG** *Rodolfo Delmonte, Marco Aldo Piccolino Boniforti*
10.00 – 10.30	**Google for the Linguist on a Budget** *András Kornai, Péter Halácsy*
10.30 – 11.00	Coffee break

Session 2: Cleaning up the Web

11.00 – 11.30	**Victor: the Web-Page Cleaning Tool** *Miroslav Spousta, Michal Marek, Pavel Pecina*
11.30 – 12.00	**Segmenting HTML pages using visual and semantic information** *Georgios Petasis, Pavlina Fragkou, Aris Theodorakos, Vangelis Karkaletsis, Constantine D. Spyropoulos*
12.00 – 12.45	Star Talk: **Identification of Duplicate News Stories in Web Pages** *John Gibson, Ben Wellner, Susan Lubar*
12.45 – 13.30	Group discussion on **The Next CLEANEVAL**
13.30 – 15.00	Lunch break

Session 3: Compilation of Web corpora

15.00 – 15.30	**GlossaNet 2: a linguistic search engine for RSS-based corpora** *Cédrick Fairon, Kévin Macé, Hubert Naets*
15.30 – 16.00	**Collecting Basque specialized corpora from the web: language-specific performance tweaks and improving topic precision** *Igor Leturia Azkarate, Iñaki San Vicente, Xabier Saralegi, Maddalen Lopez de Lacalle*
16.00 – 16.30	Coffee break

Session 3 (cont'd)

16.30 – 17.15	Star Talk: **Introducing and evaluating ukWaC, a very large Web-derived corpus of English** *Adriano Ferraresi, Eros Zanchetta, Silvia Bernardini, Marco Baroni*

Session 4: Technical applications of Web data

17.15 – 17.45	**RoDEO: Reasoning over Dependencies Extracted Online** *Reda Siblini, Leila Kosseim*
17.45 – 18.15	General discussion
18.15	Wrap-up & Conclusion

Workshop Organisers

Stefan Evert, *University of Osnabrück*

Adam Kilgarriff, *Lexical Computing*

Serge Sharoff, *University of Leeds*

Programme Committee

Silvia Bernardini, *U of Bologna, Italy*

Massimiliano Ciaramita, *Yahoo! Research Barcelona, Spain*

Jesse de Does, *INL, Netherlands*

Katrien Depuydt, *INL, Netherlands*

Stefan Evert, *U of Osnabrück, Germany*

Cédrick Fairon, *UCLouvain, Belgium*

William Fletcher, *U.S. Naval Academy, USA*

Gregory Grefenstette, *Commissariat à l'Énergie Atomique, France*

Péter Halácsy, *Budapest U of Technology and Economics, Hungary*

Katja Hofmann, *U of Amsterdam, Netherlands*

Adam Kilgarriff, *Lexical Computing Ltd, UK*

Igor Leturia, *Elhuyar Fundazioa, Basque Country, Spain*

Phil Resnik, *U of Maryland, College Park, USA*

Kevin Scannell, *Saint Louis U, USA*

Gilles-Maurice de Schryver, *U Gent, Belgium*

Klaus Schulz, *LMU München, Germany*

Serge Sharoff, *U of Leeds, UK*

Eros Zanchetta, *U of Bologna, Italy*

Contents

Preface

We want the Demon, you see, to extract from the dance of atoms only information that is genuine, like mathematical theorems, fashion magazines, blueprints, historical chronicles, or a recipe for ion crumpets, or how to clean and iron a suit of asbestos, and poetry too, and scientific advice, and almanacs, and calendars, and secret documents, and everything that ever appeared in any newspaper in the Universe, and telephone books of the future . . .

Stanisław Lem (1985). *The Cyberiad*, translated by Michael Kandel.

Can we beat Google? It is a big question.

First, it is as well to remember that Google is the *non plus ultra* of Internet startups. It is amazing. It is an outrageous fantasy come true, in terms of both speed and accuracy and the fabulous wealth accruing to its founders. If the Internet has fairy tales, this is it. We don't even think of it as an Internet startup any more: it transcended that long ago,[1] as it entered the lexicons of the world,[2] changed the way we live our lives, and diverted a substantial share of the world's advertising spend through its coffers.

Their success is no accident. What they do, they do very well. It would be a bad idea to compete head-on. The core of their business is to index as much of the Web as possible, and make it available very very quickly to people who want to find out about things or – better, from Google's point of view – buy things. And, of course, to carry advertisements and thereby to make oodles of money. In order to do that, they address a large number of associated tasks, including finding text-rich Web pages, finding the interesting text in a Web page, partitioning and identifying duplicates, near-duplicates and clusters.

Much of what they do overlaps with much of what we do, as Web corpus collectors with language technology and linguistic research in mind. But the goals are different, which opens up a space to identify tasks that they perform well from their point of view but that is different to ours, and others that they do, but are not central to their concerns and we can do better.

An example of the first kind is de-duplication. In an impressive study of different methods, Monika Henzinger, formerly Director of Research at Google, discusses pages from a UK business directory that list in the centre the phone number for a type of business for a locality. Two such pages differ in five or less tokens while agreeing in about 1000. From Google's point of view they should not be classified as near-duplicates. From ours, they should. The paper by Gibson *et al.* in this volume addresses duplication from a WAC point of view.

The two biggest languages in the world, one of which is Google's home language, don't have much, or any, inflectional morphology, which may be why Google doesn't consider it so important for search. Speakers of most of the world's languages might give it a higher priority. In general, Google's spectacular performance relates to the languages where they have applied most effort, notably English. For Basque (for which the Web is not so large, and which has ample inflectional morphology) Leturia and colleagues clearly do beat Google on a number of counts.

We know that Google must do lots of text cleaning, as they succeed in finding terms for indexing and also are able to provide, for example, HTML versions of PDF or Word pages. But they do not publish details, so how might we find out what they do, and how it compares to what we do?

[1]As far as anything is long ago in its ten-year life. It was not yet a company when Tony Blair became UK Prime Minister, and was only a two-year-old when George W. Bush arrived in the White House.

[2]Most of Google's 6570 hits for *googlant* are for the present participle of the French verb; most of the 57,900 for *googlest* are for the second person singular of the German verb; most of the 66,400 hits of гуглить are for the infinitive of the Russian verb.

One way to explore the question is by looking at Web1T, a remarkable resource that Google generously provided for academic research in 2006 which lists all 1-, 2-, 3-, 4- and 5-grams occurring more than 40 times on the Google-indexed English Web. According to the brief description of the resource that is all that is provided, it is based on a trillion words. It seems likely that the counts are from de-duplicated pages. The text in the pages has clearly been identified as text (in contrast to images, formatting, etc), tokenised, and has had its language identified.

This resource can be compared to results used in the WaC community[3] and to traditional corpora, such as the BNC. Preliminary results show that our corpora are not worse than the results of Google. Web1T unigrams and bigrams contain more boilerplate (*unsubscribe, rss, forums*), business junk (*poker, viagra, collectibles*) as well as porn (*porn, lingerie*). There are reasons why this information is kept by Google: it is necessary to keep them as relevant keywords if someone is searching for a forum, poker or pornography.

However, we are different: we are searching constructions, not products. So we need different tools and resources, which cannot be provided by Google. Submissions to this volume show that the tools and resources can be provided by us.

Adam Kilgarriff, Serge Sharoff, Stefan Evert

[3]Sharoff in `http://wackybook.sslmit.unibo.it/` or Ferraresi *et al.* in this volume

RERANKING GOOGLE WITH GReG

Rodolfo Delmonte, °Marco Aldo Piccolino Boniforti

° University of Cambridge
marcoaldo.piccolinoboniforti@poste.it

Department of Language Sciences
Università Ca' Foscari – Ca' Bembo
30123, Venezia, Italy
delmont@unive.it

Abstract

We present an experiment evaluating the contribution of a system called GReG for reranking the snippets returned by Google's search engine in the 10 best links presented to the user and captured by the use of Google's API. The evaluation aims at establishing whether or not the introduction of deep linguistic information may improve the accuracy of Google or rather it is the opposite case as maintained by the majority of people working in Information Retrieval and using a Bag Of Words approach. We used 900 questions and answers taken from TREC 8 and 9 competitions and execute three different types of evaluation: one without any linguistic aid; a second one with tagging and syntactic constituency contribution; another run with what we call Partial Logical Form. Even though GReG is still work in progress, it is possible to draw clearcut conclusions: adding linguistic information to the evaluation process of the best snippet that can answer a question improves enormously the performance. In another experiment we used the actual associated to the Q/A pairs distributed by one of TREC's participant and got even higher accuracy.

1. Introduction

We present an experiment run using Google API and a fully scaled version of GETARUNS, a system for text understanding [1;2], together with a modified algorithm for semantic evaluation presented in RTE3 under the acronym of VENSES [3]. The aim of the experiment and of the new system that we called GReG (GETARUNS ReRANKS Google), is that of producing a reranking of the 10 best candidates presented by Google in the first page of a web search. Reranking is produced solely on the basis of the snippets associated to each link – two per link.
GReG uses a very "shallow" linguistic analysis which nonetheless ends up with a fully instantiated sentence level syntactic constituency representation, where grammatical functions have been marked on a totally bottom-up analysis and the subcategorization information associated to each governing predicate – verb, noun, adjective. More on this process in the sections below.
At the end of the parsing process, GReG produces a translation into a flat minimally recursive Partial Logical Form (hence PLF) where besides governing predicates – which are translated into corresponding lemmata – we use the actual words of the input text for all linguistic relations encoded in the syntactic structure.
The idea behind the experiment was this:
- given the recurrent criticisms raised against the possibility to improve web searches by means of information derived from linguistic representations we intended to test the hypothesis to the contrary;
- to this aim we wanted to address different levels of representations – syntactic and (quasi) logical/semantic, and measure their contribution if any in comparison to a simple (key) word-based computation;
- together with linguistic representation, we also wanted to use semantic similarity evaluation techniques already introduced in RTE challenges which seem particularly adequate to measure the degree of semantic similarity and also semantic consistency or non-contradictoriness of the two linguistic descriptions to compare.

The evaluation will focus on a subset of the questions used in TREC [4] made up of 900 question/answers pairs and produces the following data:
- how many times the answer is contained in the 10 best candidates retrieved by Google;
- how many times the answer is ranked by Google in the first two links – actually we will be using only snippets (first two half links);
- as a side-effect, we also know how many times the answer is not contained in the 10 best candidates and is not ranked in the first two links;
- how many times GReG finds the answer and reranks it in the first two snippets;
- how much contribution is obtained by the use of syntactic information;
- how much contribution is obtained by means of LF, which works on top of syntactic representation;
- how much contribution is obtained by modeling the possible answer from the question, also introducing some Meta operator – se use OR and the *.
Eventually, we compute accuracy measures by means of the usual Recall/Precision formula.

2. The Parser

The architecture of the parser is shown in Fig. 1 below and will be commented in this section. It is a quite common pipeline and all the code runs in Prolog and is made up of manually built symbolic rules.

We defined our parser "mildly bottom-up" because the structure building process cycles on a procedure that collects constituents. This is done in three stages: at first chumks are built around semantic heads – verb, noun, adjective. Then prepositions and verb particles are lumped together. In this phase, also adjectives are joined to the nominal head they modify. In a third phase, sentential structure information is added at all levels – main, relative clauses, complement clauses. In presence of conjunction different strategies are applied according to whether they are coordinating or subordinating conjunctions.

An important linguistic step is carried out during this pass: subcategorization information is used to tell complements – which will become arguments in the PLF – and adjuncts apart. Some piece of information is also offered by linear order: SUBJect NPs will usually occur before the verb and OBJect NP after. Constituent labels are then substituted by Grammatical Function labels. The recursive procedure has access to calls collecting constituents that identify preverbal Arguments and Adjuncts including the Subject if any: when the finite verb is found the parser is hampered from accessing the same preverbal portion of the algorithm and switches to the second half of it where Object NPs, Clauses and other complements and adjuncts may be parsed. Punctuation marks are also collected during the process and are used to organize the list of arguments and adjuncts into tentative clauses.

The clause builder looks for two elements in the input list: the presence of the verb-complex and punctuation marks, starting from the idea that clauses must contain a finite verb complex: dangling constituents will be adjoined to their left adjacent clause, by the clause interpreter after failure while trying to interpret each clause separately.

The clause-level interpretation procedure interprets clauses on the basis of lexical properties of the governing verb. This is often non available in snippets. So in many cases, sentence fragments are built.

If the parser does not detect any of the previous structures, control is passed to the bottom-up/top-down parser, where the recursive call simulates the subdivision of structural levels in a grammar: all sentential fronted constituents are taken at the CP level and the IP (now TP) level is where the SUBJect NP must be computed or else the SUBJect NP may be in postverbal position with Locative Inversion structures, or again it might be a subjectless coordinate clause. Then again a number of ADJuncts may be present between SUBJect and verb, such as adverbials and parentheticals. When this level is left, the parser is expecting a verb in the input string. This can be a finite verb complex with a number of internal constituents, but the first item must be definitely a verb. After the (complex) verb has been successfully built, the parser looks for complements: the search is restricted by lexical information. If a copulative verb has been taken, the constituent built will be labelled accordingly as XCOMP where X may be one of the lexical heads, P,N,A,Adv.

The clause-level parser simulates the sentence typology where we may have verbal clauses as SUBJect, Inverted postverbal NPs, fronted that-clauses, and also fully inverted OBJect NPs in preverbal position.

2.1 Parsing and Robust Techniques

The grammar is equipped with a lexicon containing a list of fully specified inflected word forms where each entry is followed by its lemma and a list of morphological features, organized in the form of attribute-value pairs. However, morphological analysis for English has also been implemented and used for Out of Vocabulary (hence OOV) words. The system uses a core fully specified lexicon, which contains approximately 10,000 most frequent entries of English. Subcategorization is derived from FrameNet, VerbNet, PropBank and NomBank. These are all consulted at runtime. Eventually the semantics from the WordNet and other sources derived from the web make up the encyclopaedia. In addition to that, there are all lexical forms provided by a fully revised version of COMLEX. In order to take into account phrasal and adverbial verbal compound forms, we also use lexical entries made available by UPenn and TAG encoding. Their grammatical verbal syntactic codes have then been adapted to our formalism and is used to generate an approximate subcategorization scheme with an approximate aspectual and semantic class associated to it. Semantic inherent features for OOV words , be they nouns, verbs, adjectives or adverbs, are provided by a fully revised version of WordNet – 270,000 lexical entries - in which we used 75 semantic classes similar to those provided by CoreLex.

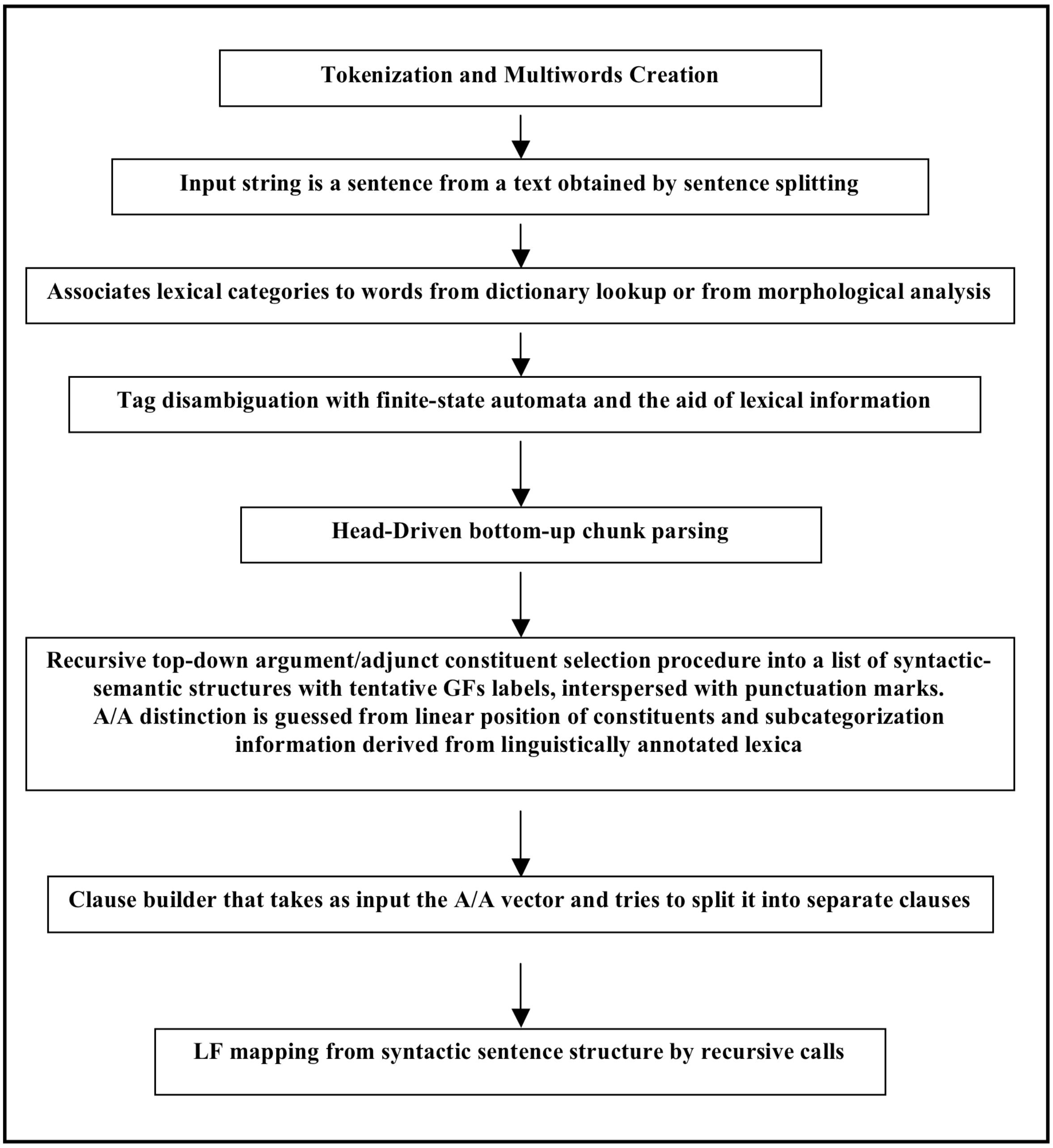

Tokenization and Multiwords Creation
Input string is a sentence from a text obtained by sentence splitting
Associates lexical categories to words from dictionary lookup or from morphological analysis
Tag disambiguation with finite-state automata and the aid of lexical information
Head-Driven bottom-up chunk parsing
Recursive top-down argument/adjunct constituent selection procedure into a list of syntactic-semantic structures with tentative GFs labels, interspersed with punctuation marks. A/A distinction is guessed from linear position of constituents and subcategorization information derived from linguistically annotated lexica
Clause builder that takes as input the A/A vector and tries to split it into separate clauses
LF mapping from syntactic sentence structure by recursive calls

Figure 1: Pipeline of parsing modules for hybrid (bottomup-topdown) version of GReG

3. The Experiment

As said above, the idea is to try to verify whether deep/shallow linguistic processing can contribute to question answering. As will be shown in the following tables, Google's search on the web has high accuracy in general: almost 90% of the answers are present in the first ten results presented to the user. However, we wanted to assume a much stricter scenario closer in a sense to TREC's tasks. To simulate a TREC task as close as possible we decided that only the first two snippets – not links - can be regarded a positive result for the user. Thus, everything that is contained in any of the following snippets will be computed as a negative result.

The decision to regard the first two snippets as distinctive for the experiment is twofold. On the one side we would like to simulate as close as possible a TREC Q/A task, where however rather than presenting precise answers, the system is required to present the sentence/snippet containing it. The other reason is practical or empirical

and is to keep the experiment user centered: user's attention should not be forced to spend energy in a tentative search for the right link. Focussing attention to only two snippets and two links will greatly facilitate the user. In this way, GReG could be regarded as an attempt at improving Google's search strategies and tools.
In order to evaluate the contribution of different levels of computation and thus get empirical evidence that a linguistically-based approach is better than a bag-of-words approach we organized the experiment into a set of concentric layers of computation and evaluation as follows:
- at the bottom level of computation we situated what we call the "default semantic matching procedure". This procedure is used by all the remaining higher level of computation and thus it is easy to separate its contribution to the overall evaluation;
- the default evaluation takes input from the first two processes, tokenization & multiword creation plus sentence splitting. Again these procedures are quite standard and straightforward to compute. So we want to assume that the results are easily reproducible as well as the experiment itself;
- the following higher level of computation may be regarded more system dependent but again it also can be easily reproduced using off-the-shelf algorithms made available for English by research centers all over the world. It regards tagging and context-free PennTreebank-like phrase-structure syntactic representation. Here we consider not only words, but word-tag pairs and word-as-head of constituent N pairs.
- the highest level is constituted by what we call Partial Logical Form, which builds a structure containing a Predicate and a set of Arguments and Adjuncts each headed by a different functor. In turn each such structure can contain Modifiers. Each PLF can contain other PLFs recursively embedded with the same structure. More on this below.
We now present three examples taken from TREC8 question/answer set, no. 3, 193, 195, corresponding respectively to ours 1,2,3. For each question we add the answer and then we show the output of tagging in PennTreebank format, then follows our enriched tagset and then the syntactic constituency structure produced by the parser. Eventually, we show the Partial Logical Form where the question word has been omitted. It can be reinserted in the analysis when the matching takes place and may appear in the other level of representation we present which is constituted by the Query in answer form passed to Google. Question words are always computed as argument or adjunct of the main predicate, so GReG will add a further match with the input snippets constituted by the conceptual substitutes of the wh-words. One substitute is visible in question no.3 when the concept "AUTHOR" is automatically added by GReG in front of the verb and after the star.

(1) What does Peugeot company manufacture? – Cars
(2) Who was the 16th President of the United States? – Lincoln
(3) Who wrote "Dubliners"? – James Joyce

Here below are the analysis where we highlight the various levels of linguistic representation relevant for our experiment only – except for the default word level:

(1) Tagging and Syntactic Constituency
what-wp, does-md, the-dt, Peugeot-nnp, company-nn, manufacture-vin, ? – pun
[what-int, does-vsup, the-art, Peugeot-n, company-n, manufacture-vin, ? - puntint]

cp-[cp-[what], f-[subj-[the, company, mod-[Peugeot]], ibar-[does, manufacture]], fint-[?]]

Partial Logical Form
pred(manufacture) arg([company, mod([Peugeot])]) adj([[], mod([[]])])

Query launched to Google API
Peugeot company manufacture *

(2) Tagging and Syntactic Constituency
who-wp, was-vbd, the-dt, 16th-cd, President-nnp, of-in, the-dt, United_States-nnp, ? – pun
[who-int, was-vc, the-art, 16th-num, President-n, of-p, the-art, United_States-n, ? - puntint]

fint-[cp-[who], ibar-[was], sn-[the, 16th, President, mod-[of, the, United_States]], fint-[?]]

Partial Logical Form
[pred(be) arg([President, mod([united, States, 16th])]) adj([])]

Query launched to Google API
United States 16th President was *

(3) Tagging and Syntactic Constituency
who-wp, wrote-vbd_vbn, "-pun, Dubliners-nns, "-pun, ? - pun
[who-int, wrote-vt, "-par, Dubliners-n, "-par, ? - puntint]

cp-[cp-[who], ibar-[wrote], fp-["], sn-[Dubliners], fp-["], fint-[?]]

Partial Logical Form
pred(write) arg([Dubliners, mod([])]) adj([])

Query launched to Google API
* author wrote Dubliners

3.1 Default Semantic Matching Procedure

This is what constitutes the closest process to the BOWs approach we can conceive of. We compare every word contained in the Question with every word contained in each snippet and we only compare content words. Stopwords are deleted.

We match both simple words and multiwords. Multiwords are created on the basis of lexical information already available for the majority of the cases. The system however is allowed to guess the presence of a multiword from the information attached to the adjacent words and again made available in our dictionaries. If the system recognizes the current word as a word starting with uppercase letter and corresponding to one of the first names listed in one of our dictionary it will try to concatenate this word to the following and try at first a match. If the match fails the concatenated word is accepted as a legitimate continuation – i.e. the name – only in case it starts by uppercase letter. Similar checking procedures have been set up for other NEs like universities, research centers, business related institutions etc. In sum, the system tries to individuate all NEs on the basis of the information stored and some heuristic inferential mechanism.

According to the type of NE we will licence a match of a simple word with a multiword in different ways: person names need to match at least the final part of the multiword, or the name institutions, locations etc. need to match as a whole.

3.2 Tags and Syntactic Heads

The second level of evaluation takes as input the information made available by the tagger and the parser. We decided to use the same approach reported in the challenges called RTE where the systems participating could present more than one run and use different techniques of evaluation. The final task was – and is – that of evaluating the semantic similarity between the question and the input snippets made available by Google. However, there is a marked difference to be taken into account and is the fact that in RTE questions where turned into a fully semantically complete assertion; on the contrary, in our case we are left with a question word to be transformed into the most likely linguistic description that can be associated with the rest of the utterance. As most systems participating in TREC challenge have done, the question has to be rephrased in order to predict the possible structure and words contained in the answer, on the basis of the question word and overall input utterance. Some of the questions contained in the TREC list do not actually constitute wh-questions (factoid or list), but are rather imperatives or iussive utterance, which tell the system – and Google – to “describe” or to “name” some linguistic item specified in the following portion of the utterance.

As others have previously done, we classify all wh-words into semantic types and provide substitute words to be place in the appropriate sentence position in order to simulate as close as possible the answer. However, this is only done in one of the modalities in which the experiment has been run. In the other modality, Google receives the actual words contained in the question.

As to experiment itself, and in particular to the matching procedure we set up, the wh- words is not used to match with the snippets. Rather we use possible linguistic items previously associated to the wh- word in a set. We also use the actual wh- words to evaluated negatively snippets containing them. In this way, we prevent similar and identical questions contained in a snippet and pointed by a link to receive a high score. We noticed that Google is unable to detect such mismatches.

We decided to use tag-word pairs in order to capture part of the contextual meaning associated to a given word. Also in the case of pairs word-as-head-of-constituent/constituent label we wanted to capture part of the contextual import of a word in a structural representation and thus its syntactic and semantic relevance in the structure. As will be clear in the following section, this is different from what is being represented in a Logical Form for how partial it may be.

3.3 Partial Logical Form and Relations

The previous match intended to compare words as part of a structure of dependencies where heads played a more relevant role than non-heads, and thus were privileged. In the higher level match what we wanted to check was the possible relations intervening between words: in this case, matching regarded two words at a time. The first and most relevant word was the PREDicate governing a given piece of PLF. The PRED can be the actual predicate governing at sentence level, with arguments and adjuncts, or it can be just the predicate of any of the Arguments/Adjuncts which in turn governed their modifiers.

Matching is at first applied to two predicates and if it succeeds, it is extended to the contents of the Argument or the Adjunct. In other words, if it is relations that this evaluation should measure, any such relations has to involve at least two linguistic elements of the PLF representation under analysis.

Another important matching procedure applied to the snippet is constituted by a check of the verbal complex. We regard the verbal compound as the carrier of semantic important information to be validated at propositional level. However, seen the subdivision of tasks, we assume that we can be satisfied by applying a partial match. This verbal complex match is meant to ascertain whether the question and the answer should be both containing a positive or a negative polarity – thus they should not convey contradictory information. It is also important to check whether the two verbal complexes are factive or not, in that case they can contain opaque or modality operators. This second possibility needs to be matched carefully.

4. Evaluation and Conclusions

Here below we show the output of GReG in relation to one of the three questions presented above, question n.2

```
*********************************************
google7
Evaluation Score from Words and Tags : 31
Evaluation Score from Syntactic Constituent-Heads : 62
Evaluation Score from Partial Logical Form : 62
62 google8
Evaluation Score from Words and Tags : 35
Evaluation Score from Syntactic Constituent-Heads: 70
Evaluation Score from Partial Logical Form :  0
google9
Evaluation Score from Words and Tags : 33
Evaluation Score from Syntactic Constituent-Heads : 66
Evaluation Score from Partial Logical Form : 66

Snippet No.  google9
16th President of the United States ( March 4 , 1861 to April 15 , 1865 ). Nicknames : “ Honest Abe “ “ Illinois Rail - Splitter “. Born : February 12 , 1809 , . . .

Snippet No.  google7
Abraham Lincoln , 16th President of the United States of America , 1864 , Published 1901 Giclee Print by Francis Bicknell Carpenter - at AllPosters . com .

The right answer is : Lincoln

Google's best snippets containing the right answer are:
google8
Who was the 16th president of the united states ? pissaholic . . . . Abraham Lincoln was the Sixteenth President of the United States between 1861 - 1865 . . .
google7
Abraham Lincoln , 16th President of the United States of America , 1864 , Published 1901 Giclee Print by Francis Bicknell Carpenter - at AllPosters . com .

Google's best answer partially coincides with GReG.
*********************************************
```

Passing Questions to Google with GReG’s analysis produces as a result that only for 642 questions the 10 best links contain the answer. Passing Questions to Google as is produces as a result that only in 737 questions the 10 best links contain the answer. In other words, GReG’s analysis of the question that attempts at producing a model answer to use to trigger best results from Google, in fact lowers the ability of Google to search for the answer and select it in the best 10 links.

This should be due to the difficulty in producing the appropriate answer form, by reordering words of the question and adding metasymbols in the appropriate positions. In fact, Google exploits also the linear order of the words contained in the question. So in case there is some mismatch the answer is not readily found or perhaps is available further down in the list of links.

	With GReG’s preanalysis	Without GReG’s anal.
Google’s 10 Best links contain the answer	755 83.01%	694 77.12%
Google’s 10 Best links do not contain the answer	145 16.9%	206 22.8%
Google Rank answer in first 2 snippets	216 28.61%	168 24.21%
Google Rank answer not in first 2 snippets	814 90.45%	803 89.23%

Table 1: Google outputs with and without the intervention of GReG’s question analysis

GReG reranks the answer in first 2 snippets	Only word match	Tagging and Syntactic heads	Partial Logical Form
With GReG’s analysis	375 58.41%	514 68.08%	543 71.92%
Without GReG’s analysis	406 55.09%	493 66.89%	495 67.16%

Table 2: GReG’s outputs at different levels of linguistic complexity

4.2 Comments

The conclusions we may safely draw is the clear improvements in performance of the system when some linguistic information is introduced in the evaluation process. In particular, when comparing the contribution of PLF to the reranking process we see that there is a clear improvement: in the case of reranking without GReG’s question analysis there is a slight but clear improvement in the final accuracy. Also, when GReG is used to preanalyse the question to pass to Google the contribution of PLF is always apparent. The overall data speak in favour of both preanalysing the question and using more linguistic processing.
If we consider Google’s behaviour to the two inputs, the one with actual questions and the one with prospective answers we see that the best results are again obtained when the preanalysis is used (28.6 vs. 24.2); also the number of candidates containing the answer increases remarkably when using GReG preprocessing (83 vs. 77).

4.3 GReG and Question-Answering from Text

In order to verify the ability of our system to extract answers from real text we organized an experiment which used the same 900 question run this time again the texts made available by TREC participants. These texts have two indices at the beginning of each line indicating respectively the question number which they should be able to answer, and the second an abbreviation containing the initial letters of the newspaper name and the date. In fact each line has been extracted by means of automatic splitting algorithms which have really messed up the whole text. In addition, the text itself has been manipulated to produce tokens which however do not in the least correspond to actual words of current orthographic forms in real newspapers. So it took us quite a lot of work to normalize the texts (5Mb.) to make them as close as possible to actual orthography.
Eventually, when we launched our system it was clear that the higher linguistic component could not possibly be used. The reason is quite simple: texts are intermingled with lists of items, names and also with tables. Since there is no principled way to tell these apart from actual texts with sentential structure, we decided to use only tagging and chunking.
We also had to change the experimental setup we used with Google snippets: in this case, since we had to manipulate quite complex structures and the choice was much more noisy, we raised our candidate set from two to four best candidates. In particular we did the following changes:

- we choose all the text stretches containing the answer/s and ranked them according to their semantic similarity;
- then we compared and evaluated these best choices with the best candidates produced by our analyses;
- we evaluated to success every time one of our four best candidates was contained in the set of best choices containing the answer;
- otherwise we evaluated to failure.

In total, we ran 882 questions because some answers did not have the corresponding texts. Results obtained after a first and only run – which took 4 days to complete on an HP workstation with 5GB of RAM, 4 Dual Core Intel processors, under Linux Ubuntu – were quite high in comparison with the previous ones, and are reported here below:

GReG finds the answer in first 4 text stretches	Tagging and Syntactic heads
Without GReG's analysis	684 / 882 77.55%

Table 3: GReG's results with TREC8/9 texts

With respect to the favourable results, we need to consider that using texts provides a comparatively higher quantity of linguistic material to evaluate and so it favours better results.

5. Conclusions

We intend to improve both the question translation into the appropriate format for Google, and the rules underlying the transduction of the Syntactic Structures into a Partial Logical Form. Then we will run the experiments again. Considering the limitations imposed by Google on the total number of questions to submit to the search engine per day, we are unable to increase the number of questions to be used in a single run.
We also intend to run GReG version for text Q/A this time with question rephrasing. We would also like to attempt using PLF with all the text stretches, excluding manually all tables and lists. We are aware of the fact that this would constitute a somewhat contrived and unnatural way of coping of unrestricted text processing. At the same time we need to check whether the improvements we obtained with snippets are attested with complete texts.
Overall, we believe to have shown the validity of our approach and the usefulness of linguistically-based evaluation methods when compared with BOWs approaches. Structural and relational information constitutes a very powerful addition to simple tagging or just word level semantic similarity measures.

6. References

Delmonte R., (2007), Computational Linguistic Text Processing – Logical Form, Semantic Interpretation, Discourse Relations and Question Answering, Nova Science Publishers, New York, ISBN: 1:60021-700-1.

Delmonte R., (2005), Deep & Shallow Linguistically Based Parsing, in A.M.Di Sciullo(ed), UG and External Systems, John Benjamins, Amsterdam/Philadelphia, pp.335-374.

Delmonte R., A. Bristot, M.A.Piccolino Boniforti, S.Tonelli (2007), Entailment and Anaphora Resolution in RTE3, in Proc. ACL Workshop on Text Entailment and Paraphrasing, Prague, ACL Madison, USA, pp. 48-53.

Litkowski, K. C. (2001). Syntactic Clues and Lexical Resources in Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), The Ninth Text Retrieval Conference (TREC-9). NIST Special Publication 500-249. Gaithersburg, MD., 157-166.

Google for the Linguist on a Budget

András Kornai and Péter Halácsy

Budapest University of Technology Media Research Center
{kornai,hp}@mokk.bme.hu

Abstract

In this paper, we present GLB, yet another open source and free system to create and exploit linguistic corpora gathered from the web. A simple, robust web crawl algorithm, a multi-dimensional information retrieval tool, and a crude parallelization mechanism are proposed, especially for researchers working in resource-limited environments.

Introduction

The GLB (Google for the Linguist on a Budget) project grew out of the realization that the current open source search engine infrastructure, in particular the `nutch/lucene/hadoop` effort, is in many ways inadequate for the creation, refinement, and testing of language models (both statistical and rule-based) on large-scale web corpora, especially for researchers working in resource-limited environments such as startup companies and academic departments unlikely to be able to devote hundreds, let alone thousands, of servers to any project.

Section 1 describes `nut`, a simple, robust web crawl algorithm designed with the needs of linguistic corpora gathering in mind. Section 2 details `luc`, an information retrieval tool that facilitates querying along multiple dimensions. We leave `had`, a crude parallelization mechanism sufficient for load balancing dozens (or perhaps hundreds) of CPUs and offering fine control over rerunning versions of different processing steps, to the concluding Section 3.

Many other ways out of the budget predicament have been proposed, and in the rest of this Introduction we discuss these briefly, not so much to criticize these approaches as to highlight the design criteria that emerged from considering them. First, what do we mean by being on a budget? The Google search appliance (GSA) starts at $30,000, which puts it (barely) within reach of grants to individual investigators, and certainly within the reach of better endowed academic departments. Unfortunately, the GSA is an entirely closed system, the internals cannot be tweaked by the investigators, and the whole appliance model is much better suited for a relatively static document collection than for rapid loading of various corpora. Also, the size limitations (maximum of 500k documents) make the GSA too small for typical corpus-building crawls, and the query language is not flexible enough to handle many of the queries that arise in linguistic practice. There is no breaking out of separate software and hardware costs in the GSA, and as our project is providing free (as in beer) and open source (LGPL) software, our goal was to design algorithms that run well on any (x86-) 64-bit system with 8-16 GB memory and 5-10 TB attached storage – today such systems are available at a quarter of the cost of the GSA.

Another, in many ways more attractive, approach is to rely on the Google API, Alexa, or some similar easily accessible search engine cache. Methods of building corpora by selective querying of major search engines have been pioneered by Ghani (2001), and a set of very useful bootstrapping scripts was made available by Baroni and Bernardini (2004). But being parasitic on a major search engine has its own risks. Many of these were discussed in Kilgarriff (2007) and require no elaboration here, but there are issues, in particular *integration, query depth,* and *replicability,* which are worth further discussion.

First, there are many corpora which may be licensed to the researcher but are not available on crawlable pages (and thus are not indexed by the host engine at all). Such corpora, including purpose-built corpora collected by the researchers themselves, can be extremely relevant to the investigation at hand, and the integration of results from the web-based and the internal corpora is a central issue. This applies to the community-based solution proposed by Kilgarriff as well, inasmuch as researchers are often bound by licenses and other contractual obligations that forbid sharing their data with the rest of the community, or even uploading it to the Sketch Engine CorpusBuilder.

Second, with the leading search engine APIs, deeper querying of the sort provided by the Linguist's Search Engine (Resnik and Elkiss, 2005) or the IMS Corpus Workbench (see `http://www .ims.uni-stuttgart.de /projekte/CorpusWorkbench`) is impossible, a matter we shall return to in the concluding Section. Finally, owing to the ever-changing nature of the web, the work is never replicable. This is quite acceptable for brief lexicographic safaris where the objective is simply to find examples, but in the context of system building and regression testing replicability is essential. The main design requirements for GLB stemming from these considerations are as follows. The system must

1. run on commodity hardware (less that $15k per node)
2. hold a useful number of pages (one billion per node)
3. provide facilities for logging, checkpointing, repeating, and balancing subtasks
4. have a useful throughput (one million queries per day)
5. not be a drain on external resources/goodwill

There are various tradeoffs among these requirements that are worth noting. Trivially, relaxing the budget constraint (1) could lead to more capable systems in regards to (2) and (4), but the proposed system is already at the high end of what financially less well endowed researchers, departments, and startups can reasonably afford. In the other direction, as long as the reliability of storage is taken out of the equation (a terabyte non-redundant disk space is now below $1k), memory becomes the limiting factor, and the same design, deployed on 500m or just 100m items, becomes proportionally less memory-intensive, so running the system on a modern laptop with 4GB memory is feasible. As described in Section 2, GLB does not mandate storage of web pages as such, the items of interest may also be sentences or words. For smaller corpora (in the 1m page range) it may make perfect sense to change the unit of indexing from pages to words and, if disk space is available, to store more information about a unit than the raw text, e.g. to precompute the morphological analysis of each word (or even a full or partial syntactic parse, see some speculative remarks at the end of Section 3). Finally, we note that the design goal of 1m queries per day (12 queries/sec) may be too ambitious if all reads are taken on the same non-redundant disks: while in principle this is well within the speed and latency capabilities of ordinary disk drives, in practice a drive may not stand up against sustained use of this intensity for long. However, those who cannot afford high quality SANs may also be in less of a need to issue millions of queries.

1. Nut

Replicability means that pages once crawled and deemed useful must be kept around forever, otherwise later versions of some processing step cannot be run on the same data as the earlier version, which would throw into question whether improvements are due to improvements in the processing algorithm itself or simply to better data. This is not to say that all pages must be in the scope of all queries, just that a simple, `berkdb`-style list of what was included in which experiment must be preserved. This is in sharp contrast to full-function crawler databases, which manage information about when a host and a particular page was last crawled, when it was created/last changed, how many in-links it has, etc.

In general, neither link structure nor recency matters a great deal for a linguistic corpus, as made plain by the fact that the typical (gigaword) corpora in common use are composed of literary and news text that are entirely devoid of links and are, for the news portion, several years outdated. The exhaustiveness of a crawl is also a secondary concern, since there are far more pages than we can expect to be able to analyze in any depth. This means that it is sufficient to download any page just once, and we can have near-zero tolerance toward buggy, intermittent pages: connection timeouts and errorful http responses are sufficient reason never to go to the page again. Also, the simplest breadth-first algorithm has as good a chance to turn up linguistically relevant pages as the more complex approaches taken in large-scale crawlers.

Among the public domain crawlers, `heritrix` (see `http://crawler.archive.org`) has been successfully utilized by Baroni and Kilgarriff (2006) to create high quality gigaword corpora, achieving a crawl throughput of 35 GB/day. Our own experience with `heritrix`, `nutch`, and `larbin` was that sustained rates in this range are difficult to maintain. We had the best results the WIRE crawler (Castillo, 2005), 8-10 GB/day sustained throughput for domains outside `.hu` and nearly twice that for `.hu` (the crawls were run from Budapest, see Halácsy et al 2008).

Our main loop is composed of three stages: management, fetching, and parsing. Since most of the time is spent fetching, interleaving the steps could save little, and would entail concurrency overhead. We manage three data sets: downloaded URLs, forbidden URLs (those that have already displayed some error), and forbidden hosts (those with dns resolution error, no route to host, host unreachable). We do not manage at all, let alone concurrently, link data, recency of crawl per host, or URL ordering. This simplifies the code enormously, and eliminates nearly all the performance problems that plague `heritrix, nutch, larbin` and other highly developed crawlers where clever management of such data is the central effort. To speed up name resolution (host,ip) pairs already resolved are stored in a simple hash table, and we ignore issues of hosts with multiple IPs and the existence of CNAMEs. The three lists we maintain are read into memory once and written on disk for the next stage, so nothing is ever overwritten. As a matter of fact, it is sufficient for the fetcher to simply append to the list of downloaded URLs on disk, since duplicate elimination (which is not a big issue here) can happen as part of building the hash table on the next cycle.

The bulk of the time is spent fetching, and the efficiency of the fetcher is due essentially to the tightly written `ocamlnet` library, which was designed for high performance from the ground up. We use asynchronous, non-blocking I/O throughout, with callbacks that mesh well with the functional paradigm. We keep a maximum of N (in the range 1000-2000) connections open. Just as the `WIRE` and `larbin` (Ailleret, 2003) crawlers, we use `GNU ADNS` (Jackson and Finch, 2006), an asynchronous-capable DNS client library to resolve IP address of unknown hosts. We keep every resolved IP cached, ignoring changes and TTL issues entirely. Asynchronous name resolution improves speed by a factor of 10. Since the fetcher runs in a single process (with OS-level callbacks), the downloaded HTML file is simply appended to the tail of a large batch. In case of errors (including the case when mime type is not text/html) the URL is placed on the forbidden list. Because charset-normalization is a step that cannot always be performed by standard libraries, we prefer to save out the charset information that is given in the http together with the original text and perform the conversion at a later stage. This facility would actually be a very useful addition in crawlers like `WIRE` or `larbin` which perform charset-normalization at download time, especially if the target is a less commonly taught language where the standard conversion libraries are not mature.

The parse step locates `<a href=` and pulls out the following quoted string, normalizing this using the base URL of the page. URLs containing angled brackets, question marks, or space/tab/newline are discarded. It is the responsibility of the management stage to detect duplicates, filter out the forbidden URLs and hosts, and to organize the next pass search in a manner that puts less load on smaller sites,

leveraging the built in ability of `ocamlnet` to serialize requests to a single host.
Altogether, the effort to tailor the crawl to the need of linguists pays off in notably improved throughput: instead of the 35 GB/day reported in Baroni and Kilgarriff (2006), `nut` has a sustained throughput of over 330 GB/day. This number is largely delimited by bandwidth availability at the Budapest Institute of Technology: `nut` is three times as fast (over 20 GB/hour) at night than during the day (8 GB/hour).

2. Luc

In search engine work the assumption that the fundamental unit of retrieval is the document (downloaded page) is rarely questioned. Yet in many classical IR/IE applications, books are broken up into chapters to be ranked (and returned) separately, and in question answering it is generally necessary to pinpoint information even more precisely, breaking documents down to the section, paragraph, or even sentence level. In many linguistic applications the objects of interest are the sentences, but for purposes of morphological analysis we are also interested in systems capable of responding to queries by single words or morphemes. For the smallest elements it is tempting to keep the entire dataset in main memory, but this would entail a drastic loss of efficiency for corpora that go beyond a single DVD: under more realistic query loads the system would page itself to death.
The `luc` IR subsystem of GLB stands neutral on the size or composition of the retrieved unit, but it assumes that in the typical (non-cached) case it will take at least one disk seek to get to it. At the 2GHz clock speeds and 10ms seek latencies typical of contemporary hardware, one can easily invert a 100x100 matrix the time it takes to fetch a single disk block. Thus the name of the game is to minimize the seeks, which means that all information about a retrieval unit that is relevant for speeding up queries must be precomputed and stored in an index kept in memory. `Luc` limits the size of the indexes to 4GB with the idea that at any given time two copies (a working copy and one under update/refresh) must stay in main memory. Since a billion retrieval units (see our goal 2) will require four-byte pointers (seek offsets), the 4GB limit on indexes is very tight, leaving no room for auxiliary indexes or meta-information stored with the offset. But if such information cannot be stored with the document pointer, how can it be accessed?
The key idea is to use the pointer itself, or more precisely, the location of the pointer in memory, to encode this information. We assume a small set of k dimensions, each dimension taking values in the [0,1] interval. Typical features that could be encoded in such numbers include the *page rank* of a document, the *authority* of the site it comes from, the *recency* of the document, its normalized *length*, and so on. In practice, none of these scales requires the granularity provided by 64-bit floats, and there are many quantization techniques we can use to arrive at a more compressed but still useful representation. Without loss of generality, we can assume that in any dimension values are limited to integers in the 0 to M_i range for $i = 1 \ldots k$.
There are important retrieval keys, such as the presence of a word w in a document, which require some encoding to fit into the `luc` model. We rank words by DF (and within a single DF, lexicographically) to arrive at a canonical ordering: in a typical gigaword corpus there will be on the order of a million different words. A single document will be indexed as many times as it has different words, so a gigaword corpus will require perhaps a hundred million pointers (but not more, since the per-document token multiplicities are collapsed).
The entire index is conceptualized as a single k-dimensional array with static bounds M_i. The main advantage of this view is that pointers to documents that should come early on the postings list are located close to the origin, and are accessible as $k-1$ dimensional slices of the original array. For example, if our query involves the terms *plane, of, immanence*, it is the last word which has the highest IDF, and query execution may begin by fetching the contents of the subarray that has the kth coordinate fixed at the value assigned to this word. Since the index array is very sparse, the key to fast execution is to compress it by kd-tree techniques.
In the `luc` model the impact of the different dimensions of classification on memory usage is similar to the impact that building a secondary array would have, but this fact is carefully hidden from the retrieval routines. For example, if we wish our posting lists to contain not just words, but POS-tagged words, the number of pointers per document grows (assuming that not every token of a type gets tagged the same way), and this impacts the size of the tree that supports the sparse array. Once the meta-information stored with a retrieval unit grows beyond 4 bytes, either index size cannot be kept at 4 GB or the number of retrieval units per node must be curtailed. Either way, the design aims squarely at what is likely to be the sweet spot in the memory price/performance curve for the next decade or so, with 8-16 GB DIMMs already reasonably cheap today and 64-128 GB machines likely to be commodity by 2020.

3. Conclusions

GLB is work in progress. `Nut,` the best developed component, is already in the performance tuning stage. It is currently capable of 50-200 URLs/sec, (20 MB/s download bandwidth, more than what our network can sustain), which we consider satisfactory for a single node, and large-grain parallelization in `had` style is not complicated. At the time of this writing `nut` still ignores `robots.txt`, but once this antisocial behavior is fixed it will be ready for release (planned by the time of the meeting) under LGPL.
`Luc` is in a more preliminary stage, especially as we strive to optimize query execution. The design described above is really optimized for the situations where the bulk of the subselection work is carried by the partial ordering that is encoded in any coordinate dimension. This works well for IDF, recency, and all other examples described in the main text, but falls short of the ideal of matching subtree-like patterns in syntactic descriptions (parse structures) that is explored in LSE. Realistically, we do not believe we can keep as much information as a parse tree in memory for each sentence and still maintain high performance characteristics, but this is largely a question of encoding parse information efficiently in an array-based system.

While our current goal is to first support regular expression queries composed of lexical entries and POS tags (i.e. the kind of queries familiar from IMS), and to respond to the more complex LSE-type queries based on a regexp 'stapler' (Bangalore, 1999), it is tempting to speculate how one would go about supporting complex syntactic queries from the get-go. The key issue is to encode syntactic relationships in their own dimensions: for example, in a system where "parse" means identifying the deep cases (Fillmore, 1968; Fillmore, 1977), a separate dimension would be required for each deep case, and even this would only help encoding main clause syntax. Encoding subordination and coordination would require further additions, and so would modifiers, possessives, and other issues considered critical in parsing. The effective balance between complicating the storage structure and query execution time needs to be tested carefully, and it may well turn out to be the case that stapling (which amounts to query-time discarding of false positives) is more effective than precomputing these relationships at load time.

Finally, `had` is still in the early design stage. Again, budget considerations are paramount: we expect neither thousands of highly capable processors nor exabyte storage to be available to GLB users. In fact, we expect no more than some form of shared disk space (e.g. NFS crossmounts or AFS). Tasks are expected to run on a single node for no more than a few hours. Each node will run a demon that can start a single task, and with the volume of task-related transactions staying well below a thousand per hour a single, central batch distributor is sufficient. We expect a rudimentary but usable system to be available together with the first release of `nut`.

Acknowledgments

We thank Dániel Varga (Budapest Institute of Technology) for performing measurements on other crawlers, and the anonymous referees for their penetrating remarks – responding to the issues they raised improved the draft significantly.

4. References

Sebastien Ailleret. 2003. Larbin: Multi-purpose web crawler.

Srinivas Bangalore. 1999. Explanation-based learning and finite state transducers: Application for parsing lexicalized tree-adjoining grammars. In Andras Kornai, editor, *Extended finite state models of language*, pages 160–192. Cambridge University Press.

Marco Baroni and Silvia Bernardini. 2004. Bootcat: Bootstrapping corpora and terms from the web. In *Proceedings of Language Resources and Evaluation Conference (LREC04)*, pages 1313–1316. European Language Resources Association.

Marco Baroni and Adam Kilgarriff. 2006. Large linguistically-processed Web corpora for multiple languages. In *Companion Volume to Proceedings of the European Association of Computational Linguistics*, pages 87–90, Trento.

Carlos Castillo. 2005. *Effective Web Crawling*. PhD Thesis, Department of Computer Science, University of Chile, Santiago.

Charles Fillmore. 1968. The case for case. In E. BachandR. Harms, editor, *Universals in linguistics theory*, pages 1–90. Holt and Rinehart, New York.

Charles Fillmore. 1977. The case for case reopened. In P. ColeandJ. M. Sadock, editor, *Syntax and Semantics 8: Grammatical relations*, pages 59–82. Academic Press, New York.

Rayid Ghani. 2001. Combining labeled and unlabeled data for text classification with a large number of categories. *ICDM, First IEEE International Conference on Data Mining (ICDM'01)*, 01:597–.

Péter Halácsy, Andrá1s Kornai, Péter Németh, and Dániel Varga. 2008. Parallel creation of gigaword corpora for medium density languages – an interim report. In *Proceedings of Language Resources and Evaluation Conference (LREC08)*, page to appear. European Language Resources Association.

Ian Jackson and Tony Finch. 2006. Gnu adns – advanced, easy to use, asynchronous-capable DNS client library and utilities.

Adam Kilgarriff. 2007. Googleology is bad science. *Computational Linguistics*, 33(1):147–151.

Philip Resnik and Aaron Elkiss. 2005. The linguist's search engine: an overview. In *ACL '05: Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 33–36, Morristown, NJ, USA. Association for Computational Linguistics.

Victor: the Web-Page Cleaning Tool

Miroslav Spousta, Michal Marek, Pavel Pecina

Institute of Formal and Applied Linguistics,
Charles University, Prague, Czech Republic
{spousta,marek,pecina}@ufal.mff.cuni.cz

Abstract

In this paper we present a complete solution for automatic cleaning of arbitrary HTML pages with a goal of using web data as a corpus in the area of natural language processing and computational linguistics. We employ a sequence-labeling approach based on Conditional Random Fields (CRF). Every block of text in analyzed web page is assigned a set of features extracted from the textual content and HTML structure of the page. The blocks are automatically labeled either as *content segments* containing main web page content, which should be preserved, or as *noisy segments* not suitable for further linguistic processing, which should be eliminated. Our solution is based on the tool introduced at the CLEANEVAL 2007 shared task workshop. In this paper, we present new CRF features, a handy annotation tool, and new evaluation metrics. Evaluation itself is performed on a random sample of web pages automatically downloaded from the Czech web domain.

1. Introduction

The idea of using “web as a corpus” has been very attractive for many researchers in computational linguistics, natural language processing, and related areas, who would really appreciate having access to such amount of data. The traditional way of building text corpora is a very expensive and time-consuming process and does not satisfy current requirements of modern methods. By automatic downloading of textual data directly from the web we can build extremely large corpus with relatively low cost and within short period of time.

Creating such a corpus comprises two steps: a) *web crawling* – automatic browsing the web and keeping a copy of visited pages and b) *cleaning* the pages to be included in the corpus. While there is a number of suitable web crawlers available (e.g. Heritrix[1], Holmes[2] or Egothor (Galamboš, 2006)), challenging task to clean up acquired web pages remains. Apart from main (linguistically valuable) content, a typical web page contains also material of no linguistic interest, such as navigation bars, panels and frames, page headers and footers, copyright and privacy notices, advertisements and other uninteresting data (often called *boilerplate*). The general goal is to detect and remove such parts from an arbitrary web page.

In this paper we describe a complete set of tools that enables transformation of a large number of web pages downloaded from the Internet into a corpus usable for NLP and computational linguistic research. The basis of our solution is the web-page cleaning tool first introduced at the CLEANEVAL 2007 shared task workshop (Marek et al., 2007). In order to approach structure of traditional corpora, we significantly modified the cleaning requirements and restricted the set of possible labels to *text* and *header* for *content segments* to be preserved and *other* for *noisy segments* to be eliminated.

First, we review the cleaning algorithm and its features, then we introduce an annotation tool developed for our purpose to prepare data for training and evaluation, and finally we present several experiments and their results. Our focus on the Czech language (mainly in the evaluation section) is induced by an intention to create a large Czech corpus, comparable to the largest corpora currently available. Needless to say, our tools are language independent and can be used for any language.

2. Related Work

Most of the work related to web page cleaning originated in the area of web mining and search engines, e.g. (Cooley et al., 1999) or (Lee et al., 2000). In (Bar-Yossef and Rajagopalan, 2002), a notion of pagelet determined by the number of hyperlinks in the HTML element is employed to segment a web page; pagelets whose frequency of hyperlinks exceeds a threshold are removed. (Lin and Ho, 2002) extract keywords from each block content to compute its entropy, and blocks with small entropy are identified and removed. In (Yi et al., 2003) and (Yi and Liu, 2003), a tree structure is introduced to capture the common presentation style of web pages and entropy of its elements is computed to determine which element should be removed. In (Chen et al., 2006), a two-stage web page cleaning method is proposed. First, web pages are segmented into blocks and blocks are clustered according to their style features. Second, the blocks with similar layout style and content are identified and deleted.

Many new approaches to web page cleaning were encouraged by the CLEANEVAL 2007 contest[3] organized by ACL Web as Corpus interest group. Competitors used heuristic rules as well as different machine learning methods, including Support Vector Machines (Bauer et al., 2007), decision trees, genetic algorithms and language models (Hofmann and Weerkamp, 2007). Although methods are fundamentally different, many of them employ similar set of mostly language-independent features such as average length of a sentence or ratio of capitalized words in a page segment.

[1]http://crawler.archive.org/
[2]http://www.ucw.cz/holmes/
[3]http://cleaneval.sigwac.org.uk/

3. Victor the Cleaner

3.1. System Overview

Our system for web page cleaning, first described in (Marek et al., 2007), is based on a sequence labeling algorithm with CRF++[4] implementation of Conditional Random Fields (Lafferty et al., 2001). It is aimed at cleaning arbitrary HTML pages by removing all text except headers and main page content. Continuous text sections (sections not including any HTML tags) are considered a single *block* that should be marked by a label as a whole.
The cleaning process consists of several steps:

1) Filtering invalid documents

Text from input documents is extracted and simple n-gram based classification is applied to filter out documents not in a target language (Czech in our case) as well as documents containing invalid characters (caused mainly by incorrect encoding specified in HTTP or HTML header).

2) Standardizing HTML code

The raw HTML input is passed through Tidy[5] in order to get a valid and parsable HTML tree. During development, we found only one significant problem with Tidy, namely interpreting JavaScript inside the `<script>` element, and employed a simple workaround for it in our system. Except for this particular problem which occurred only once in our training data, Tidy has proved to be a good choice.

3) Precleaning

Afterwards, the HTML code is parsed and parts that are guaranteed not to carry any useful text (e.g. scripts, style definitions, embedded objects, etc.) are removed from the HTML structure. The result is valid HTML code.

4) Text block identification

In this step, the precleaned HTML text is parsed again with a HTML parser and interpreted as a sequence of text *blocks* separated by one or more HTML tags. For example, the snippet “`<p>Hello <b>world</b>!</p>`” would be split into three blocks, “`Hello`”, “`world`”, and “`!`”. Each of the blocks is then a subject of the labeling task and cleaning.

5) Feature extraction

In this step, a feature vector is generated for each block. The list of features and their detailed description is presented in the next section. All features must have a finite set of values[6]. The mapping of integers and real numbers into finite sets was chosen empirically and is specified in the configuration. Most features are generated separately by independent modules. This allows for adding other features and switching between them for different tasks.

6) Learning

Each *block* occurring in our training data was manually assigned one of the following labels: *header*, *text* (*content blocks*) or *other* (*noisy blocks*).

[4]http://crfpp.sourceforge.net/

[5]http://tidy.sourceforge.net/

[6]This is a limitation of the CRF tool used.

The sequence of feature vectors including labels extracted for all blocks from the training data are then transformed into the actual features used for training the CRF model according to offset specification described in a template file.

7) Cleaning

Having estimated parameters of the CRF model, an arbitrary HTML file can be passed through steps 1–4, and its blocks can be labeled with the same set of labels as described above. These automatically assigned labels are then used to produce a cleaned output. Blocks labeled as *header* or *text* remain in the document, blocks labeled as *other* are deleted.

3.2. Feature Descriptions

Features recognized by the system can be divided by their scope into three subsets: features based on the HTML markup, features based on textual content of the blocks, and features related to the document.

Markup-based Features

container.p, container.a, container.u, container.img, container.class-header, container.class-bold, container.class-italic, container.class-list, container.class-form

For each parent element of a block, a corresponding *container.** feature will be set to 1, e.g. a hyperlink inside a paragraph will have the features *container.p* and *container.a* set to 1. This feature is especially useful for classifying blocks: For instance a block contained in one of the `<hx>` elements is likely to be a header, etc. The *container.class-** features refer to classes of similar elements rather than to elements themselves.

split.p, split.br, split.hr, split.class-inline, split.class-block

For each opening or closing tag encountered since the last block, we generate a corresponding *split.** feature. This is needed to decide, whether a given block connects to the text of the previous block (classified as *continuation*) or not. Also, the number of encountered tags of the same kind is recorded in the feature. This is mainly because of the `<br>` tag; a single line break does not usually split a paragraph, while two or more `<br>` tags usually do. The *split.class-** features again refer to classes of similar elements.

Content-based Features

char.alpha-rel, char.num-rel, char.punct-rel, char.white-rel, char.other-rel

These features represent the absolute and relative counts of characters of different classes (letters, digits, punctuation, whitespace and other) in the block.

token.alpha-rel, token.num-rel, token.mix-rel, token.other-rel token.alpha-abs, token.num-abs, token.mix-abs, token.other-abs

These features reflect counts distribution of individual classes of tokens[7]. The classes are words, numbers, mixture of letters and digits, and other.

[7]Tokens being sequences of characters separated by whitespace for this purpose.

sentence.count

Number of sentences in a block. We use a naive algorithm basically counting periods, exclamation marks and question marks, without trying to detect abbreviations. Given that the actual count is mapped into a small set of values anyway, this does not seem to be a problem.

sentence.avg-length

Average length of a sentence, in words.

sentence-begin, sentence-end

These identify text blocks that start or end a sentence. This helps recognizing headers (as these usually do not end with a period) as well as continuation blocks (*sentence-end=0* in the previous blocks and *sentence-start=0* in the current block suggest a continuation).

first-duplicate, duplicate-count

The *duplicate-count* feature counts the number of blocks with the same content (ignoring white space and non-letters). The first block of a group of twins is then marked with *first-duplicate*. This feature serves two purposes: On pages where valid text interleaves with noise (blogs, news frontpages, etc), the noise often consists of some phrases like “read more...”, “comments”, “permalink”, etc, that repeat multiple times on the page.

regexp.url, regexp.date, regexp.time

While we try to develop a tool that works independently of the human language of the text, some language-specific features are needed nevertheless. The configuration defines each *regexp.** feature as an array of regular expressions. The value of the feature is the number of the first matching expression (or zero for no match). We use two sets of regular expressions: to identify times and dates and URLs.

div-group.word-ratio, td-group.word-ratio

The layout of many web pages follows a similar pattern: the main content is enclosed in one big `<div>` or `<td>` element, as are the menu bars, advertisements etc. To recognize this feature and express it as a number, the parser groups blocks that are direct descendants of the same `<div>` element (`<td>` element respectively). A direct descendant in this context means that there is no other `<div>` element (`<td>` element respectively) in the tree hierarchy between the parent and the descendant. For example in this markup

```
<div> a <div> b c </div> d <div> e
f </div> g </div>
```

the *div-group*s would be (a, d, g), (b,c) and (e, f). The *div-group.word-ratio* and *td-group.word-ratio* express the relative size of the group in number of words. To better distinguish between groups with noise (e.g. menus) and groups with text, only words not enclosed in `<a>` tags are considered.

langid1, langid2

These new features represent a probability that a text block is in given language (Czech in our experiment). We used our own implementation of two Language ID approaches: (Beesley, 1988) and (Cavnar and Trenkle, 1994).

Document-related features

position

This feature reflects a relative position of the block in the document (counted in blocks, not bytes). The rationale behind this feature is that parts close to the beginning and the end of documents usually contain noise.

document.word-count, document.sentence-count, document.block-count

This feature represents the number of words, sentences and text blocks in the document.

document.max-div-group, document.max-td-group

The maximum over all *div-group.word-ratio* and a maximum over all *td-group.word-ratio* features. This allows us to express “fragmentation” of the document – documents with a low value of one of these features are composed of small chunks of text (e.g. web bulletin boards).

4. The Annotation Tool

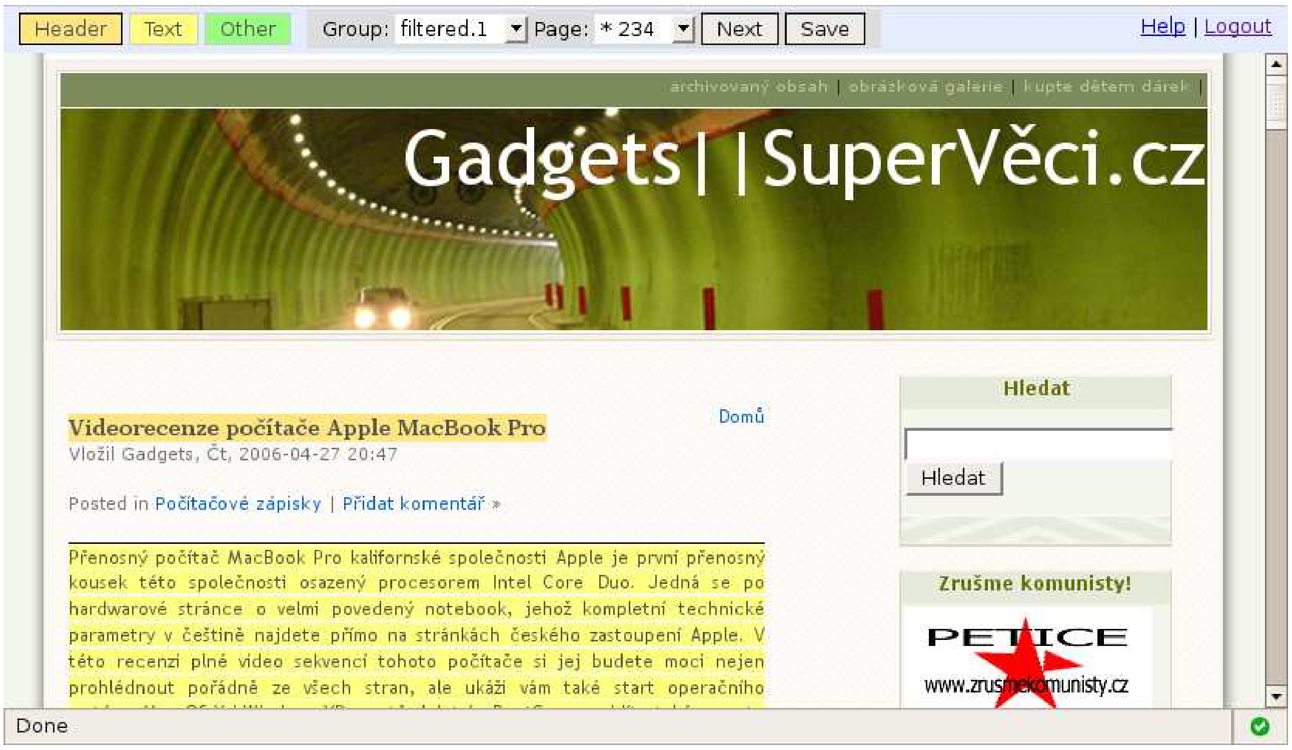
In order to enable fast and efficient annotation of the web page text blocks we developed a new annotation tool. Our aim was to offer a possibility to see the web page in a similar fashion to regular web browsing. This greatly simplifies the process of selection of the most important parts of given web page and distinguishing important text passages from other page sections.

Our annotation tool is a client–server based application using common web browser and JavaScript for the web page annotation on the client side and PHP based server application for serving pages to the client and storing current user annotation judgments.

The tool accepts either a list of HTML pages for annotation or a list of URLs to be downloaded and annotated. A simple pre-processing is applied for every web page before it can be annotated: all JavaScript is stripped and links are disabled so that annotator cannot accidently exit current web page.

The annotation process is quite straightforward (see Figure 1): user chooses a label selecting appropriate button and marks text blocks by clicking on the beginning and end of the text section to be marked. Different colors are used for every annotation label. Current annotation mark-up is stored on the server and can be easily retrieved and merged into the original HTML document when the annotation is finished.

We found that using a web browser for annotation significantly improves the annotation speed compared to using word processor or simple text based selection tool. The major speed up is due to the fact that not all the blocks must be



Videorecenze počítače Apple MacBook Pro
Vložil Gadgets, Čt, 2006-04-27 20:47

Posted in Počítačové zápisky | Přidat komentář »

Přenosný počítač MacBook Pro kalifornské společnosti Apple je první přenosný kousek této společnosti osazený procesorem Intel Core Duo. Jedná se po hardwarové stránce o velmi povedený notebook, jehož kompletní technické parametry v češtině najdete přímo na stránkách českého zastoupení Apple. V této recenzi plné video sekvencí tohoto počítače si jej budete moci nejen prohlédnout pořádně ze všech stran, ale ukáži vám také start operačního

Figure 1: The annotation tool: browser window is split into two parts: narrow upper frame is used for annotation control, lower frame contains the page to be annotated. Current annotation is shown using different colors for every label.

judged and annotated — remaining unannotated blocks are implicitly classified as *other*. Our volunteer annotator was able to achieve speed of 200 web pages per hour.

5. Evaluation

5.1. Data preparation

In order to perform the cleaning task, we have to train the Conditional Random Fields algorithm on the data from pre-annotated web pages. For training and evaluation purposes, we selected a random sample of 2 000 web pages from the data set downloaded using the Egothor search engine robot (Galamboš, 2006) from the Czech web domain (.cz).

Large proportion of downloaded pages contains only HTML mark-up with none or very small amount of textual information (root frames, image gallery pages, etc.). In addition, many pages use invalid character encoding or contain large passages in a language different from our target language. In order to exclude such pages from further processing, we apply a Language ID filter. Each page is assigned a value which can be interpreted as a probability of the page being in Czech. Pages not likely to be in Czech are discarded. We used our own implementation of Language ID methods by Beesley (1988) and Cavnar and Trenkle (1994). Out of 2 000 web pages, only 907 were accepted as reasonable documents containing non-trivial amount of Czech text.

All documents were annotated using our HTML annotation tool described in the previous section. We provided only short annotation guidelines, discouraging a markup of short, incomplete sections of the text (product descriptions, lists of items, discussions) and only marking headlines belonging to already selected text sections. All non-annotated text blocks are considered to be labeled as *other*.

According to the annotation, only 271 (29.9%) documents contained text blocks with useful content (text and headers) to be preserved. Complete overview of the label distribution can be found in Table 1.

label	*count*	%
header	1 009	1.14
text	5 571	6.32
other	81 528	92.53
total	88 108	100.00

Table 1: Label distribution in the development data set.

5.2. Experiments and Results

Following our experience from the CLEANEVAL 2007 we found that computation of Levensthein algorithm for evaluation of cleaning results is usually very expensive. Our approach of labeling consequent text blocks suggests an *accuracy* as a measure of success for our task – the ratio of correctly assigned labels. If we do not want differentiate between blocks labeled as *text* or *header* (they are equally good for our purposes and we would like them to be included in our corpus) we can use also *unlabeled accuracy*.

In our first experiment we used 271 manually annotated pages containing at least one content block (labeled either as *text* or *header*). Running a 10-fold cross-evaluation on such data we were able to achieve accuracy of 91.13% and unlabeled accuracy of 92.23%. This number, however, does not tell much about quality of cleaning because of the discrepancy in proportion of content (*text*, *header*) and noise (*other*) blocks in our data (see table 1) which could be expected also in real pages downloaded from the web. A trivial algorithm assigning *other* label to all blocks performs with accuracy even higher (92.53)%.

Precision and Recall

In such cases, using *precision* and *recall* measures would be more appropriate. We do not differentiate between blocks labeled as *text* or *header* (they are equally good for our purposes and we would like them to be included in our corpus) and define *precision* and *recall* as follows:

$$Precision = \frac{TH_{correct}}{TH_{labeled}}$$

$$Recall = \frac{TH_{correct}}{TH_{annotated}}$$

where $TH_{correct}$ refers to a number of correctly labeled content blocks [8], $TH_{labeled}$ is the total number of labeled content blocks and $TH_{annotated}$ is the total number of all blocks annotated as content blocks (*text* or *header*).
Precision and recall scores of our first experiment are shown in the first column of the table bellow. We can expect the pages to be cleaned with 80.75% precision and 79.88% recall, i.e. 19.25% of blocks in the cleaned data are noise and we miss 21.12% of content blocks that should be preserved.

using LangID	no	yes
Text/Header/Other		
Accuracy	91.13	90.82
Text+Header/Other		
Accuracy	92.23	91.84
Precision	80.75	83.80
Recall	79.88	72.95

Table 2: Effect of Language ID features.

Language ID features

In the next experiment we evaluated the Language ID features newly used by the CRF component of our system and representing probability that a text block is in given language (see section 3.2.). As it can be seen in the second column of Table 2, using these features we were able to increase the precision up to 83.8%.

Balancing Precision and Recall

The huge number of texts available on the web even for relatively rare languages such as Czech, enables us to focus on acquisition of high quality data only. In other words, we prefer high-precision cleaning procedure to the high-recall one.
While CRF algorithm does not offer a direct method to fine-tune precision and recall trade-off, we propose an alternative approach to achieve this. For every block, it is possible to obtain marginal probability assigned to all possible labels. In common sequence-labeling scenario, the label with highest probability wins, not matter what is a distribution of other labels' probabilities. In order to achieve higher precision, we only allow *text* and *header* labels to win if probability of *other* label is under given threshold. Figure 2 illustrates, that we are really able to achieve arbitrary precision by giving preference to the *other* label.

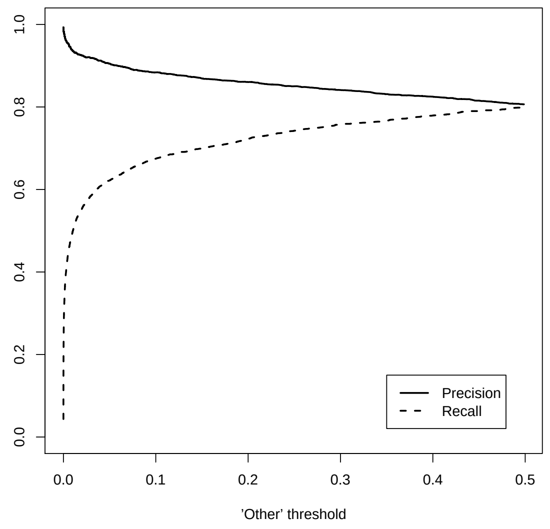


Figure 2: Precision and recall graph obtained by setting the threshold value of *other* label in the interval [0, 0.5]. Default value of the threshold is 0.5.

training data size (documents)	271	907
Text/Header/Other		
Accuracy	91.13	95.92
Text+Header/Other		
Accuracy	92.23	96.15
Precision	80.75	74.78
Recall	79.88	66.95

Table 3: Results of 10-fold cross-evaluation on 271 annotated documents (containing at least one block marked as *text* or *header*) and entire set of 907 annotated documents.

The last experiment we performed was a comparison of two systems: the one as in the first experiment trained on the 271 manually annotated pages containing at least one content block (labeled either as *text* or *header*) and the other (more real) one trained on all 907 manually annotated pages that passed the Language ID test. The performance of these two systems can be compared only in terms of precision and recall (the number of content blocks remains the same, but number of the noise blocks is much higher for the latter system). As it is shown in Table 3 the additional low quality data added to the second systems significantly hurt its performance. Precision dropped from 80.75% to 74.78% and recall from 79.88% to 66.95%. We can conclude that the precleaning step where the low-quality web pages (naviga-

[8]*Text* blocks mislabeled as *header* and vice versa are counted too.

tional, image-only, etc.) are removed completely, should be improved.

The speed of the cleaning tool is about 3.5 pages (80 kB) per second. Approximate time for cleaning entire web data set we have (30 million web pages) is 10 days on 10 common CPU cores. We didn't perform this task yet, though.

6. Conclusion and Further Work

We presented a complete solution for cleaning the web pages content, including annotation tool and evaluation metrics. However, this is still an ongoing work and we will continue in research of this challenging task. Current versions of our tools are available for download at `http://ufal.mff.cuni.cz/victor/`.

In the near future, we would like to focus also on real applications – to compare traditional corpora with our web-based corpus using a task that requires large textual data. For Czech, we propose an experiment to compare performance of Czech part-of-speech tagger trained using unsupervised training (Spoustová, 2008) on the data obtained from the cleaned web pages and data from the Czech National Corpus (Institute of Czech National Corpus, 2005).

Acknowledgments

This work has been supported by the Ministry of Education of the Czech Republic, project MSM 0021620838 and the Czech Science Foundation, project 201/05/H014.

We would like to thank Dr. Leo Galamboš for allowing us to obtain the web data and our anonyMouse volunteer annotators for their effort.

7. References

Ziv Bar-Yossef and Sridhar Rajagopalan. 2002. Template detection via data mining and its applications. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*.

Daniel Bauer, Judith Degen, Xiaoye Deng, Priska Herger, Jan Gasthaus, Eugenie Giesbrecht, Lina Jansen, Christin Kalina, Thorben Krüger, Robert Märtin, Martin Schmidt, Simon Scholler, Johannes Steger, Egon Stemle, and Stefan Evert. 2007. Fiasco: Filtering the internet by automatic subtree classification. In *Proceedings of the Web as Corpus Workshop (WAC3), Cleaneval Session*, Louvain-la-Neuve, Belgium.

K. Beesley. 1988. Language identifier: A computer program for automatic natural-language identification on on-line text.

William B. Cavnar and John M. Trenkle. 1994. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, Las Vegas, US.

Liang Chen, Shaozhi Ye, and Xing Li. 2006. Template detection for large scale search engines. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, Dijon, France.

Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. 1999. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*.

Leo Galamboš. 2006. Egothor, full-featured text search engine written entirely in java. http://www.egothor.org/.

Katja Hofmann and Wouter Weerkamp. 2007. Web corpus cleaning using content and structure. In *Proceedings of the Web as Corpus Workshop (WAC3), Cleaneval Session*, Louvain-la-Neuve, Belgium.

Institute of Czech National Corpus. 2005. Český národní korpus – SYN2005. http://ucnk.ff.cuni.cz/.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, USA.

Mong-Li Lee, Tok Wang Ling, and Wai Lup Low. 2000. Intelliclean: a knowledge-based intelligent data cleaner. In *Knowledge Discovery and Data Mining*.

Shian-Hua Lin and Jan-Ming Ho. 2002. Discovering informative content blocks from web documents. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, Canada.

Michal Marek, Pavel Pecina, and Miroslav Spousta. 2007. Web page cleaning with conditional random fields. In *Proceedings of the Web as Corpus Workshop (WAC3), Cleaneval Session*, Louvain-la-Neuve, Belgium.

Drahomíra "johanka" Spoustová. 2008. Combining statistical and rule-based approaches to morphological tagging of czech texts. *To appear in Prague Bulletin of Mathematical Linguistics*, 89.

Lan Yi and Bing Liu. 2003. Web page cleaning for web mining through feature weighting. In *Proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*, Acapulco, Mexico.

Lan Yi, Bing Liu, and Xiaoli Li. 2003. Eliminating noisy information in web pages for data mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, Washington, DC, USA.

Segmenting HTML pages using visual and semantic information

Georgios Petasis, Pavlina Fragkou, Aris Theodorakos, Vangelis Karkaletsis, Constantine D. Spyropoulos

Software and Knowledge Engineering Laboratory,

Institute of Informatics and Telecommunications,

National Centre for Scientific Research (N.C.S.R.) “Demokritos”,

P.O. BOX 60228, Aghia Paraskevi,

GR-153 10, Athens, Greece.

e-mail: {petasis, fragou, artheo, vangelis, costass}@iit.demokritos.gr

Abstract

The information explosion of the Web aggravates the problem of effective information retrieval. Even though linguistic approaches found in the literature perform linguistic annotation by creating metadata in the form of tokens, lemmas or part of speech tags, however, this process is insufficient. This is due to the fact that these linguistic metadata do not exploit the actual content of the page, leading to the need of performing semantic annotation based on a predefined semantic model. This paper proposes a new learning approach for performing automatic semantic annotation. This is the result of a two step procedure: the first step partitions a web page into blocks based on its visual layout, while the second, performs subsequent partitioning based on the examination of appearance of specific types of entities denoting the semantic category as well as the application of a number of simple heuristics. Preliminary experiments performed on a manually annotated corpus regarding athletics proved to be very promising.

1. Introduction

Nowadays, the vast amount of information available in the Web remains in a considerable degree an unexploited thesaurus of knowledge resources. This is due to the fact that, even though proposed methods for performing fast search, clever ranking, data analysis and web page indexing proved to be highly effective, the actual content of the web pages is very poorly exploited. In order to face this challenge, a number of approaches have been proposed, which try to attach metadata to web pages, annotating them with linguistic or even semantic information. Linguistic metadata typically involve the identification of tokens, part of speech tags, stems, lemmas and other syntactic information. However, these metadata are not enough if an in depth analysis of the content is required, in order to interpret and extract the facts and events described by the content. Effective content processing requires the presence of semantic metadata, which can be extremely useful for the efficient retrieval, extraction and interpretation of the information contained into a web page. In addition, it may help in overcoming problems such as search engine duplicate results elimination, efficient query refinement and high precision answering. Being a resource demanding and time consuming process, semantic annotation needs to be automated as much as possible through the automatic acquisition of semantic metadata according to a predefined semantic model.

An important subtask of semantic annotation is the semantic segmentation. Semantic segmentation can be defined as the task of identifying parts of a web page that refer to the same fact/event (topic), as defined by a semantic model. For example, web pages of a news portal may contain multiple news items on a single page, while pages from an electronic shop can contain multiple product descriptions per page. Semantic segmentation must disambiguate these multiple instances into areas describing a single item, by extracting and subsequently annotating the “semantic” structure of a web page with predefined categories (such as “news item” or “product”) according to a predefined semantic model. Various subtasks of information extraction can benefit from the results of semantic segmentation. Tasks like named entity recognition may benefit less, as they usually can be performed on a web page as a whole. However task like co-reference resolution and relation extraction can benefit more, as they can be applied in more confined areas that are semantically “coherent”, possibly limiting erroneous relations and thus increasing performance.

In this paper we present a novel method which performs automatic semantic segmentation on web pages. Most specifically, our method partitions the text on a web page into blocks, which refer to a single topic, and annotates identified blocks with predefined, domain specific, categories. The partitioning is based not only on the visual layout of a web page but also on information related to the distribution of named entity instances in the page. Starting from a detailed visual segmentation, the proposed method follows a two staged approach. During the first stage, a set of heuristics is applied to transform the visual segmentation into an initial semantic segmentation. This initial semantic segmentation is refined during the second stage by exploiting the distribution of named entities, with the help of a machine learning algorithm.

Preliminary results obtained from a manually annotated corpus about athletics, show that the segmentation performed by using information from named entities clearly outperform the segmentation that exploits only visual information, even if visual information is combined with heuristics, in order to increase its performance. The exploitation of the distribution of named entities, which is one of the main innovative aspects of the presented approach, appears to significantly improve segmentation performance over heuristics that are frequently employed for the task. In addition, the elimination of the heuristics lead to a more adaptable system, as usually these heuristics are domain specific and quite often site specific. Finally, the importance of our work lies on the fact that it can be easily applied to a number of web-driven applications such as search engines, web-based question answering, web-based data mining as well as voice enabled web navigation.

The structure of the present paper is as follows. Section 2 provides an overview of related work found in the literature, Section 3 provides a description of the research project under which the aforementioned work took place as well as a detailed description of our proposed method, Section 4 provides experimental results while Section 5 provides conclusion results as well as future steps.

2. Related Work

Research approaches found in the literature use a variety of methods to accomplish the identification of the structure of a web page i.e. page segmentation in order to perform automatic semantic segmentation. A family of methods perform automatic extraction of segments i.e. blocks by defining appropriate patterns or grammars that contain the target data in web pages usually conforming to a common schema and repeating pattern mining or adopting pattern matching and string alignment techniques to refine extraction rules (Zhai and Liu, 2005; Zhai and Liu, 2006. Crescenzi et al, 2001; Crescenzi et al., 2002; Arasu and Garcia-Molina, 2003; Chang and Kuo, 2004; Chang et al., 2003). Such methods usually require (labeled or unlabelled) training examples. A second family of methods is based on machine learning requiring human labeling from each Web site that is interested in extracting data from (Kushmerick, 2000; Hsu and Dung, 1998; Muslea et al., 1999). Recent approaches (Wu et al.,2006; He et al., 2005; Song et al., 2004; Feng et al., 2005; Yang et al., 2006; Wang and Richard, 2007) also exploit the visual layout of a web page by looking in more depth in the actual structure of a web page. Such approaches are based on the observation that, blocks of interest usually appear in the central-main region of the web page and most specifically exhibit similar visual appearance. The exploitation of the structure of a web page is either performed in a "flat" manner i.e. by searching for repeated sequences of html tags or in a "hierarchical" manner by exploiting the DOM tree [1]or tag tree of the page. Blocks of interest are retrieved by exploiting the nodes of the tree and by applying heuristic rules, tree pattern matching, by examining the geometrical layout of the leaf nodes, linguistic features, spatial features (such as the position and size) and content features (such as the number of images and links) that are used to construct a feature vector for each block. The feature vector of each block is in his turn being used to train a model to assign importance to each block in the web page. Alternative approaches exploit the attributes of the DOM tree. Those approaches try to attribute a block importance value based on the existence of block features, spatial cues as well as on heuristic rules. They alternatively try to attribute a block importance value by defining content splitting delimiters that are applied in the DOM tree in order to create block nodes that are refined based on cue phrases appearing in the anchor text.

One among the most promising approaches is the one implemented by the VIPS algorithm (Cai et al., 2003; www.ews.uiuc.edu/~dengcai2/VIPS/VIPS.html) which tries to segment a web page by exploiting its visual layout after its rendering in a web browser. Using the visual information stored by the web browser into the DOM tree, VIPS identifies the segments that cover large areas of the rendered web page and then tries to find from the visual representation of the page horizontal or vertical lines that separate these segments. Separators denote the horizontal or vertical lines in a web page without visually crossing with blocks. The vision-based content structure of the page is constructed by recursively segmenting the already processed segments, while the granularity of segmentation can be controlled through a threshold that represents the coherence of the segment's content. A variation of the VIPS algorithm introduce an additional criterion i.e. the Degree of Isolation (Li et al., 2005) to describe the difference between a block and its other siblings and use language modeling to estimate a model for each block.

3. Method

The majority of existing algorithms works well in recognizing the visual layout of a web page. However, this proves to be insufficient for the semantic annotation, which requires semantic segmentation. Such produced metadata may be useful to create an effective representation of a web page to assist in tasks such as information retrieval, query refinement and information extraction. This is due to the fact that, for example, the information extraction's purpose is to identify segments of text containing semantic information, i.e. instances of named entities and relations. The identification of such instances and relations is more effective when it is performed into page segments describing a single event or "topic" instead of the whole page.

The method presented here goes a step further by making use of non-visual information for improving web page segmentation in order to be more compatible with semantic metadata information. More specifically we present a method consisting of two stages, which can act independently, in order to segment HTML web pages using visual and semantic information. As a first step, the VIPS visual segmentation algorithm is applied in order to visually segment each page as thorough as possible. With the help of heuristics applied on the results of visual segmentation, basic building blocks such as columns, paragraphs, headings, tables, headers and footers are identified.

As a subsequent step, the distribution of named entities in the text of a web page is exploited, in order to semantically segment the page. A named entity recognizer that can handle the domain of the web page is applied on the page. Then, with the use of machine learning, entities that can be attributed to refer to the same topic, are

[1] The Document Object Model (DOM) is an application programming interface (API) for valid HTML and well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated.

grouped together. These groups form the basis for semantically segmenting the page, by classifying basic building blocks obtained during visually based segmentation that contain an entity group into the same semantic segment. Grouping is achieved by capturing the relations between named entity instances, as those are defined by a semantic model. Grouping is performed by the use of a machine learning algorithm which takes as input a web page and creates a representation containing special and proximity relations between named entity instances appearing in parts of texts. Such relations are not captured by single repetition of the same value if named entity instances. Relation instances of interest are identified by searching for repetition of specific types of named entity instances with respect to other ones. New topics may be also be identified by novel instances of predefined types of named entities. The result of the implementation of the machine learning algorithm on a web page is its segmentation into “semantic segments” according to the BOEMIE semantic model.

3.1 Visual segmentation with the use of heuristics

The first stage of our method, which is based on heuristics operating on the results of visual segmentation, tries to identify a web page’s content structure by exploiting spatial attributes such as relevant position or size. The VIPS algorithm is applied, in order to perform a thorough segmentation based solely on visual information. The segmentation detail is controlled through the granularity parameter of VIPS, which controls the coherence of identified segments based on visual perception. By using the maximum granularity value, this process aims to identify the smallest possible visual segments, which will be concatenated through simple heuristics to form the basic building blocks, such as headers, footers, columns, headings, paragraphs, tables, images and captions. VIPS uses placement and size information, placed by a web browser in the DOM tree nodes that correspond to visual structures, in order to annotate the DOM tree with segment information. From the resulting DOM tree, our method tries to identify the “main” or “central” region of a page, usually denoted by the wider column whose length occupies a significant percentage of the total page length. In the majority of cases this column is represented by the <td> HTML tag, but in cases where this assumption fails, the DOM tree node having the greatest text length (in characters) is used instead. After detecting the main region, the leaves of the sub-tree of the main region node that contain at least some text are examined in order to detect titles and paragraphs. The assumption is that such nodes have maximum coherence since they were retrieved at the final level of VIPS segmentation. Leaves which correspond to image captions or contain linked text only are omitted in order to result in a set of nodes appearing in the main region and contain only text i.e. nodes that are either titles or paragraphs. In order to identify such cases the following heuristics were applied: (a) if the text of the node is up to 5 words, it is considered as a title (b) if it contains more than 20 words it is considered as a paragraph (c) if it contains a number of words in the range from 6 to 20 words, it is considered as a paragraph in case it ends with a dot, otherwise it is considered as a title. The procedure finishes after all possible nodes have been parsed, while the output consists of the identified titles and paragraphs.

The choice of the aforementioned heuristics is based on results obtained from the observation of web pages contained in the corpus at our disposal regarding athletics, as well as the statistical calculation of the number of words appearing in titles and paragraphs in each of those pages.

3.2 Transforming visual segments into semantic segments

While the first stage of our method focuses on the exploitation of spatial attributes, the second one is focused on identifying segments which describe the same topic, with respect to the thematic domain of a web page. For example, in pages containing news about athletic events, the semantic model of the domain may dictate the recognition of segments describing a single sport competition, in a page describing all sport competitions that took place in a specific event date. Topic segmentation is accomplished by parsing titles and paragraphs in order to identify changes in the frequency of appearance and distribution of specific types of named entities, adopting a machine learning approach. The assumption here is that the presence of specific types of entity instances as well as the repetition of some of them may prove to be determinative for the presence of the same topic. Consequently, the change of the instance values of specific types of entities may signal change of topic. Spatial and proximity information of named entity instances play a major role to the signaling of the presence or change of the same topic.

In order to perform segmentation, a semi-supervised machine learning approach is followed, which, given an unknown web page, it tries to identify portions of texts that refer to a single topic. This identification is based on an already learned model. To construct such as model, a number of manually annotated web pages with the desired semantic segments are used to form a training corpus. Each page belonging to the training corpus is processed in order to form one or more feature vectors each of which represents a different topic. A feature vector is constructed for every title or paragraph containing those features that may signal the existence or not of the same topic. The feature vector contains information regarding the type of the node and the frequency of occurrence of named entities in the node. While all identified types of named entities can be used in the vector, a reduction of the used types is also possible, if some types can be identified as being more important than others in advance. For example in the athletics domain, if topic is assumed to be news about sport competitions, entities like sport names, athlete names and gender of athletes can be more significant for the task, than entities like locations or nationalities. Each feature vector also contains contextual information, by containing the equivalent information found into previous node(s). The feature vector finally contains the number of common named entity instance values with those of the previous node(s).

3.3 Case study: the BOEMIE R&D project

The proposed semantic segmentation method has been applied in the context of the BOEMIE (Bootstrapping Ontology Evolution with Multimedia Information Extraction) R&D project (http://www.boemie.org).

BOEMIE implements ontology-based information extraction systems which extract metadata information compliant to an ontology from multimedia content regarding athletic events. Driven by domain-specific multimedia ontologies, the BOEMIE information extraction system is able to identify high-level semantic features in images, videos, audios and texts, and then fuse modality specific extracted information to interpret multimodal resources.

When dealing with web pages about news, usually obtained from web sites of official associations like IAAF, topic detection mainly concerns the detection of news items describing a single sport. A news item is usually represented by an (optional) title, followed by one or more paragraphs that refer to a specific sport, taking place during either a specific event or during multiple events. News-item detection is performed by parsing the titles and paragraphs in order to identify appropriate units. Units are usually portions of a web page describing the athlete's participating in a specific sport during a specific event or the athlete's participating in more than one event when the sport in question remains the same.

The named entities defined in the semantic model can be grouped as follows: (a) athlete_name, age, gender, nationality (corresponding to the information regarding an athlete), (b) performance and ranking (of an athlete in a specific round of a specific sport), (c) round_name and heat_name (corresponding to a specific round with or without the information of the heat of a specific sport) (d) sport_name, (corresponding to a specific sport), (e) event_name, country (corresponding to a specific event and country in which the event takes place) (f) city, date (corresponding to a specific city or date in which a specific sport or event takes place). Our assumption is that, when during the text, a change of an instance of a named entity such as the sport name, the athlete name and/or the gender takes place, this change may signal the appearance of a new news item. Figure 1 depicts a part of the domain ontology for the text modality in BOEMIE.

3.3.1 Extracting semantic segments

In order to adapt the proposed method to the domain of athletics, we have decided to limit the entity types involved to occurrences of named entities with types sport name, athlete name and gender. The selection of those types was based on the following observations: (a) the most frequent information found in paragraphs and titles belonging to a news item is the athlete and sport named entities, (b) paragraphs constituting a news item usually describe the progress of a competition, thus contain a number of common sport and athlete name instances, (c) the gender information signals the description of men or women competitions, which are considered as different sports and thus news items (topics), (d) the information of performance can only act indirectly to the identification of a news item (i.e. through the range of observed values), (e) change on the sport name usually corresponds to a different sport competition.

It is worth mentioning that our feature vector takes advantage of the actual names of the named entities of interest. Thus, it focuses on the change of values of specific types of named entities but not on the change of the types on named entities used.

From the portion of a webpage which is depicted in figure 2, the following feature vectors are produced from examining its nodes

- **Title:** Title , athlete_instance_1 = "Richards", #athlete_instances = 1
- **Paragraph 1:** Paragraph, sport_instance_1 = "400m", athlete_instance_2 = "Sanya Richards", gender_instance_1 = "women", #athlete_instances = 1, #sport_instances = 1, #gender_instances = 1, *common_information_with_title* = (athlete_instance_1, #=1)
- **Paragraph 2:** Paragraph, athlete_instance_3 = "Richards", #athlete_instances = 1, *common_information_with_title* = (athlete_instance_1, #=1), *common_information_with_previous_paragraphs* = (athlete_instance_2, #= 1)
- **Paragraph 3:** Paragraph, athlete_instance_4 = "Richards", #athlete_instances = 1, *common_information_with_title* = (athlete_instance_1, #=1), *common_information_with_previous_paragraphs* = (athlete_instance_2, athlete_instance_3, #= 2)
- **Paragraph 4:** Paragraph, athlete_instance_5 = "Lashawn Merritt", athlete_instance_6 = "Angelo Taylor", sport_instance_2 = "400m"', sport_instance_3 = "400 Hurdles", gender_instance_2 = "men, #athlete_instances = 2, #sport_instances = 2, #gender_instances = 1, *common_information_with_title* = (nothing),*common_information_with_previous_paragraphs* = (sport_instance_1, #= 1)

In order to learn a model for semantically segmenting a web page, the CRF++ algorithm (http://crfpp.sourceforge.net/) was chosen, due the fact that it has been proved to be a very effective framework for building probabilistic models to segment and label sequential data (Sha and Pereira, 2003). CRF++ is a simple, customizable, and open source implementation of Conditional Random Fields (CRFs) for segmenting/labeling sequential data. CRF++ has been designed for generic use, and has been applied on a variety of NLP tasks, such as Named Entity Recognition, Relation Extraction and Text Chunking.

4. Experiments

The experiments of our algorithm were performed on a manually annotated corpus. This is due to the fact that a suitable benchmark corpus for this specific scientific area is not publicly available. The collection of the corpus, as well as the way documents were annotated are described in the following subsections.

4.1 The dataset

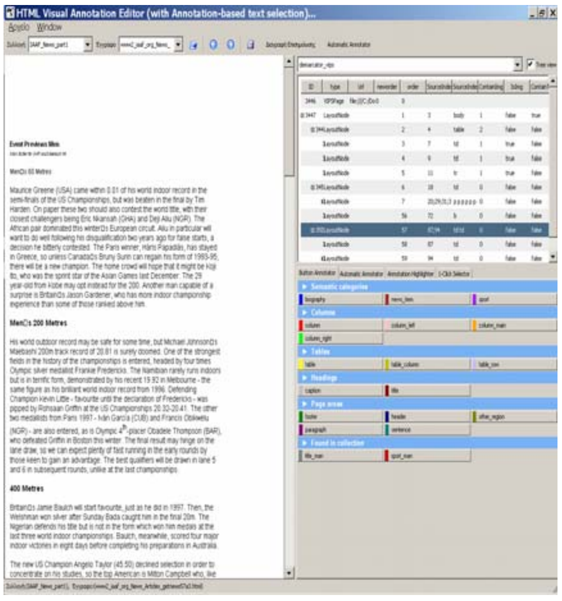
In order to evaluate our method we collected a corpus which contains 100 web pages taken from eight different web sites, which are the following:

- IAAF (http://www.iaaf.org) from the www.iaaf.org site, pages containing news regarding athletic events were collected, where both text and images may appear within it. (37 web pages)

- USA Track & Field (http://www.usatf.org/), pages containing news regarding athletic events, where both text and image may appear within it were collected. (28 web pages)
- news.bbc.co.uk/, (12 web pages)
- http://www.sportinglife.com/, (10 web pages)
- http://www.scc-events.com/ (4 web pages)
- http://sportsofworld.com/. (9 web pages)

In all web pages manually annotation of the boundaries of the blocks of interest i.e. the “news items” was performed using an annotation facility provided by the Ellogon platform (http://www.ellogon.org). More specifically, the annotation tool of Ellogon was appropriately configured in order to annotate the following entities:

- Semantic categories: news item, sport
- Columns: column left, column main, column right
- Tables: table, table column, table row
- Headings: caption, title
- Page areas: footer, header, paragraph, sentence, other region
- Semantic model entities: athlete name, age, gender, nationality, performance, ranking, round name, heat name, city, date, country, sport name, stadium name and event name which correspond to the BOEMIE semantic model. It is worth mentioning that overlapping annotation may exist within a web page. A screenshot of the annotation tool is given in Figure 3.



Event Previews Men

Men's 60 Metres

Maurice Greene (USA) came within 0.01 of his world indoor record in the semi-finals of the US Championships, but was beaten in the final by Tim Harden. On paper these two should also contest the world title, with their closest challengers being Eric Nkansah (GHA) and Deji Aliu (NGR). The African pair dominated this winter's European circuit. Aliu in particular will want to do well following his disqualification two years ago for false starts, a decision he bitterly contested. The Paris winner, Haris Papadias, has stayed in Greece, so unless Canada's Bruny Surin can regain his form of 1993-95, there will be a new champion. The home crowd will hope that it might be Koji Ito, who was the sprint star of the Asian Games last December. The 29 year-old from Kobe may opt instead for the 200. Another man capable of a surprise is Britain's Jason Gardener, who has more indoor championship experience than some of those ranked above him.

Men's 200 Metres

His world outdoor record may be safe for some time, but Michael Johnson's Maebashi 200m track record of 20.81 is surely doomed. One of the strongest fields in the history of the championships is entered, headed by four times Olympic silver medallist Frankie Fredericks. The Namibian rarely runs indoors but is in terrific form, demonstrated by his recent 19.92 in Melbourne - the same figure as his brilliant world indoor record from 1996. Defending Champion Kevin Little - favourite until the declaration of Fredericks - was pipped by Rohsaan Griffin at the US Championships 20.33-20.41. The other two medallists from Paris 1997 - Iván García (CUB) and Francis Obikwelu (NGR) - are also entered, as is Olympic 4th-placer Obadele Thompson (BAR), who defeated Griffin in Boston this winter. The final result may hinge on the lane draw, so we can expect plenty of fast running in the early rounds by those keen to gain an advantage. The best qualifiers will be drawn in lane 5 and 6 in subsequent rounds, unlike at the last championships.

400 Metres

Britain's Jamie Baulch will start favourite, just as he did in 1997. Then, the Welshman won silver after Sunday Bada caught him in the final 20m. The Nigerian defends his title but is not in the form which won him medals at the last three world indoor championships. Baulch, meanwhile, scored four major indoor victories in eight days before completing his preparations in Australia.

The new US Champion Angelo Taylor (45.50) declined selection in order to concentrate on his studies, so the top American is Milton Campbell who, like

Figure 3: Annotation of a single web page

4.3 Evaluation

In order to evaluate the proposed method, we measured its performance on the task of news item identification, in terms of the Precision, Recall and F-measure metrics. More specifically, precision is defined as “the number of the estimated news items which are actual news items” divided by “the number of the estimated news items, returned by the method”. Recall is defined as “the number of the estimated news items which are actual news items” divided by “the number of the true news items”. F-measure was defined as the double of the product of precision and recall divided by their corresponding sum. More specifically, we examine the correct assignment of boundaries of our algorithm with those appearing in the gold corpus with respect to the paragraph’s sequence of appearance within the page.

Two different approaches were examined in order to evaluate the correct identification of news items. The first approach applies a simple heuristic on top of the heuristics described in Section 3.1. Making the assumption that news items usually consist of a title followed by one or more paragraphs, the heuristic approach uses the identified information about paragraphs and titles, to form news items. Each identified title is merged with all paragraphs until the next title (or the end of the page) to form a news item. Thus, changes in news items are signaled only by the presence of a new title.

The second approach consists of the application of the machine learning algorithm described in Section 3.2, on top of the heuristics described in Section 3.1. A five fold cross validation was performed on the machine learning approach. Both the heuristics as well as the combined approach (heuristics and machine learning algorithm) were applied on the web pages DOM trees produced by VIPS, in order to evaluate the correct identification of news items. The results obtained by both evaluations are listed in the table below.

	Only heuristics	**Heuristics & machine learning algorithm**
Precision	40.83%	71.19%
Recall	18.64%	70.87%
F-measure	25.59%	71.01%

Table 1: Evaluation results in news item detection with heuristics and combination of heuristics and machine learning

From the obtained results we can see that the mere application of heuristics is not enough for the detection of news items. This can be attributed to several facts. The most important one is that the heuristics chosen proved to be too general (not domain specific because they do not exploit the presence of named entity instances that appear within them which proves to be the “key” for detecting news items) and able to capture a small portion of cases, the majority of those belonging to the IAAF site. A second important factor is the assumption that a news item consists of a title and a number of paragraphs below it proves to be true in few cases. This is due to the fact that, the paragraphs appearing below a title practically prove to belong to more than one news items. The low performance of the heuristics can also be attributed to the variety of web pages layout. Web pages layout proves to strongly differ from site to site but also from page to page within the same site.

The use of semantic features via machine learning proves to significantly improve news item detection. This can be attributed to several factors: (a) the high efficiency of the CRF algorithm in segmenting and labeling sequential data (b) the choice of the types of the named entities chosen to form the feature vector. This is due to the fact that the sport name entity in web pages containing news is a strong indicator of the appearance of a novel news item, although it tends to appear only once inside a news item, and not necessarily in the first paragraph of a news item. The same also holds for the appearance of novel athlete name instances which indicate that the subject in question refers to a different sport or event. Finally the change of gender name instance also signals the change of topic, which is indirectly assisted by the change of athlete name instances.

It is worth mentioning that, news item detection in web pages containing news must be dealt as a difficult problem. This can be attributed to several factors. The first and probably most important reason is the fact that those pages have quite different layouts. More specifically, pages belonging to this category may either contain only plain text i.e. paragraphs without section titles, or containing sections accompanied with section titles. In the first case the news item detection is defined as the problem of finding the appropriate number of paragraphs that corresponds to a news item. In the second case, the text portion appearing between two section titles may actually belong to more than one news items due to the fact that the description of more than one sport for the same event, or the description in more than one event for the same sport may take place. The aforementioned situation is aggravated by the fact that it is difficult even for a domain expert to decide upon a single topic for such cases. For example, there are documents regarding athletics where in their beginning, the “highlights” of more than one sport (along with their top-scoring athletes) are described in the same paragraph, making impossible to classify this paragraph with a single topic.

5. Conclusions – Future Work

In the current paper, we presented a novel approach to the problem of automatic semantic segmentation, which tries to automatically identify portions of text within a web page corresponding to semantic categories, in order to create semantic annotation to web pages. The produced semantic annotation corresponds to semantic segments, each of which captures a single topic i.e. “news” item, where named entity instances related to the topic are contained. This type of segmentation provides an overview of the content of a web page, with respect to the semantic model in use. The innovation of our method lies in the fact that it takes advantage of the semantic information, as expressed by the named entity instances, and combines it with visual layout information to perform semantic segmentation.

The examination of the results produced, especially concerning the second stage of our method, lead to the conclusion that the use of specific types of names such as sport and athlete names for web pages describing athletic events proved to be beneficial. A significant property of our method is that it can be applied to additional topic areas, possibly significantly different from athletics, as far as a semantic model is appropriately defined.

As future work, we plan to apply our algorithm to a larger corpus of web pages, exhibiting even greater layout variance, possibly originating from a larger variety of web sites, by exploiting better the large corpus collected in the context of the BOEMIE project. In order to accomplish this as well as to higher scalability regarding the domain in use, we plan to combine our method with others used for topic change i.e. text segmentation such as those of (Kehagias, Fragkou and Petridis, 2004) and (Utiyama and Isahara, 2001). Text segmentation methods benefit from the fact that they are domain independent and they are based on the statistical distribution of words within paragraphs or sentences. Those methods can be used as a preprocessing step. Alternatively, they can ground the calculation of their statistical distributions on specific types of words i.e. named entities and more specifically on the degree of change of the types of named entities and/or the degree of change of the actual names.

Regarding heuristics, we plan to make use of the information of the named entity instances contained in titles and paragraphs i.e. parts of texts to which they apply. We intend to examine the use of new features, such as different types of named entities that can further improve the performance of our algorithm. We also plan to exploit information resulting from the application of shallow parsing and/or co reference resolution in order to identify portions of text referring to the same topic as well as to exclude “false alarm” instances. Furthermore, we additionally consider examining alternative learning algorithms, besides conditionally random fields. Finally, due to the fact that, the measures of Precision, Recall and F-measure penalize equally every inaccurately estimated news item boundary whether it is near or far from a true segment boundary, we plan to evaluate our method using the Beeferman’s P_k metric (Beeferman, 1999) as well as the WindowDiff metric (Pevzer & Hearst, 2002). Those measures are successfully used for the evaluation of change of topic and text segmentation algorithms.

6. References

Arasu, A., Garcia-Molina, H. (2003). Extracting structured data from Web pages. In the *International Conference on Management of Data Proceedings, session: Data integration and sharing II*, pp: 337—348.

Beeferman, D., Berger, A. and Lafferty, J. (1999). Statistical models for text segmentation. *Machine Learning, 34*, pp.177--210.

Cai, D., Yu, S., Wen, J-R., Ma, W-Y. (2003). Extracting Content Structure for Web Page based on Visual Representation". In the *Fifth Asia Pacific Web Conference*.

Cai, D., Yu, S., Wen, J-R., Ma, W-Y. (2003). VIPS: a Vision-based Page Segmentation Algorithm. Microsoft Technical Report (MSR-TR-2003-79).

Chang, C-H., Kuo, S-C. (2004). OLERA: Semisupervised Web-Data Extraction with Visual Support. In *IEEE Intelligent Systems*, 19(6), pp. 56--64.

Chang, C-H., Hsu, C-N., Lui, S-C. (2003). Automatic information extraction from semi-structured Web pages by pattern discovery. *Web retrieval and mining*, 35(1), pp. 129 -- 147.

Crescenzi, V., Mecca, G., Merialdo, P. (2001). RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pp. 109 -- 118.

Crescenzi, V., Mecca, G., Merialdo, P. (2002). The RoadRunner Project: Towards Automatic Extraction of Web Data. In *Proceedings of the International*

Workshop on Adaptive Text Extraction and Mining in conjunction with the 17th International Joint Conference on Artificial Intelligence (IJCAI 2001) Symposium on Applied Computing, pp. 1108 -- 1112.

Feng, J., Haffner, P., Gilbert, M. (2005). A learning approach to discovering Web page semantic structures. In the *Proceedings of the Eighth International Conference on Document Analysis and Recognition*, pp. 1055--1059.

He, Z., Gao, Z., Xu, H., Qu, Y. (2005). DeSeA: A Page Segmentation based Algorithm for Information Extraction. In the *Proceedings of the First International Conference on Semantics, Knowledge and Grid,* pp. 14--14.

Hsu, C-N., Dung, M-T. (1998). Generating Finite-State Transducers for Semi-structured Data Extraction from the Web. In *Information Systems,* 23(9), pp. 521--538.

Kehagias, Ath., Fragkou P. and Petridis V. (2004). A Dynamic Programming Algorithm for Linear Text Segmentation. *Journal of Int. Information Systems*, 23, pp. 179--197.

Kushmerick, N. (2000). Wrapper Induction: Efficiency and Expressiveness. In *Artificial Intelligence*, nos. 1-2, pp. 15--68.

Li, S., Huang, S., Xue, G-R., Yu, Y. (2005). Block-based Language Modeling Approach towards Web Search. In *Proceedings of the seventh Asia-Pacific Web Conference*, pp. 170--182.

Muslea, I., Minton, S., Knoblock, C. (1999). A Hierarchical Approach to Wrapper Induction. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pp. 190--197.

Pevzner, L. and Hearst, M. (2002). A critique and improvement of an evaluation metric for text segmentation. Computational Linguistics, 28(1), pp.19--36.

Sha F., and Pereira F. (2003). Shallow Parsing with Conditional Random Fields. In *Proceedings of HLT-NAACL 2003*, Edmonton, Canada, 2003, pp. 213-220.

Song, R., Liu, H., Wen, J-R., Ma, W-Y. (2004). Learning important models for web page blocks based on layout and content analysis. In *ACM SIGKDD Explorations Newsletter,* 6(2), pp.14 --23.

Utiyama, M. and Isahara, H. (2001). A statistical model for domain - independent text segmentation. *In Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 491--498.

Wang, Y., Richard, R. (2007). Rule-based Automatic Criteria Detection for Assessing Quality of Online Health Information. In the *International Conference Addressing Information Technology and Communications in Health (ITCH).*

Wu, C., Zeng, G., Xu, G. (2006). A Web page Segmentation Algorithm for extracting product information. In the *IEEE International Conference on Publication*, pp. 1374--1379.

Yang, X., Xiang, P., Shui, Y. (2006). Semantic HTML Page Segmentation using Type Analysis. In the *1st International Symposium on Pervasive Computing and Applications*, pp. 669--674.

Zhai, Y., Liu, B. (2006). Structured Data Extraction from the Web Based on Partial Tree Alignment. In the *IEEE Transactions on Knowledge and Database Engineering*, 18(12), pp. 1614--1628.

Zhai, Y., Liu, B. (2005). Web data extraction based on partial tree alignment**.** In *Proceedings of the 14th international conference on World Wide Web*, pp: 76--85.

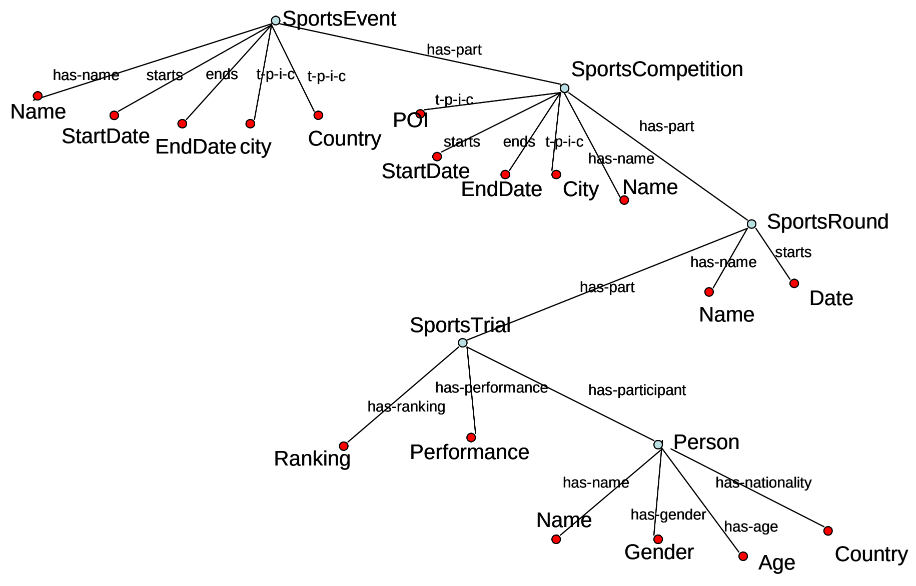
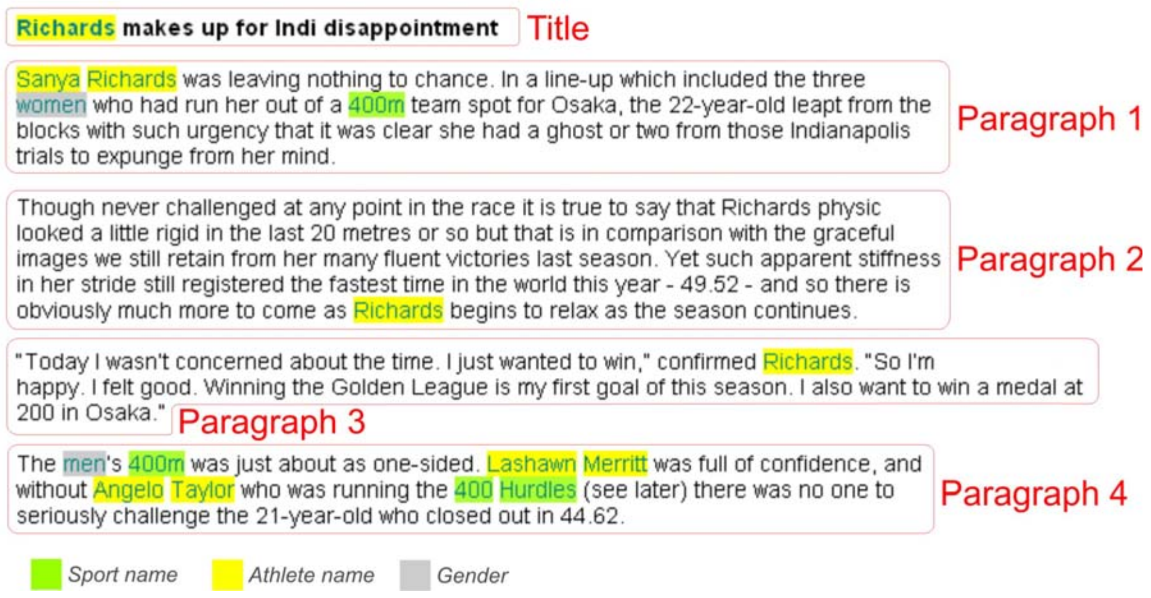


Figure 1: Part of the BOEMIE domain ontology



Richards makes up for Indi disappointment

Sanya Richards was leaving nothing to chance. In a line-up which included the three women who had run her out of a 400m team spot for Osaka, the 22-year-old leapt from the blocks with such urgency that it was clear she had a ghost or two from those Indianapolis trials to expunge from her mind.

Though never challenged at any point in the race it is true to say that Richards physic looked a little rigid in the last 20 metres or so but that is in comparison with the graceful images we still retain from her many fluent victories last season. Yet such apparent stiffness in her stride still registered the fastest time in the world this year - 49.52 - and so there is obviously much more to come as Richards begins to relax as the season continues.

"Today I wasn't concerned about the time. I just wanted to win," confirmed Richards. "So I'm happy. I felt good. Winning the Golden League is my first goal of this season. I also want to win a medal at 200 in Osaka."

The men's 400m was just about as one-sided. Lashawn Merritt was full of confidence, and without Angelo Taylor who was running the 400 Hurdles (see later) there was no one to seriously challenge the 21-year-old who closed out in 44.62.

Figure 2. Example of an annotated web page

Identification of Duplicate News Stories in Web Pages

John Gibson, Ben Wellner, Susan Lubar

The MITRE Corporation
202 Burlington Rd.
Bedford MA 01730
USA

Abstract

Identifying near duplicate documents is a challenge often faced in the field of information discovery. Unfortunately many algorithms that find near duplicate pairs of plain text documents perform poorly when used on web pages, where metadata and other extraneous information make that process much more difficult. If the content of the page (e.g., the body of a news article) can be extracted from the page, then the accuracy of the duplicate detection algorithms is greatly increased. Using machine learning techniques to identify the content portion of web pages, we achieve duplicate detection accuracy that is nearly identical to plain text and significantly better than simple heuristic approaches to content extraction. We performed these experiments on a small, but fully annotated corpus.

1. Introduction

News articles published on the Internet typically appear on many different websites in either identical or revised form. For users, identical and nearly identical duplicates are an annoyance. Duplicates slow down the process of finding new information on a topic, and potentially cause missed information if the user mistakenly identifies two documents as identical duplicates when in fact one contains new information. For automated processing such as named entity recognition and visualization, redundant data can cause incorrectly weighted results, markedly skewing search engine results and automated text processing applications.

While it is straightforward to find identical news stories in plain text documents, finding identical news stories embedded in web pages is considerably more complex. This is due to the large amount of “extraneous” information, such as navigation links, ads, Javascript, and other miscellaneous content contained in these pages. While the actual news story text on two separate web pages may be identical, the extraneous content on the pages will not be. Thus standard approaches for determining identical duplicates will fail.

In our system, named CEDAR which stands for Content Extraction and Duplicate Analysis and Recognition, we have taken a two-part approach to this problem. First, we have created a method for extracting news story text from web pages that is not website-specific (Gibson et al., 2007). We then use the extracted content to identify pairs of documents with the same news story content, ignoring any extraneous information on the pages. By calculating a resemblance score for pairs of documents based on a technique called shingling (Broder et al., 1997), we can identify both identical and near duplicate news stories.

2. Background

Duplicates are undesirable for many types of data. These include databases, mailing lists, file systems, email and image data. It is common practice to locate identical pieces of data using hashing strategies. Each piece of data is hashed using one of the standard algorithms, such as MD5. Any data represented by the same hash value is considered to be an identical duplicate.

Near duplicate documents are typically determined by computing a similarity score for each pair of documents in a collection. As presented by Chowdhury (Chowdhury et al., 2002) the two most common similarity measures are *resemblance* (Broder et al., 1997) and *cosine similarity* (Salton et al., 1975).

2.1. Cosine Similarity

To compute cosine similarity, documents are mapped into a vector space, typically based on term weights. The weight for each term is computed by the number of occurrences of the term in the document and an inverse measure of its frequency across a document collection. Document similarity is then measured by the cosine distance between the vectors.

2.2. Shingling

To compute the resemblance of two documents, each is broken into overlapping fragments called shingles. To do this, a shingle length, α, is specified. The first shingle is comprised of the first α words of the document. The second shingle consists of the second word in the document through the word located at $\alpha + 1$, and so on. Resemblance for the two documents is computed as the intersection size of the two documents’ shingle sets divided by the size of the union of these sets. Let A and B denote the two sets of shingles for two distinct documents, then their resemblance is defined as:

$$r(A, B) = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|}$$

One of the main drawbacks to shingling is the massive number of shingles generated, especially for large documents. Several strategies are used to reduce the number of shingles, while only slightly reducing the effectiveness of the algorithm. The first is sketching, which is the process of taking a uniform subset of the shingles, for example by discarding all shingles S for which $S \mod 25 \neq$

Approved for public release; Distribution Unlimited; Case #08-0238

0 (Broder et al., 1997). A further refinement is supershingling, which takes the shingles for each document that remain after sketching and shingling them again. These second order shingles are supershingles. If two documents have just a single supershingle in common then they can be considered near duplicates. A final performance enhancement comes from discarding very common shingles. Broder et al. (Broder et al., 1997) experiment with the AltaVista dataset; they discarded all shingles that appeared in 1000 or more documents (Broder et al., 1997). This greatly reduces the size of the shingle sets which leads to large space and time savings.

2.3. Locality Sensitive Hashing

As an alternative to computing the resemblance or cosine distance, there exist hash functions that yield similar values for similar documents. These functions produce collisions with a probability equal to the resemblance or cosine distance. Or as a formal definition from Charikar (Charikar, 2002):

A locality sensitive hashing scheme is a distribution on a family F of hash functions operating on a collection of objects, such that for two objects x, y,

$$Pr_{h\in F}[h(x) = h(y)] = sim(x, y)$$

The main performance advantage of this approach is that the resulting hash values require much less storage space than the shingles of a document or the full term vectors of the cosine distance approach.

3. Related Work

3.1. Duplicate Detection

Recently Henzinger combined shingling and locality sensitive hashing (LSH) to achieve good results on a very large dataset (1.6 billion distinct web pages) (Henzinger, 2006). Henzinger's experiment was particularly interesting because a subset of the data was evaluated manually which allowed for a more rigorous assessment of the accuracy of the technique. The combined algorithm first used shingling and then applied an LSH that estimates the cosine distance. The shingling portion used a variant of Fetterly, et al.'s algorithm (Fetterly et al., 2003) (which is a variant of Broder, et al.'s algorithm). Specifically it used a minvalue sketching technique followed by supershingling. Pages were considered near duplicates if they had at least 2 matching supershingles. Next, the LSH was applied to all of the documents identified as near duplicates by the first pass. Then the bit sequence for each document was divided into pieces and any pair of documents that had a single piece in common were compared more thoroughly. If at least 355 of the 384 bits of the hash matched then the pair was kept as a near duplicate. The combined algorithm yielded substantial gains in precision with only a moderate impact upon recall.

A major difference between Henzinger's work and our own is that it has a different definition of near duplicates. Henzinger's near duplicates are documents that differ not in content, but in boilerplate or other miscellaneous page structure (session ids, username, hit count, etc.). We would consider those to be identical duplicates. This distinction is important because the aim of Henzinger's experiment was to eliminate redundant documents, while ours is to identify all of the related versions of each document. Henzinger's results are difficult to compare to ours for this reason; additionally the experiments in that work operate over a much larger set of web documents than we examine here, are focused more on scalability and examines all types of documents randomly taken from the Web rather than just news stories as in our work. That being said, Henzinger reports pair-wise precision scores of 79%. The level of recall is unknown because only a subset of their corpus was annotated. However, the combined algorithm did have a pair-wise recall score of 79% on the results of the shingling algorithm alone.

3.2. Content Extraction

Content extraction tools such as Columbia University's Crunch aim to reduce the size of web pages by removing what they deem as noise or clutter from the pages (Gupta et al., 2005). Additionally, a tool by the Document Analysis and Recognition Team (DART) at BCL Computers Inc. further reduces text by providing a summary of what remains (Rahman et al., 2001). These tools are motivated by a variety of goals including paring down pages for the visually impaired (Gupta et al., 2005), producing lighter weight content for small screen devices such as PDAs (Gupta et al., 2005; Rahman et al., 2001) and reducing page complexity for subsequent processing as in MetaNews (Kang and Choi, 2003) and in the CLEANEVAL challenge task as part of the Web as a Corpus Workshop (WAC 2007, 2007).

These Content Extraction tools are related to our work in that they are focused on determining which parts of web pages are relevant to their goal. However, they address the broader problem of operating on any type of web page, while we are focused solely on pages containing news articles.

3.3. News Story Content Extraction

MetaNews (Kang and Choi, 2003) and the Columbia Newsblaster project (Evans et al., 2004) both concentrate on gathering news articles on the web. MetaNews uses a two-phased approach. First, it carries out noise removal by throwing out HTML tags that it believes will not contain content. Next it uses pattern matching on the reduced page to extract news articles. Patterns for MetaNews are manually defined for each news site, and no automatic learning is involved. Thus although pattern writing is simpler than for traditional wrapper approaches, this tool is still likely to fail if a page format changes, and adding new sites requires some manual labor.

The Columbia Newsblaster team originally used individual site wrappers to identify news articles. They determined that this approach was difficult for handling new sites. As a result, they implemented a machine learning based approach which is similar to ours. The module relies on "simple surface characteristics of the text" to classify blocks of text as part of an article, or into various other categories such as title, caption, or other.

4. Data

4.1. Harvesting

To create a data set containing duplicate news stories, we started by obtaining article titles from the Reuters and Associated Press (AP) RSS feeds. [1] We then sent each title as a query to Google News and downloaded the top ten results for each query. Because of the large variance in the pagination methods of each site we limited downloads to the first page of each article. In total we harvested 2577 documents from 49 separate websites. We only annotated a subset of these; the resulting data set included 1621 documents from 27 different websites.

After harvesting, we took two steps to turn the data into a reference standard that could be used for training and scoring. First, we annotated the documents to indicate which portions of text were news story content. Second, we identified and recorded which document pairs contained identical or near duplicate news stories.

4.2. News Article Annotation

Manually annotating 2577 documents was a daunting task, so we instead automated the process by writing site-specific taggers. In the end, we reduced the size of our dataset for a variety of reasons. First, we concluded that it was not worth the effort to write taggers for sites that contributed only a few documents to the dataset. Secondly, there were two sites whose page format would have been difficult to annotate automatically. Finally, we decided to exclude over 500 articles that came directly from various Reuters sites. We felt that it would not be as interesting to use these articles as a basis for content extraction because it is possible to obtain a plain text feed directly from Reuters. As a result of these exclusions, our final dataset was comprised of 1621 documents from 27 different websites.

For each of these documents, we first corrected poorly formatted HTML using Beautiful Soup[2] and Tagsoup.[3] We then inserted tags around the body of the news article. Where possible, we also tagged the article's author, date, location, source, and title.

4.3. Duplicate Identification

The second step in creating a reference standard was to identify and record duplicate pairs in the document collection. First we located identical pairs by normalizing the article text and comparing all of the documents directly. This analysis found that there were 564 distinct articles.

Marking near duplicate pairs was carried out as a manual process because it requires a user's judgment to determine whether two articles are near duplicates. We implemented a basic GUI to accelerate the process. It displays two articles side by side, and highlights the differences between the articles' content. A picture of the viewer appears in Figure 1. Because there are over 1 million possible pairings of 1621 documents, a manual comparison of all possible pairs would have been extremely time consuming. Therefore, we used a few techniques to speed up the process. First we display only one document for each unique article as identified by the first pass, bringing the total number of documents for comparison down to 564. When a document pair was marked as near duplicates we automatically added near duplicate pairings for all of the exact duplicates of the two marked documents. Because of the method we used for harvesting documents, we were confident that the bulk of the duplicates would also share the same title. Thus we began by reviewing document pairs that had similar titles. Once we had implemented our shingling algorithm we used it to find the few remaining documents that were near duplicates but had different titles.

We had six annotators review the data and record near duplicate pairs. Because determining whether a fairly different, but related pair is somewhat subjective, we held group discussions to determine the status of questionable pairs. Also, after the first pass of all of the documents was completed, we assigned a different annotator from the original group to carry out a second pass and verify the pairs that were recorded. In the end the reference standard contained 3591 identical duplicate pairs and 1231 near duplicate pairs.

5. Content Identification

Once we had created a reference standard we used the data to develop a machine learning-based system for identifying the content of the news articles.

5.1. Division into Blocks

The first step in our approach was to divide the web page into smaller pieces for our algorithms to identify as CONTENT or NOTCONTENT.

We sanitized the raw HTML by transforming it into XHTML using Tagsoup and Beautiful Soup. In the rare case that a document that could not be transformed into XHTML, we discarded it. Otherwise we tokenized the document. We excluded all words inside style and script tags from tokenization. It is safe to assume that those tags will never contain any content because they will not be rendered by a browser.

Next we partitioned the sequences of tokens into blocks. Intuitively, we define a block as a sequence of text that when rendered in a browser does not cause a line break. More formally we defined blocks as sequences of text that are bounded by any tag except the following: `<a>`, `<ins>`, `<del>`, `<span>`, `<bdo>`, `<em>`, `<strong>`, `<dfn>`, `<code>`, `<samp>`, `<kbd>`, `<var>`, `<cite>`, `<abbr>`, `<acronym>`, `<q>`, `<sub>`, `<sup>`, `<tt>`, `<i>`, `<b>`, `<big>`, `<small>`, `<u>`, `<s>`, `<strike>`, `<basefont>`, and `<font>`.

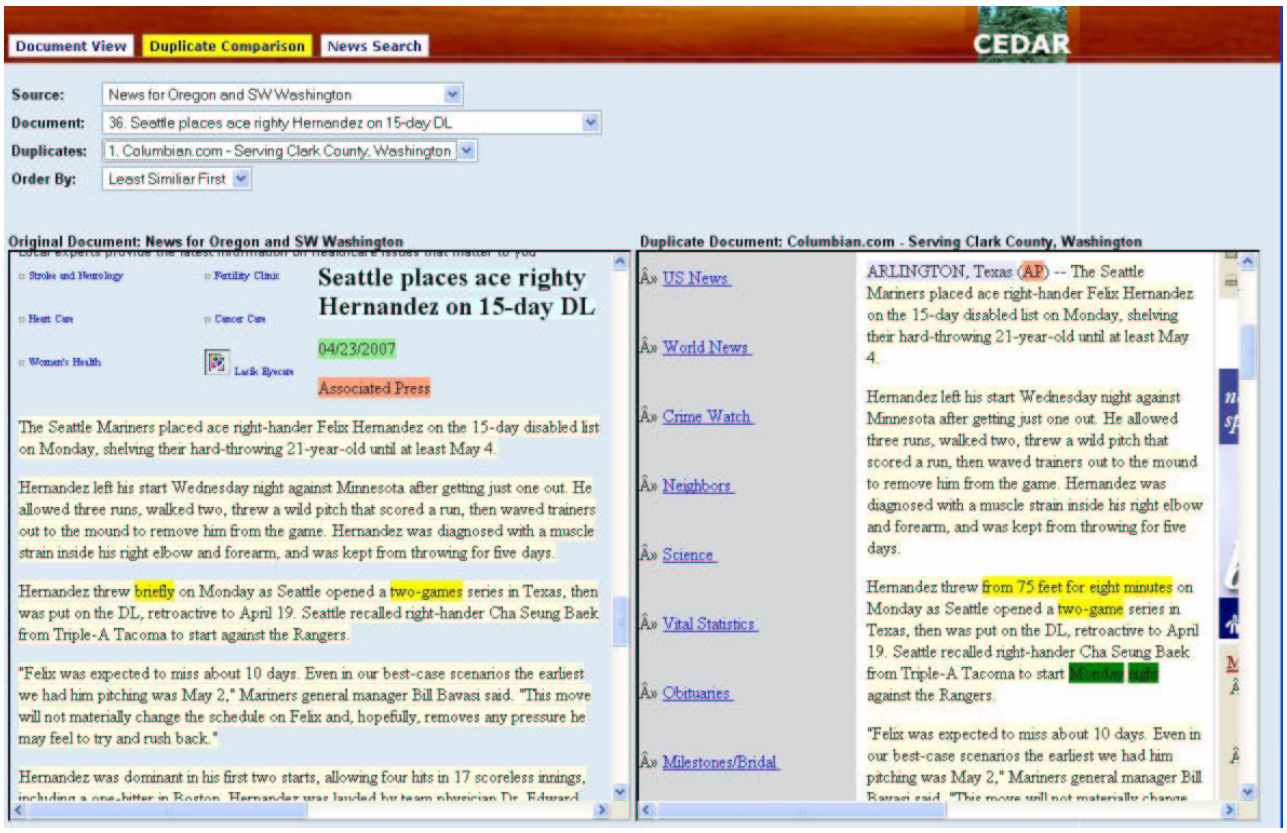
5.2. Feature Generation and Classification

Once the individual blocks were identified we generated a variety of feature types from each block that we subsequently used to train our algorithms. These feature types included a simple bag of words with frequency, a count of the tokens in a block, the percentage of tokens in a block that were contained within an anchor tag (`<a>`), tags in and

[1]We used different feeds from http://www.reuters.com/tools/rss and http://hosted.ap.org/dynamic/fronts/RSS?SITE=AP.

[2]Beautiful Soup was written by Leonard Richardson, et al. and can be found at http://www.crummy.com/software/BeautifulSoup/

[3]TagSoup was written by John Cowan and can be found at http://ccil.org/ cowan/XML/tagsoup/



Source: News for Oregon and SW Washington
Document: 36. Seattle places ace righty Hernandez on 15-day DL
Duplicates: 1. Columbian.com - Serving Clark County, Washington
Order By: Least Similiar First

Original Document: News for Oregon and SW Washington

Local experts provide the latest information on healthcare issues that matter to you

Seattle places ace righty Hernandez on 15-day DL

04/23/2007

Associated Press

The Seattle Mariners placed ace right-hander Felix Hernandez on the 15-day disabled list on Monday, shelving their hard-throwing 21-year-old until at least May 4.

Hernandez left his start Wednesday night against Minnesota after getting just one out. He allowed three runs, walked two, threw a wild pitch that scored a run, then waved trainers out to the mound to remove him from the game. Hernandez was diagnosed with a muscle strain inside his right elbow and forearm, and was kept from throwing for five days.

Hernandez threw briefly on Monday as Seattle opened a two-games series in Texas, then was put on the DL, retroactive to April 19. Seattle recalled right-hander Cha Seung Baek from Triple-A Tacoma to start against the Rangers.

"Felix was expected to miss about 10 days. Even in our best-case scenarios the earliest we had him pitching was May 2," Mariners general manager Bill Bavasi said. "This move will not materially change the schedule on Felix and, hopefully, removes any pressure he may feel to try and rush back."

Hernandez was dominant in his first two starts, allowing four hits in 17 scoreless innings, including a one-hitter in Boston. Hernandez was lauded by team physician Dr. Edward

Duplicate Document: Columbian.com - Serving Clark County, Washington

ARLINGTON, Texas (AP) -- The Seattle Mariners placed ace right-hander Felix Hernandez on the 15-day disabled list on Monday, shelving their hard-throwing 21-year-old until at least May 4.

Hernandez left his start Wednesday night against Minnesota after getting just one out. He allowed three runs, walked two, threw a wild pitch that scored a run, then waved trainers out to the mound to remove him from the game. Hernandez was diagnosed with a muscle strain inside his right elbow and forearm, and was kept from throwing for five days.

Hernandez threw from 75 feet for eight minutes on Monday as Seattle opened a two-game series in Texas, then was put on the DL, retroactive to April 19. Seattle recalled right-hander Cha Seung Baek from Triple-A Tacoma to start Monday night against the Rangers.

"Felix was expected to miss about 10 days. Even in our best-case scenarios the earliest we had him pitching was May 2," Mariners general manager Bill Bavasi said. "This move will not materially change

Figure 1: Example duplicates

around the block, inverse stop wording, named entities, and a few other feature types. A complete list with details can be found in our previous paper (Gibson et al., 2007).

We experimented with a variety of machine learning approaches that made use of the above features to identify content containing blocks. A key insight is to not label each block *independently*, but rather to label them using sequential classification methods that take into account the whole sequence of decisions over all the blocks in a single article. Conditional Random Fields (CRFs), a flexible statistical model able to capture sequential dependencies while also allowing rich, arbitrary features proved to be the most successful approach (Gibson et al., 2007).

We also tried using a simpler maximum entropy model with only word features to judge the effect of the quality of the content extraction step upon duplicate detection.

5.3. Content Extraction Methodology

In order to determine the effectiveness of content extraction for duplicate identification, we needed to run our content extraction system over our entire duplicate data set. However, the content extraction system relies on the same articles as a source of training data. Further, many of the articles in the collection are duplicates, and many are from the same web-site. Ideally, in the most conservative and fair setting, the content extractor would not be trained on any of the same articles (including duplicates and near duplicates) or any of the same sources as to which it is applied.

To achieve this on our collection we used a four-fold cross validation approach. Each of the four folds contained 6 or 7 sources, with each source only belonging to exactly one fold. We assigned sources to folds in round-robin fashion by selecting a source at random weighted by the number articles belonging to that source. This kept each fold at roughly the same size in terms of the number of articles.

Three folds constituted the training data while a fourth served as test data. This would allow each article to be processed with a content extractor that was trained on articles from web-sites not appearing in the test data. However, due to many duplicates in our corpus, the training data will likely contain many of the same articles as in the test data, though from different sources. To compensate for this problem, if any duplicate documents spanned the training and test data for a particular fold, we removed those articles from the training set. An illustration of this process can be seen in Figure 2. This process was repeated four times, with each fold taking its turn as test data.

6. Experiments and Results

Our experiments in this section aim to demonstrate 1) the effect that various system parameters and options have on overall accuracy and 2) how well the system, tuned on development data, performs on held-out evaluation data. We believe that the latter is an indicator of how well the system will perform on new web pages from news sources.

6.1. Evaluation Methods

Before describing our experiments in detail, we highlight two methods for evaluating duplicate detection accuracy.

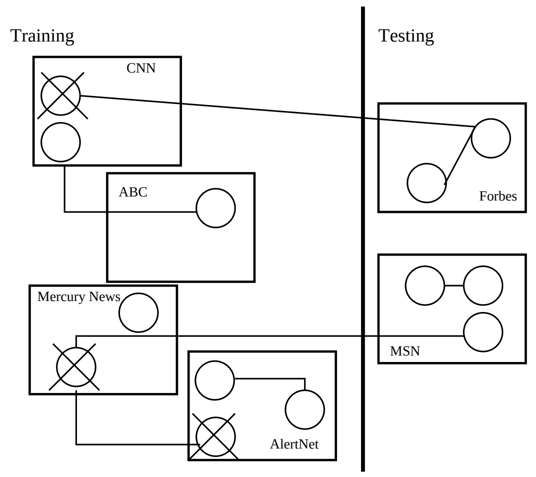


Figure 2: Data partitioning for content extraction model. Boxes indicate sources, circles are documents. Circles connected by lines are duplicates. Crossed out circles were dropped from the training set.

6.1.1. Pair-wise Evaluation

The most natural way to evaluate a duplicate detection system is with a pair-wise evaluation metric. Given a set of documents, the reference standard provides information as to whether each pair of documents is a duplicate pair. A system for duplicate detection is responsible for assigning a similarity score, s, to each pair of documents. Given a threshold, T, each pair with a score, $s \geq T$ is hypothesized to be a duplicate pair. For a particular threshold, T, we can evaluate the precision and recall of the duplicate system where precision is computed as:

$$\frac{\text{\# correct duplicate pairs}}{\text{\# hypothesized duplicate pairs}}$$

and recall computed as:

$$\frac{\text{\# correct duplicate pairs}}{\text{\# true (reference standard) duplicate pairs}}$$

For different applications, different threshold values may be appropriate depending on whether precision or recall is preferred. A way to compare the results of different systems across all threshold values is to look at Receiver Operator Characteristic (ROC) curves, which plots the true positive rate against the false positive rate, essentially capturing the precision rate at all possible recall levels. The area under this curve provides a single number useful for comparing two systems that assign scores to positive and negative instances. The area under the curve, in our context, can be interpreted as the probability that the system/model will provide a higher resemblance score to an arbitrary document pair p_1 than a pair p_2 when p_1 is, in fact, a duplicate pair and p_2 is not. We use the area under the ROC, AU-ROC curve for some of the results that follow. Note that, in particular, the graphs below plot how the AU-ROC changes as a parameter is adjusted - the graphs are not ROC curves themselves.

Note that for our full data set of 1621 web page documents, there are 1,313,010 unordered document pairs of which just 4815 are duplicate pairs.

6.1.2. Cluster-Based Evaluation

While pair-wise evaluation is intuitive and appropriate in many cases, it has the disadvantage of skewing the results when the pair-wise relation is an equivalence relation[4]. This is because mistakes made with documents belonging to larger equivalence classes will be penalized more than documents within small equivalence classes. This can provide for a rather unintuitive evaluation metric. The cluster-wise evaluation metric considers the degree to which the resulting equivalence classes match each other rather than considering all pair-wise relations.

Equivalence class-based evaluation metrics have been used for evaluation within document and cross document co-reference in natural language processing (Vilain et al., 1995; Amit and Baldwin, 1998). For our formal evaluation of the system, we use the B^3 scoring metric (Amit and Baldwin, 1998) and its implementation within the LingPipe suite of NLP tools[5]. Very briefly, the B^3 metric takes as input the reference standard clusters and the clusters derived from the system output (obtained in our case by asserting that all pairs with a resemblance above a certain threshold are duplicates and then taking the transitive closure). For a *document*, d, precision, P_d, and recall, R_d, are computed as follows:

$$P_d = \frac{\text{\# of correct documents in the output cluster containing } d}{\text{\# of documents in the output cluster containing } d}$$

and

$$R_d = \frac{\text{\# of correct documents in the output cluster containing } d}{\text{\# of documents in the truth cluster containing } d}$$

The final precision and recall scores for the output clusters are the average precision and recall scores across all the documents.

6.2. System Configuration Analysis

In this section we consider a set of experiments aimed at identifying the effect of different system parameters on duplication detection accuracy.

The first set of experiments looks at the effect of varying the system parameters:

Block size filter This option removes blocks (described above) if their word/token count is below a certain threshold. The threshold varies from 1 to 20.

Shingle frequency filter This option removes the N% most frequent shingles in the corpus, values range from 0.0% to 10.0%.

Shingle size This parameter controls the size of the shingles in words. It is an integer varying from 1 to 10.

[4]While we have not restricted our similarity relation to be transitive, the manually annotated duplicate pairs reveal a similarity relation that is transitive in nearly all cases. This may be a reflection on our particular data set, however.

[5]LingPipe is available at http://alias-i.com/lingpipe/

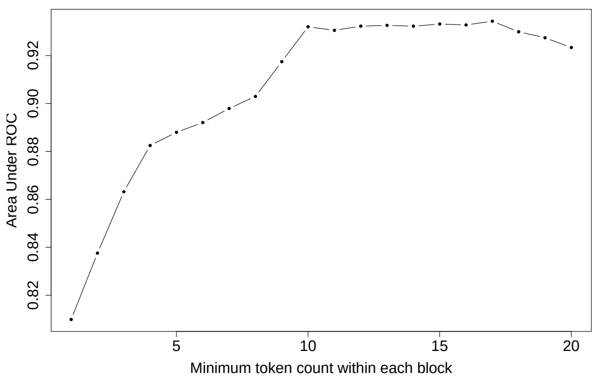


Figure 3: Duplicate detection accuracy with different minimum word counts per block with word shingles of size 6.

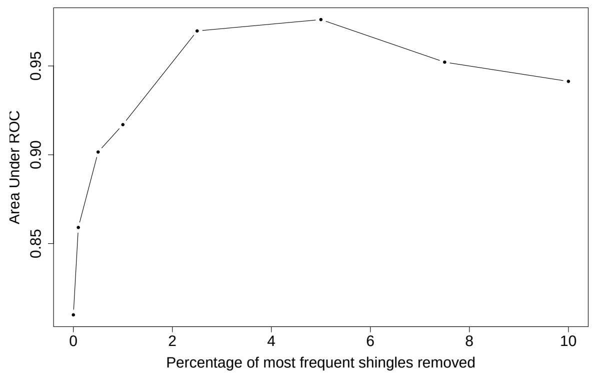


Figure 4: Duplicate detection accuracy with different shingle cutoff percentages using word shingles of size 6.

Figure 3 shows how duplication detection accuracy, in terms of AU-ROC (on the Y-axis), varies as more and more blocks are filtered from the documents based on the minimum token count (shown on the X-axis). As the minimum count is increased, accuracy improves until a minimum count of 10 with increasing minimum counts not resulting in noticeable improvement and eventually degradation in accuracy after reaching a minimum of 17 or more. The shingle size is fixed at 6 while the shingle frequency filter is inactive.

Figure 4 contains a graph illustrating the effect of removing varying percentages of the most frequent shingles. Accuracy improves even more substantially as compared with the block filter here. The explanation for this is that many of the shingles in the non-content portions of the document appear over and over again as part of the web-page boilerplate, in contrast to shingles found in the content portion of the article. Again, shingle size was fixed at 6 and no block-level filtering was performed.

A final graph is shown in Figure 5 demonstrating the effect different shingle sizes have on overall accuracy. The best duplicate identification accuracy is found with shingles of size 2. Surprisingly, single word shingling performs very well, outside the standard deviation of accuracy with shingles of size 4.

In addition to the above contributions, we also examined the effect of content extraction on accuracy. The hypothesis here is that extracting the article content by removing extraneous parts of the document, will improve duplicate detection accuracy. We look at four different content extraction (CE) approaches here:

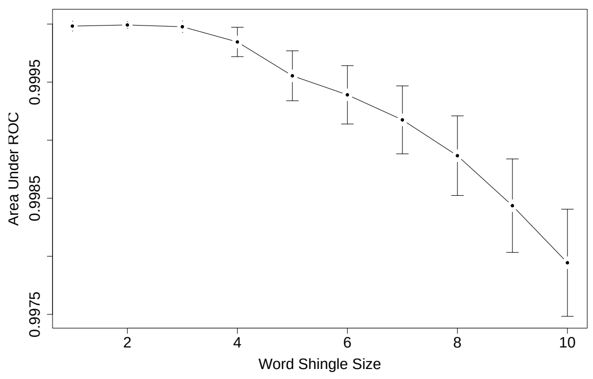


Figure 5: Duplicate detection accuracy with different word shingle sizes. The standard deviation bars, computed using the approach in (Hanley and McNeil, 1982), demonstrate significantly lower accuracy with larger shingles.

Shingle/Block Filter	AU-ROC
Heuristic	0.91076 ± 0.00283
CE-SIMPLE	0.9999922 ± 0.0000284
CE-BEST	0.9999933 ± 0.0000272
Ref. Standard	0.9999935 ± 0.0000259

Table 1: Content Extraction-based results using word shingles of length 2.

Heuristic Removal of document regions between `<script>` and `<style>` tags.

CE-SIMPLE A simple maximum entropy classifier to identify blocks containing article content. The feature set used for the classifier consisted of only the words present in the block. The document-level accuracy of this CE system is 58% – meaning that it is able to perfectly identify the content portion of a web-page 58% of the time. The block-level precision is 96.0 with recall at 98.1.

CE-BEST The automatic best content extraction system based on CRFs using a richer feature set that considered the presence of particular HTML tags, the presence of named entities (e.g. people, organizations, etc.) and other features. Document-level accuracy for this CE system is 80% with block-level precision and recall at 97.9 and 99.5, respectively.

Ref. Standard Only the portions of the documents considered content by site-specific taggers and reviewed by human annotators were used to compute resemblance.

The results in Table 1 illustrate the effects of these different methods on duplicate detection.

A final analysis shown in Table 2 looks at combining the block filter and the shingle frequency filter with and without using the heuristic-based content extraction. The results here indicate that using the heuristic-based CE *together* with filtering out the top 5% most frequent shingles and removing blocks with fewer than 12 tokens (bottom

Shingle/Block Filter	AU-ROC
Entire Web Document	
None	0.94275 ± 0.00233
MinBlock = 5	0.95883 ± 0.00200
Shingle Freq. = 7.5%	0.99751 ± 0.00005
Heuristic Content Extraction	
Heuristic Only	0.91076 ± 0.00283
Shingle Freq.= 7.5%	0.99500 ± 0.000718
Shingle Freq.= 5.0% & MinBlock = 12	0.99994 ± 0.000079

Table 2: Combinations of block filtering, shingle filtering and heuristic CE using shingles of length 2.

row) achieves very high results in terms of AU-ROC. These results appear to be competitive with duplication detection approaches that use a statistically trained CE system (in Table 1). However, as we demonstrate below, statistically-driven content extraction does significantly improve duplication detection accuracy over the best system that doesn't use statistical CE.

6.3. Formal Evaluation

In this section we describe the results of a somewhat more formal evaluation with a fixed development and test split of the data. While the above analysis based on AU-ROC provides insight into the contributions of different aspects of the system on overall accuracy, in a realistic setting one must pick a fixed threshold.

We split the data into two roughly equal-sized sets of documents such that no equivalence classes of duplicate documents (according to the reference standard) overlapped both sets. The development portion of the data was used to tune the threshold value and to optimize various parameters such as the minimum word count block filter and the shingle frequency cutoff. Table 3 shows the official results in terms of the cluster-based B^3 metric as well as the more standard pair-wise metric.

We also provide scores for using a normalized string comparison on the reference standard's content. This approach finds all of the identical duplicates and none of the near duplicates.

The most obvious result of these experiments is that effective content extraction provides a significantly higher level of accuracy than more basic techniques. By enabling the duplicate detection algorithm to focus on the *article content*, filtering out extraneous web page material, accuracy is improved considerably. The overall results here are very promising indicating that a very high percentage of duplicate document clusters can be identified perfectly.

Another interesting result here is that even a very simple machine learning content extraction approach provides for duplicate detection accuracy that is nearly identical to using the reference standard extracted content. Implementing this approach requires little effort provided a set of training documents with the content portions annotated (see (Gibson et al., 2007)).

7. Conclusions

Our approach to detecting identical and near duplicate news articles embedded in web pages is a two step process. Our system, CEDAR, first extracts the text of the news articles from the pages, and then computes resemblance scores for the articles using a shingling approach. We carried out a number of experiments which show that basing the duplicate detection purely on the extracted content results in more accurate results than computing resemblance across the text of the original documents.

Additionally, we have implemented a flexible, non-brittle approach for identifying news article text in web pages. We created a model for our CRF classifier based on the structure of web news pages across twenty-seven different news websites. The classifier can now be used to find news article text embedded in pages from previously unseen web sites, and does not break when the formatting on a web site changes.

Though this content extraction technique does not always provide perfect results, it is accurate enough to allow our duplicate detection system to outperform itself when using more naive approaches to identifying article text. Moreover, this content extraction module can be used to improve results for many different tasks involving news articles on the web. Some examples are named entity extraction, visualization, search indexing and display on a small screen such as a PDA or cell phone.

8. Future Work

Our future work is focused on making CEDAR deployable. For the content extraction this means optimizing the preprocessing and feature generation as well as moving to a faster implementation language (currently, we are using Python). In situations where the only purpose of extracting content is to improve duplicate detection accuracy, we can use the simpler MaxEnt system which requires many fewer features and is faster in general than the full CRF system.

Because of the relatively small size of the dataset, our duplicate analysis implementation was performed in memory. To handle larger datasets we will experiment with sketching and supershingling, as well as redesigning the code to work on smaller chunks at a time. We can also switch to an entirely different algorithm such as cosine distance or fingerprinting. Finally, if we were only concerned with improving the precision of a system, then we could use our system on the clusters generated by other systems. As long as the clusters themselves are not prohibitively large, our current performance limitations would not be an issue and we could eliminate a large number of false positives.

Another direction for our work is to experiment with processing foreign language text, particularly for languages written in other character sets. It is our expectation that translating the documents to English before shingling will generate poor results. However, by applying word segmentation techniques we should be able to achieve reasonable accuracy using word based shingling. Or, by normalizing the text to remove all segmentation whatsoever, shingling based on characters rather than words should achieve very good results. However, this approach will create an enormous number of shingles and therefore performance can

System	Threshold	Cluster-based			Pair-wise		
		Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.
Heur. and Shingle Freq. = 5%	0.35	0.929	0.854	0.89	0.977	0.782	0.869
Heur. and MinBlock = 8	0.4	0.919	0.860	0.889	0.919	0.860	0.889
Heur., Shingle Freq. = 5% & MinBlock = 9	0.35	0.981	0.862	0.917	0.981	0.862	0.917
CE-BEST	0.4	0.992	0.979	0.985	0.992	0.979	0.985
CE-SIMPLE	0.4	0.992	0.977	0.985	0.992	0.977	0.985
Ref.	0.4	0.992	0.977	0.985	0.992	0.977	0.985
Ref. String Comparison	N/A	1.0	0.802	0.89	1.0	0.745	0.854

Table 3: Cluster-based and pair-wise results on the evaluation data with threshold and system parameters tuned on the development data.

become unacceptably slow. Our plan is to experiment with character based shingling combined with sketching to increase speed while maintaining reasonable accuracy.

Acknowledgements

This work was supported by the Department of the Army's Communications-Electronics Lifecycle Management Command (C-E LCMC) and performed under MITRE Mission Oriented Investigation and Experimentation (MOIE) project M130 of contract W15P7T-07-C-F600, sponsored by the Pacific Regional Service Center. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the sponsor.

9. References

Amit, B. and B. Baldwin, 1998. Algorithms for scoring coreference chains. In *Proceedings of the Seventh Message Understanding Conference (MUC7)*.

Broder, Andrei Z., Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig, 1997. Syntactic clustering of the web. In *Selected papers from the sixth international conference on World Wide Web*. Essex, UK: Elsevier Science Publishers Ltd.

Charikar, Moses S., 2002. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*. New York, NY, USA: ACM.

Chowdhury, Abdur, Ophir Frieder, David Grossman, and Mary Catherine McCabe, 2002. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191.

Evans, D. K., J. L. Klavans, and K. McKeown, 2004. Columbia Newsblaster: Multilingual news summarization on the web. In *Proceedings of Human Language Technology Conference/ North American Chapter of the Association for Computational Linguistics*.

Fetterly, Dennis, Mark Manasse, and Marc Najork, 2003. On the evolution of clusters of near-duplicate web pages. In *LA-WEB '03: Proceedings of the First Conference on Latin American Web Congress*. Washington, DC, USA: IEEE Computer Society.

Gibson, John, Ben Wellner, and Susan Lubar, 2007. Adaptive web-page content identification. In *WIDM '07: Proceedings of the 9th annual ACM international workshop on Web information and data management*. New York, NY, USA: ACM.

Gupta, S., D. Daiser, P. Grimm, M. Chiang, and J. Starren, 2005. Automating content extraction of html documents. *World Wide Web - Internet and Information Systems*, 8(2):179–224.

Hanley, JA and BJ McNeil, 1982. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143:29–36.

Henzinger, Monika, 2006. Finding near-duplicate web pages: a large-scale evaluation of algorithms. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM.

Kang, D. and J. Choi, 2003. Metanews: An information agent for gathering news articles on the web. In *International Symposium on Methodologies for Intelligent Systems*.

Rahman, A. F. R., H. Alam, and R. Hartono, 2001. Understanding the flow of content in summarizing html documents. In *International Workshop on Document Layout Interpretation and its Applications (DLIA)*.

Salton, G., A. Wong, and C. S. Yang, 1975. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620.

Vilain, Marc, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman, 1995. A model-theoretic coreference scoring scheme. In *MUC6 '95: Proceedings of the 6th conference on Message understanding*. Morristown, NJ, USA: Association for Computational Linguistics.

WAC 2007, Web as a Corpus, 2007. WAC2007. In *Web as a Corpus*. UCLouvain, Louvain-la-Neuve, Belgium.

GlossaNet 2: a linguistic search engine for RSS-based corpora

Cédrick Fairon, Kévin Macé, Hubert Naets

Centre de Traitement Automatique du Langage — Cental
Université Catholique de Louvain
Louvain-la-Neuve, Belgique
{cedrick.fairon,kevin.mace,hubert.naets}@uclouvain.be

Abstract

This paper presents GlossaNet 2, a free online concordance service that enables users to search into dynamic Web corpora. Two steps are involved in using GlossaNet. At first, users define a corpus by selecting RSS feeds in a preselected pool of sources (they can also add their own RSS feeds). These sources will be visited on a regular basis by a crawler in order to generate a dynamic corpus. Secondly, the user can register one or more search queries on his / her dynamic corpus. Search queries will be re-applied on the corpus every time it is updated and new concordances will be recorded for the user (results can be emailed, published for the user in a privative RSS feed, or they can be viewed online). This service integrates two preexisting software: Corporator (Fairon, 2006), a program that creates corpora by downloading and filtering RSS feeds and Unitex (Paumier, 2003), an open source corpus processor that relies on linguistic resources. After a short introduction, we will briefly present the concept of "RSS corpora" and the assets of this approach to corpus development. We will then give an overview of the GlossaNet architecture and present various cases of use.

1. Introduction

Over the last 10 years, growing needs in the fields of Corpus Linguistics and NLP have led to an increasing demand for text corpora. In this context, the development of the Internet was seen as a real opportunity to help meeting the demand (Kilgarriff and Grefenstette, 2003; Hundt et al., 2007). The Web is indeed immense, very diverse and easily accessible. . . but at the same time it is mixed up, incoherent and most of the time unforeseeable. Although it is technically easy to get access to online documents, it is still a challenge to extract clean data from these documents to compose a corpus. The Cleaneval competition[1] is a good indicator of the state of the art in this domain.

Corpus linguists and NLP specialists have been using the Web as corpus in two directions that complement each other. A first approach considers the Web itself as a very large corpus. Systems that implement this approach usually offer an interface for querying and seeking concordances within the Web. Basically, they add a layer to traditional search engines like Google or Microsoft Live Search and offer various display options that are useful for linguistic work (concordances, extraction of collocates, stop lists, etc.). Among other systems: WebCorp[2] (Renouf, 2003), WebCorpus[3] (Fletcher, 2007), Corpeus[4] (Leturia et al., 2007).

A second approach consists in using the Web as a source for extracting texts that will be collected, filtered and recorded in a standalone corpus. Systems of this second category rely on the use of crawlers and filtering techniques. The Wacky[5] project is a typical example of this option (Baroni and Bernardini, 2006) but other examples are numerous (Duclaye et al., 2003; Fletcher, 2007).

Similarly to the first category, the GlossaNet system we are about to present is an online tool tailored for use by linguists, but the way it sees corpora is closer to the second category. Indeed, it offers tools for defining "dynamic corpora" that will be downloaded and refreshed on a regular basis by the system.

The original GlossaNet service[6] has been in use since 1998 (Fairon, 1999) and was limited to press corpora. At the time, newspapers'websites were seen as a good source for generating "dynamic corpora" because they are updated on a daily basis[7] (new texts come continuously day after day). GlossaNet users can create an account, select a series of newspapers (which define a virtual corpus) and register search queries. The system integrates the programs and linguistic resources of Unitex[8], an open source corpus processor (Paumier, 2003). Unitex is used for applying several analysis tasks on the corpus: tokenization, sentence segmentation, dictionary lookup. Once the corpus is processed, it becomes possible to search for linguistic patterns. To make it short, one can describe GlossaNet as a linguistic search engine of limited scope. Until recently, this scope was very limited as it covered only a hundred newspapers websites. GlossaNet 2 goes beyond this limitation as it can download corpora from any RSS / Atom feed (see section 2.). GlossaNet integrates two preexisting pieces of software: Corporator (Fairon, 2006), a program that creates corpora by downloading and filtering RSS feeds and Unitex. Both software are available as standalone applications. Two steps are involved in using GlossaNet. At first, users define a corpus by selecting RSS feeds in a preselected pool of sources (they can also add their own RSS feeds). These sources will be visited on a regular basis by a crawler in

[1] `http://cleaneval.sigwac.org.uk/`
[2] `http://www.webcorp.org.uk/`
[3] `http://webascorpus.org/searchwac.html`
[4] `http://www.corpeus.org/`
[5] `http://wacky.sslmit.unibo.it/`

[6] `http://glossa.fltr.ucl.ac.be/`
[7] In addition, newspapers are available in many different languages,they cover many different themes, they represent various styles, various types of text (argumentative, informative, technical, pedagogical, etc.) and they provide texts of a constant quality.
[8] `http://www-igm.univ-mlv.fr/~unitex/`

```
<rss version="2.0">

<channel>
    <title>Paris Libre</title>
    <lastBuildDate>
    Tue, 4 Mar 2008 13:12:31 +0100
    </lastBuildDate>
    <item>
        <title>Un sport</title>
        <link>
        http://parislibre.lalibreblogs.be/
        archive/2008/03/04/un-sport.html
        </link>
        <pubDate>
        Tue, 4 Mar 2008 10:45:00 +0100
        </pubDate>
        <description>
        C'est un peu comme courir le marathon.
        Quand on est journaliste de presse...
        </description>
</item>
</channel>
</rss>
```

Figure 1: Example of RSS feed

```
<feed xmlns="http://www.w3.org/2005/Atom"
xml:lang="fr">

<title>Paris Libre</title>
<updated>2008-03-04T13:08:38+01:00
</updated>
<entry>
        <title>Un sport</title>
        <link rel="alternate" type="text/html"
        href="http://parislibre.lalibreblogs.be/
        archive/2008/03/04/un-sport.html" />
        <updated>
        2008-03-04T10:57:03+01:00
        </updated>
        <published>
        2008-03-04T10:45:00+01:00
        </published>
        <summary>
        C'est un peu comme courir le marathon.
        Quand on est journaliste de presse...
        </summary>
        </entry>
</feed>
```

Figure 2: Example of Atom feed

order to generate a dynamic corpus. Secondly, the user can register one or more search queries on his / her dynamic corpus. Search queries will be re-applied on the corpus every time it is updated and new concordances will be recorded for the user (results can be emailed, published for the user in a private RSS feed, or they can be viewed online).

2. RSS and Atom feeds

2.1. What are RSS and Atom ?

RSS is the acronym for Really Simple Syndication. As XML-based format, RSS is used to facilitate news publication on the Web and content interchange between websites. Atom is another standard built with the same objective.

Actually, most of the newspapers and blogging websites offer RSS and / or Atom-based news feeds to allow easy access to the recently published news articles. Each RSS / Atom file contains a list of articles recently published, often grouped by theme or category. Usually, these files do not contain full articles, but the title, the date of publication, a link to the full article available on the publisher website and a summary or a truncated text. On a regular basis (every day, every hour or even more frequently), the RSS / Atom feeds are updated with the new published content. The feeds are often organized by theme and / or in accordance with sections of the newspaper or the blog ("politics", "social", "nature", "editorial","regions", etc. for newspapers, or "my cats", "my friends" and "computational linguistics", etc. for blogs). Feeds can be also created for special hot topics like "Partition of Belgium" in the French-speaking press for example.

Figures 1 and 2 respectively show the basic structure of an RSS and an Atom feed. These XML-based structures are very simple. For RSS, it mainly consists of a "channel" which contains a list of "items" such as a title, a link, a description and a publication date.

An Atom feed is composed of more or less the same information, i.e. a "feed" which contains a list of "entries" comprising, among other things, a title, a link, a summary and information about the time of the last update.

2.2. RSS and corpora

Most of the time, RSS and Atom feeds contain little text in the "description" or "summary" field (even though some publishers incorporate the full article in the RSS / Atom feeds), but each "item" or "entry" has a link to the full article. Therefore, the link can be used to download the corresponding webpage (see section 3.5. and figure 3).

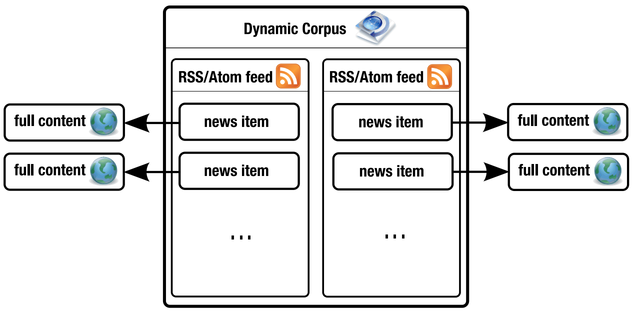


Figure 3: Dynamic Corpus using RSS / Atom feeds

As mentioned above, RSS and Atom feeds are frequently classified by genre, theme or category. Unfortunately, this classification is not standardized among newspapers, let alone blogs or other websites using feeds. In addition, no other indication is given about the classification criteria. However, if the classification fits the researchers needs, the RSS feeds can be used to build a specialized corpus.

A second characteristic of the RSS / Atom feeds is that they are frequently updated, which provides a continuous flow of data that can be easily collected in a corpus.

A third characteristic is that, by selecting "trusted" RSS or Atom sources (for example newspaper sources), the quality of the retrieval texts will be constant, which resolves the well known problem of Web corpus quality: when the Web is crawled for finding sources, the quality between documents may differ tragically.

For all these reasons and despite the problem of classification criteria, using RSS or Atom feeds probably represents one of the most interesting ways to build a corpus from the

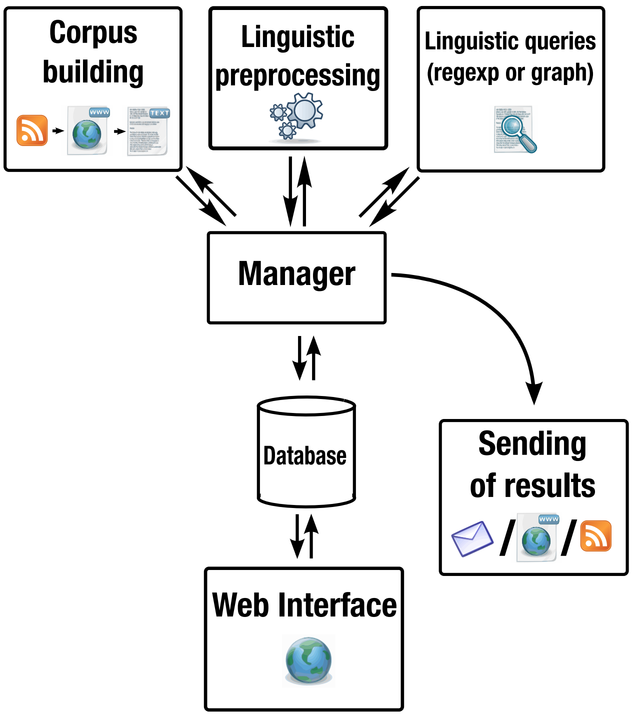


Figure 4: GlossaNet architecture

Web. But it also comes with some limitations since all of the Web content is not available through these feeds.

3. Architecture and process

3.1. Architecture

GlossaNet is made of three parts (figure 4):

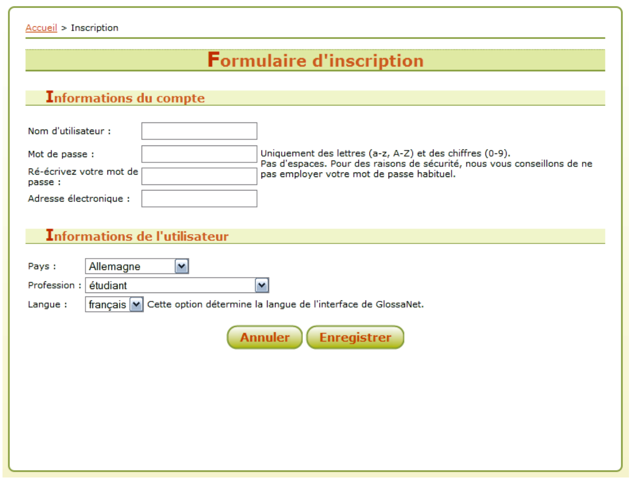
1. a Web interface to interact with users;
2. a database which contains data about each user, his / her corpora and his / her linguistic queries;
3. a server-side part composed of five servers organized in an asynchronous and distributed framework.

This architecture enables the use of these five servers on various computers and to split a server in two or more instances. This characteristic was necessary to prevent server overloads due to the ability given to the users to create many corpora from a virtually infinite number of RSS and Atom feeds.
We use the Perl Object Environment[9] (POE) framework to provide the asynchronous and distributed layer.

3.2. Process

The basic mechanism of GlossaNet is the following (figure 4). A Web interface (section 3.3.) allows the user to build one or more dynamic corpora from one or more RSS / Atom feeds (any feed available online can be selected). In addition, the user can create or use linguistic queries (made of regular expressions or graphs) and apply these queries to one or more previously built corpora. Information about user, corpora and linguistic queries are recorded into a database.

[9]`http://poe.perl.org/`


Accueil > Inscription

Formulaire d'inscription

Informations du compte

Nom d'utilisateur :		
Mot de passe :		Uniquement des lettres (a-z, A-Z) et des chiffres (0-9). Pas d'espaces. Pour des raisons de sécurité, nous vous conseillons de ne pas employer votre mot de passe habituel.
Ré-écrivez votre mot de passe :		
Adresse électronique :		

Informations de l'utilisateur

Pays :	Allemagne	
Profession :	étudiant	
Langue :	français	Cette option détermine la langue de l'interface de GlossaNet.

Figure 5: Login screen

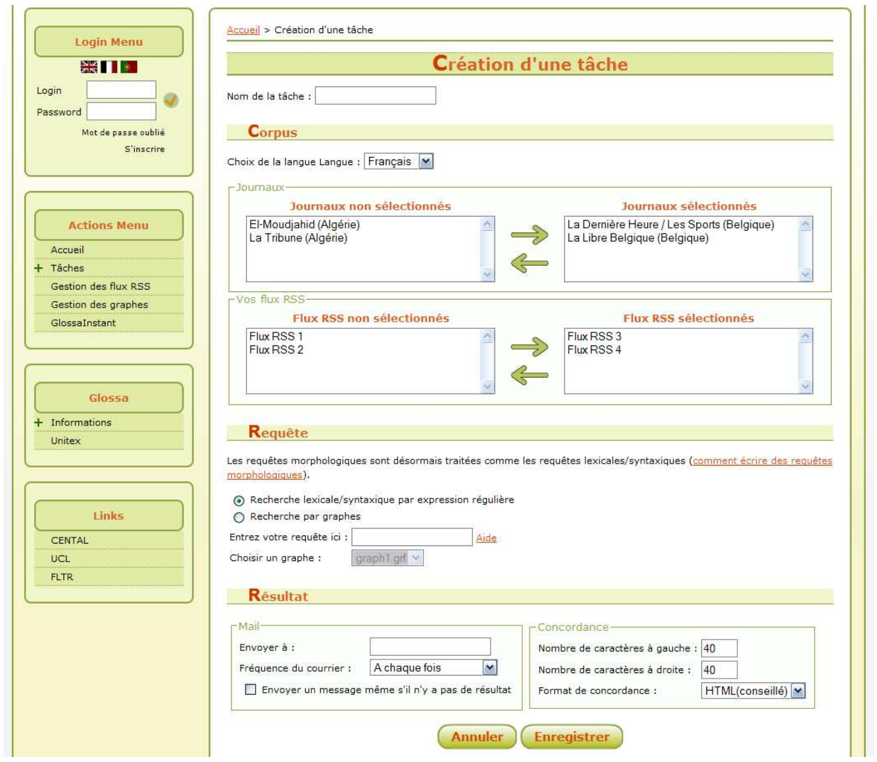
A central server named "Manager" (section 3.4.) detects when a new corpus is added into the database. When it happens, the "Manager" appends the RSS / Atom feeds of this corpus to the list of the feeds to download. On a regular basis, the list is sent to the "corpus building" server (section 3.5.) which downloads each feed, extracts the new entries ("items" in the RSS jargon) and, for each entry, gets and cleans the corresponding webpage. Clean texts are transmitted back to the "Manager" which stores them into the database.
Once a day, the "Manager" collects all the new texts of a corpus, concatenates and sends them to the "Linguistic preprocessing" server (section 3.6.). This server uses the Unitex software to tokenize and normalize the text. If linguistic resources are available for the corpus language, Unitex is also used to apply dictionaries and tag the corpus. Then a path to the preprocessed corpus is sent back to the "Manager".
When corpora are ready, the "Manager" checks the database to determine which linguistic queries have to be applied. The path to the corpus and each query (in the form of a finite state transducer) are posted to the next server which will retrieve all the concordances between the corpus and the query (section 3.7.). Results are sent back to the "Manager".
Once all the queries have been applied for a given user, results are transmitted to him (section 3.8.). Depending on the users preferences, results are sent on a daily or weekly basis.

3.3. Web interface

The Web interface provides several control pannels for creating and managing user's profile (figure 5), copora (figure 8) and linguistic queries (figure 7 and 6).

3.4. Manager

The manager role consists in supervising all the processes and sending the data needed by each server at a suitable time. It makes sure the results or useful information (for example about the unavailability of some RSS / Atom feeds) is sent to the user.



Accueil > Création d'une tâche

Création d'une tâche

Nom de la tâche :

Corpus

Choix de la langue Langue : Français

Journaux

Journaux non sélectionnés	Journaux sélectionnés
El-Moudjahid (Algérie)	La Dernière Heure / Les Sports (Belgique)
La Tribune (Algérie)	La Libre Belgique (Belgique)

Vos flux RSS

Flux RSS non sélectionnés	Flux RSS sélectionnés
Flux RSS 1	Flux RSS 3
Flux RSS 2	Flux RSS 4

Requête

Les requêtes morphologiques sont désormais traitées comme les requêtes lexicales/syntaxiques (comment écrire des requêtes morphologiques).

[✓ Recherche lexicale/syntaxique par expression régulière]
Recherche par graphes
Entrez votre requête ici : Aide
Choisir un graphe : graph1.grf

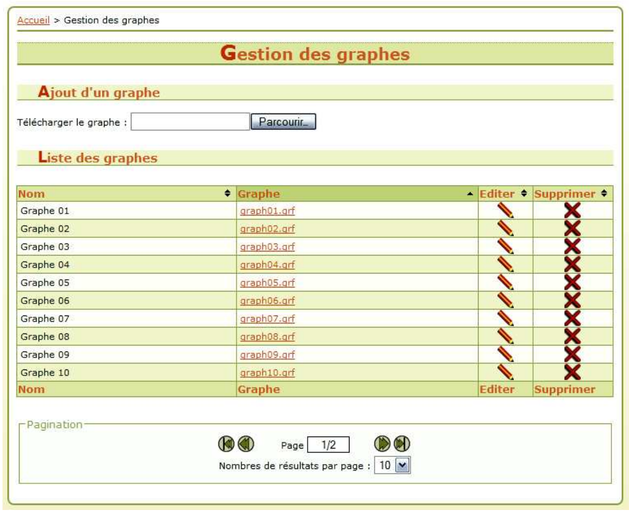
Résultat

Mail
Envoyer à :
Fréquence du courrier : A chaque fois
Envoyer un message même s'il n'y a pas de résultat

Concordance
Nombre de caractères à gauche : 40
Nombre de caractères à droite : 40
Format de concordance : HTML(conseillé)

Annuler Enregistrer

Figure 6: Task creation


Accueil > Gestion des graphes

Gestion des graphes

Ajout d'un graphe

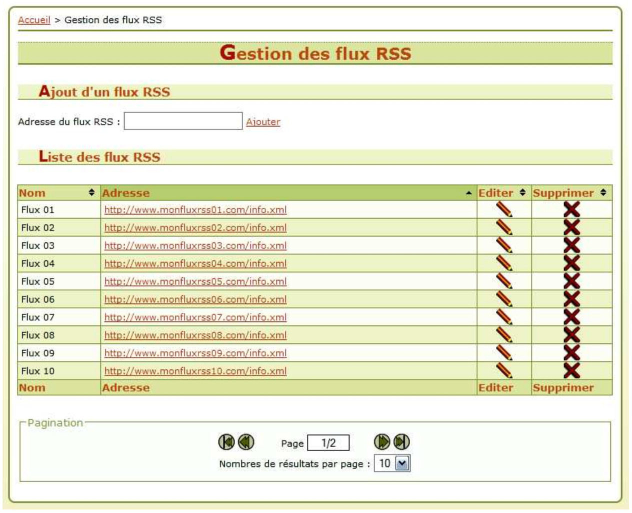
Télécharger le graphe : Parcourir...

Liste des graphes

Nom	Graphe	Editer	Supprimer
Graphe 01	graph01.grf	✎	✗
Graphe 02	graph02.grf	✎	✗
Graphe 03	graph03.grf	✎	✗
Graphe 04	graph04.grf	✎	✗
Graphe 05	graph05.grf	✎	✗
Graphe 06	graph06.grf	✎	✗
Graphe 07	graph07.grf	✎	✗
Graphe 08	graph08.grf	✎	✗
Graphe 09	graph09.grf	✎	✗
Graphe 10	graph10.grf	✎	✗
Nom	Graphe	Editer	Supprimer

Pagination
Page 1/2
Nombres de résultats par page : 10

Figure 7: Graphs manager


Accueil > Gestion des flux RSS

Gestion des flux RSS

Ajout d'un flux RSS

Adresse du flux RSS : Ajouter

Liste des flux RSS

Nom	Adresse	Editer	Supprimer
Flux 01	http://www.monfluxrss01.com/info.xml	✎	✗
Flux 02	http://www.monfluxrss02.com/info.xml	✎	✗
Flux 03	http://www.monfluxrss03.com/info.xml	✎	✗
Flux 04	http://www.monfluxrss04.com/info.xml	✎	✗
Flux 05	http://www.monfluxrss05.com/info.xml	✎	✗
Flux 06	http://www.monfluxrss06.com/info.xml	✎	✗
Flux 07	http://www.monfluxrss07.com/info.xml	✎	✗
Flux 08	http://www.monfluxrss08.com/info.xml	✎	✗
Flux 09	http://www.monfluxrss09.com/info.xml	✎	✗
Flux 10	http://www.monfluxrss10.com/info.xml	✎	✗
Nom	Adresse	Editer	Supprimer

Pagination
Page 1/2
Nombres de résultats par page : 10

Figure 8: Feeds manager

3.5. Corpus building

The "corpus building" server frequently receives new RSS / Atom feeds from the "Manager" to check. When the program detects that a RSS feed was updated, it retrieves the new items from the RSS file and, for each item, stores its title, description and URL. Then, it downloads each webpage corresponding to the URL, removes the boilerplate and sends the page to the "Manager" in text format.

During this process, two tasks must be considered: the boilerplate filtering and the identification of duplicated news.

3.5.1. Boilerplate removal

The main difficulty for building the corpus is to remove irrelevant text and links from webpages. This automatic boilerplate suppression is necessary to obtain clean texts which can be used with Unitex.

There are three differences with the Cleaneval task:

- At first, GlossaNet filters can rely on the metadata provided by the RSS feed (title and description). It is for instance helpful to locate the title in the HTML document because it indicates where the (relevant) text begins;

- Next, a lot of newspaper websites split long articles over two or more pages, but the RSS item refers only to the first page;
- Finally, some websites using RSS / Atom have common features, like the "print" button that opens a new webpage displaying the entire text in a simplified layout.

Considering all these specificities, we are developing two kinds of filters:

- specific filters for frequently used newspaper websites or for common blogging systems like Blogger, Wordpress, etc.;
- generic filters using RSS titles and descriptions, or frequent features like the "print" button.

As it is the case in most web corpus projects, filtering remains a challenge.

- Although it is possible to create specific filters for the most popular sources (like the main national newspapers or blog publishing systems), generic filters remain necessary for all other type of webpages.
- Since website layouts change over the time, specific filters must be continuously adapted.

3.5.2. Duplicated news

The system also needs a filter that prevents a single text from appearing more than once in the corpus. Several situations may lead to text duplicates:

- a text that was published earlier on a RSS feed can be broadcasted again for various reasons: that can happen simply by mistake or on purpose, for instance after small corrections (typos, etc.) or after the addition of a new paragraph to the text;
- on newspaper websites, it is common to find, next to a news article, links to other related articles. These links can consist in short titles or long paragraphs borrowed from the referred article;
- on blogs, several posts can be displayed on the same webpage. Each time a new post is added, the RSS feed is updated and when the Corporator crawler follows the link to get the full article, it retrieves not only the last article but also older articles presented on the same webpage.

As we can see, these difficulties go beyond the usual 'boilerplate removal'. Of course, if we do not address this problem, it could have important consequences on word frequencies in the corpus. Moreover, concordances built from this corpus will contain duplicates.

3.5.3. Corporator

The "corpus building" server corresponds to the new core of Corporator. This new version of the software will be released as a standalone version with a GUI and will include the improvements made for GlossaNet (boilerplate filters and similarities detection). Corporator will be distributed under an open source licence.

3.6. Linguistic preprocessing

The preprocessing step is mainly dependent on the Unitex software. The corpus is first normalized and tokenized and then, if linguistic resources are available for the corpus language, a dictionary lookup program is used to tag the text (Unitex provides linguistic resources for English, Finnish, French, German, Ancient Greek, Modern Greek, Italian, Korean, Norwegian, Polish, Portuguese from Brazil, Portuguese from Portugal, Russian, Serbian, Spanish and Thai).

3.7. Linguistic queries

If the corpus language is supported by Unitex, complex queries may be created, using word form, lemma, part of speech, morphological and semantic information.
Figure 9 shows a complex query using:

- a simple form ("to");
- a lemma ("be" between chevrons, i.e. all the inflectional forms of "be");
- parts of speech and morphological information ("<V:G>", i.e. any verb with an -ing form, and "<V:W>, i.e. any verb with an infinitive form);
- parts of speech and semantic information ("<N+conc>", i.e. any concrete noun).

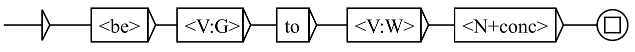


Figure 9: Example of a complex linguistic query

This query matches for example with the sentence "I am going to eat bread" but also with "He is going to get books". Otherwise, if the language is not supported, more simple queries using linguistic forms can be created (figure 10).

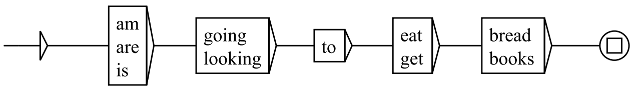


Figure 10: Example of a simple linguistic query

The "linguistic queries" server build a concordance and sends it to the "Manager". The characteristics of the concordance (sort order, length, number of occurrences, etc.) can be determined by the user in the Web interface.

3.8. Sending of results

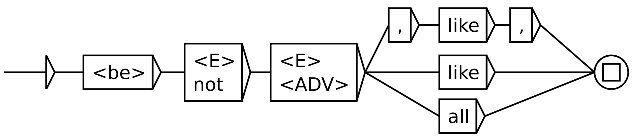
The user can also define how and how often results will be provided to him: on a daily or weekly basis; by e-mail, by RSS feed, or via the GlossaNet website interface.

4. Use cases

The use cases presented here are just a few examples of enquiries that can be made using GlossaNet.

4.1. Linguistics

Fairon and Singler (2006) used GlossaNet in their study of a particular type of quotative that occurs frequently in American Vernacular English and might be becoming part of the Standard English: (be) like. In order to evaluate how this quotative is spreading in written English, GlossaNet was used to monitor newspapers from various parts of the World. The Finite State Graphs search option facilitated the extraction of variants of this quotative.



4.2. Websites monitoring

If GlossaNet was initially conceived for linguists, it also proved to be useful for another category of users. We have indeed noticed that many users are more interested by ”information” than ”linguistic patterns”. Their queries contain keywords intended to collect news clippings. Once registered into the system, queries are reapplied every time corpora are updated. It is therefore possible to monitor websites or information sources. Finite state graphs offer a convenient way for representing many variants of keywords.

5. Conclusion

In this paper, we have described a new version of GlossaNet which represents a major update of the system. Now, this online application allows the user to build dynamic corpora from the Web using RSS and Atom feeds. Then, these dynamic corpora can be searched for linguistic patterns and results are presented under the form of a concordance that can be transmitted to the user by email, on a privative RSS feed or that can be read online in the user account. Four points should be highlighted:

1. The Web interface, which is the main entry point of GlossaNet, was entirely redesigned. This was necessary to improve ergonomics but also for integrating the new features related to the management of the RSS / Atom feeds.

2. The previous version of GlossaNet was limited to a series of approximately one hundred dynamic corpora (which were generated by a crawler bound to a preselected pool of newspaper websites). In this new version, the number of possible corpora is virtually infinite. Each user can design his / her own corpus by combining RSS / Atom feeds that are every day more numerous on the Web.

3. If GlossaNet corpora were formerly limited to newspapers, it is no longer the case: any site that offers content through RSS / Atom feeds can be used. It is easy to build thematic corpora (finance, recipes, sport, health, etc.) or corpora representing certain genres (blog, forum, etc.)

4. The system was completely rebuilt on improved software architecture. Thanks to the modularity of the new architecture, it is now easy to integrate new features or to bridge the system with other pieces of software. For instance, we use the system for generating flows of textual data that feed other applications (for corpus lexicography, information extraction, etc.)

6. References

F. Duclaye, F. Yvon, and O. Collin. 2003. Unsupervised incremental acquisition of a thematic corpus from the web. In *Proceedings of Natural Language Processing and Knowledge Engineering*. IEEE.

Cédrick Fairon and John V. Singler. 2006. I’am like, ‘Hey, it works’: Using GlossaNet to find attestations of the quotative (be) like in English-language newspapers. In A. Renouf and A. Kehoe, editors, *The Changing Face of Corpus Linguistics*, number 55, pages 325–337. Language and computers: Studies in Practical Linguistics, Amsterdam - New York.

Cédrick Fairon. 2006. Corporator: A tool for creating rss-based specialized corpora. In *Proceedings of the Workshop Web as corpus*, Trento. EACL.

William H. Fletcher. 2007. Implementing a BNC-Compare-able Web Corpus. In C. Fairon, H. Naets, A. Kilgariff, and G.-M. de Schryver, editors, *Building and Exploring Web Corpora*, volume 4, Louvain-la-Neuve. Cahiers du Cental.

Marianne Hundt, Nadja Nesselhauf, and Carolin Biewer, editors. 2007. *Corpus Linguistics and the Web, Language and computers studies in practical linguistics*, volume 59. Rodopi, Amsterdam - New York.

Adam Kilgarriff and Gregory Grefenstette. 2003. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3):333–348.

Igor Leturia, Antton Gurrutxaga, Iñaki Alegria, and Aitzol Ezeiza. 2007. CorpEus, a ‘web as corpus’ tool designed for the agglutinative nature of basque. In C. Fairon, A. Kilgariff, H. Naets, and G.-M. de Schryver, editors, *Building and Exploring Web Corpora*, number 4, Louvain-la-Neuve. Cahiers du Cental.

Mario Baroni and Silvia Bernardini, editors. 2006. *Wacky! Working papers on the Web as Corpus*. GEDIT, Bologna.

Cédrick Fairon. 1999. Parsing a web site as a corpus. In Cédrick Fairon, editor, *Analyse lexicale et syntaxique: Le système INTEX, Lingvisticae Investigationes*, volume XXII, pages 327–340. John Benjamins Publishing, Amsterdam/Philadelphia.

Sébastien Paumier. 2003. *De la reconnaissance de formes linguistiques à l’analyse syntaxique*. Ph.D. thesis, Université de Marne-la-Vallée.

Antoinette Renouf. 1993. A word in time: first findings from the investigation of dynamic text. In J. Aarts, P. de Haan, and N. Oostdijk, editors, *English Language Corpora: Design, Analysis and Exploitation*, pages 279–288, Amsterdam. Rodopi.

Antoinette Renouf. 2003. Webcorp: providing a renewable energy source for corpus linguistics. In S. Granger and S. Petch-Tyson, editors, *Extending the scope of corpus-based research: new applications, new challenges*, pages 39–58, Amsterdam. Rodopi.

Collecting Basque specialized corpora from the web: language-specific performance tweaks and improving topic precision

I. Leturia, I. San Vicente, X. Saralegi, M. Lopez de Lacalle
Elhuyar Fundazioa, R&D
Zelai Haundi kalea, 3. Osinalde Industrialdea, 20170 Usurbil. Basque Country
E-mail: {igor, inaki, xabiers, maddalen}@elhuyar.com

Abstract

The *de facto* standard process for collecting corpora from the Internet (with a given list of words, asking APIs of search engines for random combinations of them and downloading the returned pages) does not give very good precision when searching for texts on a certain topic. And this precision is much worse when searching for corpora in the Basque language, due to certain properties inherent in the language and in the Basque web.
The method proposed in this paper improves topic precision by using a sample mini-corpus as a basis for the process: the words to be used in the queries are automatically extracted from it, and a final topic-filtering step is performed using document-similarity measures with this sample corpus. We also describe the changes made to the usual process to adapt it to the peculiarities of Basque, alongside other adjustments to improve the general performance of the system and quality of the collected corpora.

1. Introduction

1.1 Motivation

Basque needs corpora more than many other bigger languages, as its standardisation began only very recently. And above all it is in need of specialized corpora, because terminology is the area with least *de jure* normalization. The only specialized corpus in Basque is the ZT Corpus (Areta et al., 2007), a corpus on Science and Technology that is a very valuable resource, but which does not fulfil all the needs of Basque for many reasons: it does not include texts on social sciences; it is divided into very general topics, so it is impossible to search texts dealing exclusively with anatomy or computer sciences, for example; and it is not kept up-to-date.
But building specialized corpora the classical way, i.e. out of printed texts, is normally a very costly process, and Basque is not exactly what we would call a language with plenty of economic resources. So we embarked on a project to build a system to collect specialized corpora in Basque, using the Internet as a source.

1.2 Low topic precision

Before BootCaT (Baroni & Bernardini, 2004) came onto the scene, collecting corpora on a certain topic from the web was mainly done by crawling sites related to the topic and subsequently filtering the pages using some sort of topic classifier, as in (Chakrabarti et al., 1999). BootCaT introduced a new methodology: give a list of words as input, query APIs of search engines for combinations of these seed words and download the pages. This methodology has in some cases been used to build big general corpora (Sharoff, 2006), but for collecting smaller specialized corpora, it has become the *de facto* standard. Since then, the subsequent topic-filtering stage has been left aside, as it has been assumed that the search for words on a topic suffices for obtaining the corresponding texts on it alone.
And yet there are not many studies on the precision obtained by the word-list method, and the results of the few that have been done give us reason to believe that a topic-filtering stage *is* necessary: in the aforementioned paper on BootCaT, an evaluation was performed on a small sample of 30 texts of each of the two corpora collected, and a third of them proved to be uninformative or unrelated to the topic. Depending on the application, this amount of noise in the corpora can be considered to be unacceptable.

1.3 Problems with Basque

Obtaining an increase in precision is even more important in our case, since some features of the Basque language and the Basque web cause topic precision to fall dramatically when using the standard methodology, as the experiment we describe next shows.
We used BootCaT to gather some small corpora on geology and computer sciences: we made 20 queries with 2, 3 and 4 n-gram combinations and downloaded the first 10 pages. Then we looked at all of the documents to see if they were appropriate for the corpus (desired topic and language, informative, not duplicates, etc.), and the results we obtained are shown in Table 1.

Topic	n	Total		Appropriate			
		Docs	Words	Docs	%	Words	%
Comp. Sci.	2	65	1,282,001	33	50.77	289,259	22.56
	3	60	2,853,710	25	41.67	406,426	14.24
	4	48	2,321,888	22	45.83	355,254	15.30
Geol.	2	85	2,526,820	13	15.29	379,131	15.00
	3	31	1,606,312	8	25.81	184,371	11.48
	4	3	195,246	2	66.67	101,731	52.10
Total		**292**	**10,785,977**	**103**	**35.27**	**1,716,172**	**15.91**

Table 1: BootCaT topic precision results

The percentage of each of the reasons for a document to be considered inappropriate are shown below:

Topic	n	Reason					
		Wrong topic		Wrong language		Other[1]	
		Docs	%	Docs	%	Docs	%
Comp. Sci.	2	21	65.63	5	15.63	6	18.75
	3	17	48.57	11	31.43	7	20.00
	4	16	61.54	4	15.38	6	23.08
Geol.	2	31	43.06	26	36.11	15	20.83
	3	4	17.39	2	8.70	17	73.91
	4	0	0.00	0	0.00	1	100.00
Total		**89**	**47.09**	**48**	**25.40**	**52**	**27.51**

Table 2: Kinds of inappropriate pages

This study is by no means exhaustive, but our objective was not to quantify the loss in precision exactly. We were just aiming to show that topic precision and general quality of a corpus obtained with BootCaT are much worse when looking for corpora in Basque. Besides, we must take into account that in this experiment we did not perform the bootstrapping process of extracting the words out of the downloaded pages to get new ones; if we had done so, the pages downloaded in the next stage would most likely have yielded even worse topic precision.
The reasons for this are diverse. One is that no search engine offers the possibility of returning pages in Basque alone, so when looking for technical words (as is often the case with specialized corpora), it is very probable that they exist in other languages too, and that the queries return many pages that are not in Basque. Another reason is that the Basque web is not as big as those of other languages, and this means that the only pages existing for certain queries with combinations of various words are very long documents (blogs, magazines in PDF format, etc.) where the desired topic is just a small part of the whole document, or where the words searched for are simply found by chance in different parts of the long document. This phenomenon is exacerbated by the fact that Basque is a morphologically rich language and any lemma has many different word forms, so looking for a word's base form alone, as search engines do, brings fewer results.

2. Our approach

2.1 System objectives and description

The objective of our project is to develop a system to obtain specialized corpora in Basque from the Internet, aimed at improving topic precision and solving Basque-specific problems.
In order to try to improve topic precision, our method takes, as a starting point, a sample mini-corpus of documents on the topic, instead of a list of words. This mini-corpus has two uses: first, the list of keywords to be used in the queries is automatically extracted from it; second, it is used to filter the downloaded documents according to topic by using document-similarity techniques (Lee et al., 2005).
And considering the inferior quality that is obtained when Basque is involved, we also try to improve this by using techniques and methods known to obtain better performances with Basque IR, as well as other little adjustments to the general process.

2.2 Evaluation corpora

In order to evaluate and measure the improvements of our system, we built some corpora by putting the system into practice. We chose the same two topics with which we evaluated the performance of BootCaT with Basque, i.e. computer sciences and geology. We built three sample corpora of each topic, consisting of 10, 20 and 30 documents, the two smaller ones made up of documents chosen at random out of the bigger one. For each of these six sample mini-corpora, we automatically extracted the word lists and revised them manually. Then out of each of the six lists we built three different corpora using 2-, 3- and 4-word combinations in the queries. These are the final sizes of the 18 corpora collected:

Topic	Sample size	n		
		2	3	4
Computer Sciences	10	758	274	43
	20	745	256	56
	30	674	176	52
Geology	10	97	22	3
	20	125	14	3
	30	146	27	2

Table 3: Sizes of the collected corpora

These are the corpora that have been used for the various evaluations and partial results mentioned in the next sections, which describe the method and system developed.

3. Automatic keyword extraction from a sample mini-corpus

The basis of our system is a sample mini-corpus of documents on the target topic, which will have to be collected manually. This sample will be used for extracting the word list for the queries and in the final topic-filtering stage as well, so the criteria when collecting the sample is that it should be as heterogeneous as possible and cover as many different subjects of the topic as possible. According to our experiments, as few as 10 documents may be enough for a very specialized topic, but more might be needed for more general topics.
The words to be used in the queries are automatically

[1] Duplicate, part of a much bigger text including other topics, spam, etc.

extracted from this sample corpus, thus avoiding the work of finding appropriate words on the topic. This is usually more laborious than finding texts on the topic, at least for Basque, because there are many topics for which there are still no specialised dictionaries or glossaries.
The keyword extraction method is based on the work previously performed in our team in the DokuSare project (Saralegi & Alegria, 2007). The mini-corpus is automatically lemmatised and POS-tagged, and then the most significant nouns, proper nouns, adjectives, verbs, entities and multiword terms are extracted by means of Relative Frequency Ratio or RFR (Damerau, 1993), which we calculate by dividing the relative frequency of a word in the specialized mini-corpus by the relative frequency of the word in a general corpus, and applying an empirically determined threshold. The general corpus we use is a 450,000-word corpus consisting of newspaper articles.
The extracted list consists of (mostly) topic-specific words, but some of them might be too specific or rare, as the RFR measure tends to promote on excess words that are not present in the general corpus. The usual way to avoid this is to use a raw frequency threshold to choose the candidate words for the RFR measure, but this is not so easy to apply in our case, because the sample mini-corpora are small (on purpose). And in any case, these undesired words are usually removed in the manual revision stage explained in the next paragraph.
In order to maximise the performance of the queries, the extracted list is revised manually. Too specific or too local proper nouns, too general words and polysemous words that have other meanings in other areas are removed. Normally, the total process of obtaining the mini-corpus and manual revision of the word list is still less costly than trying to obtain a word list, because of the absence of specialised dictionaries explained above.

4. Optimizing for Basque

It is a well-known fact that search engines do not work well with many non-English languages (Bar-Ilan & Gutman, 2005). In 2007 there was even a SIGIR workshop on the subject (Lazarinis et al., 2007). Specifically, performance of search engines for Basque is very poor, mostly due to the rich morphology of the language and to the fact that no search engine can restrict its results to pages in Basque alone, and these are the main reasons for the poor performance of BootCaT with Basque. But search engines can be made to work much better with Basque by applying the techniques known as morphological query expansion and language-filtering words, as shown in the projects CorpEus (Leturia et al., 2007 a) and EusBila (Leturia et al., 2007 b).

4.1 Methodology description

In Basque, a lemma can form very many different surface forms, so just looking for the exact base form does not return all the pages that actually contain occurrences of a word. This is true, to a greater or lesser extent, for many other languages too, but while search engines usually apply some sort of stemming for major languages, this does not happen in the case of Basque. Morphological query expansion, also called Frequent Case Generation in some other works (Kettunen, 2007), consists of asking the search engine not only for the lemma of a word, but also for various different word forms of the lemma, which are obtained by morphological generation, within an OR operator. In order to maximize recall, the most frequent word forms are used. In the case of Basque, the morphological generation is done using a tool developed by the IXA Group of the University of the Basque Country, and recall is improved by up to 60% in some cases. The anticipated effect of this increase in recall in our project is a smaller percentage of big PDFs in the downloaded documents, and more pages downloaded in some topics with 4-word combinations in the queries.
The other problem is caused by the fact that no search engine offers the possibility of restricting its results to pages in Basque. The result is that when searching for technical words, short words or proper nouns, many non-Basque pages are returned, since those words may be used in other languages too. The language-filtering words method, consisting of adding the most frequent Basque words to the queries within an AND operator, improves language precision from 15% to over 90%. There is also a non-negligible loss in recall, because pages not containing the filtering words may be left out, but these are normally short and so uninteresting for corpora. Besides, the practical effect in a project like ours is actually a gain in recall: where some normal searches would return many non-Basque pages that would afterwards be filtered out in the language- or size-filtering step and yield few or even no results, with the language-filtering method, however, we would obtain pages in Basque.
We are aware that BootCaT does give the option of language-filtering by means of a list of frequent words in the language, but that filtering is done after downloading the pages. If filtering is conducted that way, many searches for words that exist in other languages will bring no results in Basque and all the pages will be filtered out, thereby wasting bandwidth, time and calls to the API of the search engine.
However, the language-filtering words method ensures that almost all of the pages downloaded will have Basque in them, but not that they will be exclusively in Basque. Due to the Basque language being co-official with Spanish in the Basque Autonomous Community and in some parts of the Charter Community of Navarre, there are many bilingual web pages and documents, e.g. many local and regional government gazettes. Including those bilingual documents in the corpora would cause too much noise, but not including them means we could lose many interesting documents.
In order to solve this problem, we use LangId, a language identifier developed by the IXA Group of the University of the Basque Country, applied at paragraph level. This does not mean that we remove every non-Basque paragraph; if we did, we could also remove some short quotes important for the understanding of a text. As our

intention is to eliminate sufficiently large amounts of noise, we remove sequences of non-Basque paragraphs that exceed 10% of the length of the document, and individual paragraphs only if the total amount of the language of the paragraph in the document exceeds 40%.
But working with a minority language like Basque does not always mean more difficulties. Spam and porn filtering, for example, turn out to be very easy. Since as big an audience as possible is usually targeted, there is practically no spam or porn in Basque, so language filtering does the job perfectly.

4.2 Evaluation

The effectiveness of the language-filtering words method for obtaining pages exclusively in Basque from the queries had already been proven in the aforementioned CorpEus and EusBila projects, and the results achieved in this experiment confirm it (only 2.46% of documents retrieved by search engines did not contain any Basque).
As to the language identifier that is applied at paragraph level, it removes supposedly non-Basque parts from 28% of the downloaded documents. Due to the amount of work this entails, we did not evaluate the recall of this step (that is, we did not look at all the documents to see how many non-Basque parts had been left out). But we did look at a sample of the cleaned documents to see if the removed parts were really non-Basque, and although we did not measure it quantitatively, the performance can be considered to be very good.
The morphological query expansion method improves recall in Basque IR, so the number of long PDFs should go down when it is used, which in fact turns out to be the case: in the BootCaT experiments, almost 72% of the documents were PDFs, but now only 13% are PDFs in the computer sciences corpus and 41% in the geology corpus; and the average document length also went down by a 25%.

5. Other improvements

5.1 Description

Filtering documents by length is an effective way of reducing noise (Fletcher, 2004). In our case, we reject documents the length of which after conversion to plain text is under 1,000 characters or over 100,000 characters. That is to say, we remove documents that are roughly shorter than half a page (not enough continuous text to be interesting) or longer than 50 pages (not likely to be on a specialized topic).
Boilerplate removal is another key issue in this project, not only because boilerplate adds noise and redundancy to corpora, but also because it can affect subsequent stages (near-duplicate detection, topic filtering, etc.). For boilerplate removal, we use Kimatu (Saralegi & Leturia, 2007), a system developed in our team that scored well (74.3%) in the Cleaneval competition (Baroni et al., 2008).
We have also included a near-duplicate detection module based on Broder's shingling and fingerprinting algorithm (Broder, 2000). We have prioritised non-redundancy over recall and have rejected not only almost equal documents, but all that have a level of coincidence of over 50% with some other one. The reason for this is that nowadays many on-line news sites and blogs have a main page with some news that changes over time with the addition of new items, but at certain times many news items may coincide. Also, they often show the list of posts related to a category or a tag, and these can have many articles in common too.
Broder's earlier works on near-duplicate detection also dealt with containment (Broder, 1997). But while near-duplicate detection was improved enough to require a very small set of features and very fast processing (as much as to be used at web level), containment detection did not attain this level of optimization. However, we think containment detection is important: again, many blogs and news sites have a main page or section where many individual articles that also have their own URL are contained. And near-duplicate detection methods do not detect containment. So we took up again Broder's method for containment detection, which on our scale is perfectly usable.

5.2 Evaluation

31% of the downloaded documents were filtered out because they were too long, and 10% and 3% of the computer sciences and geology corpora, respectively, because they were too short. By taking a look at the rejected ones, we confirmed that the filter achieves its goal, as the great majority were uninteresting, general or multi-topic documents.
The near-duplicates filter removes 5% of the downloaded documents, and the containment filter another 5%. In the small evaluation we made for precision we found no errors; recall was not evaluated.

6. Topic precision obtained with the improvements

All the improvements made to the process, both Basque-specific or general, that have been described above, have already been evaluated individually. But the aim of each and every one of them is to enhance the quality of the corpora obtained, mainly regarding topic precision. So it is imperative to evaluate the collected corpora by looking at topic precision, to see if the performance tweaks for Basque and the other improvements had any effect and actually improved the BootCaT results. We took a random sample of 30 documents out of each of the 18 corpora built for the evaluation, and saw whether they belonged to the desired topic or not. Due to their small size (see Table 3), all the documents of n=4 and of geology n=3 were checked. These were the results we obtained:

Topic	Sample size	n			
		2	3	4	Avg.
Computer Sciences	10	46.66%	63.33%	82.93%	64.31%
	20	50.00%	66.66%	70.00%	62.22%
	30	53.33%	63.33%	63.89%	60.19%
	Avg.	**50.00%**	**64.44%**	**72.27%**	**62.24%**
Geology	10	53.33%	40.91%	100.00%	64.75%
	20	56.66%	64.29%	100.00%	73.65%
	30	46.66%	56.76%	100.00%	67.81%
	Avg.	**52.22%**	**53.98%**	**100.00%**	**68.74%**
Avg.		**51.11%**	**59.21%**	**86.14%**	**65.49%**

Table 4: Topic precision before topic-filtering stage

In view of these results, we can conclude that our little improvements, all together, do yield much better topic precision results when looking for corpora in Basque, and are not far short of the baseline for other languages.

7. Topic filtering

As we have pointed out already, this topic precision can be considered insufficient in many cases, and another aim of our project was to try to improve it.

7.1 Methodology description

Topic or domain detection is usually approached through machine learning methods. While these can obtain good performances, they also have their drawbacks: they need fairly big training sets and times, they are trained for a fixed set of topics, etc.
Our approach to this matter has been to try to use a small set of sample documents (i.e. the sample mini-corpus out of which the keywords are extracted) and document similarity measures based on keyword frequencies to say whether a document belongs to a topic or not. According to Sebastiani (2002), topic or domain detection can be done using keywords.
These kinds of document similarity measures are usually applied between two documents to see if they deal with the same or a similar subject, as in the aforementioned DokuSare project. But in our case, we have a document and a corpus, which are elements of different scale, and also the level of similarity to be handled is somehow smaller, since we just need to measure whether they coincide on the topic.
They have also been applied to measure similarity between two corpora (Kilgarriff & Rose, 1998), which is also a little different from our case.
However, the general idea of our project is very similar to that of DokuSare: to represent both the documents to be filtered and the sample mini-corpus through a set of features based on keywords, and to use some similarity measure to see if they share the same topic.
But as we said, we are going to measure the similarity between elements of a different scale, i.e. a document and a set of documents. So we have tried by measuring the similarity between a document and the mini-corpus directly, and also by measuring the similarity of a document with each of the documents of the sample mini-corpus, and taking the maximum.
For the representation of both the downloaded documents and the sample corpus or each of the documents of the sample corpus, we use the bag-of-words paradigm, which models the most significant keywords, i.e. nouns, proper nouns, adjectives and verbs, in a vector. The words are selected and weighted by a certain frequency measure. We have tried two: the aforementioned RFR and a new one we have defined as Relative Rank Ratio or RRR.
We felt that this new frequency measure fitted Zipf's law better (Zipf, 1949) and could be better suited for comparing documents of different sizes. It is defined as the ratio between the relative frequency-ranking of a word in the document or corpus involved, and the relative frequency-ranking of a word in a general corpus. This is its exact formula:

$$RRR(w_i, dok) = \frac{1 - \frac{Freq.Rank(w_i, dok)}{RankCount(dok)+1}}{1 - \frac{Freq.Rank(w_i, gen.corp.)}{RankCount(gen.corp.)+1}}$$

We have observed that this measure works better if we apply some sort of smoothing to words that are not found in the general corpus, because otherwise the formula gives them very high values, and they are often rare words or spelling errors that worsen the results.
For measuring the similarity we use the cosine, the most extended way to measure the similarity between documents represented in the vector space model.
So for comparing two documents *x* and *y*, w_i ($i \in \{1, n\}$) being the keywords present in any of the two, we prepare the vectors ($x_1, x_2, \dots x_n$) and ($y_1, y_2, \dots y_n$), where x_i and y_i are the RFR or RRR ratios of the word w_i in the documents *x* and *y* respectively, and then we calculate the cosine between the two, which is specified as follows:

$$\cos(x, y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x^2} \sqrt{\sum_{i=1}^{n} y^2}}$$

7.2 Evaluation

As an evaluation experiment, we took the corpora collected for the evaluation, and out of each of them we manually chose a sample of appropriate documents and another one of inappropriate ones, each made up of 15 documents (if the corpus was large enough). Then we applied the aforementioned similarity measures to these development datasets in the two ways explained, and for each of the 18 corpora we obtained charts like those shown in figures 1 to 4. More precisely, these correspond to the average of the geology and computer sciences corpora collected using 20-document sample mini-corpora and using 2-word combinations.

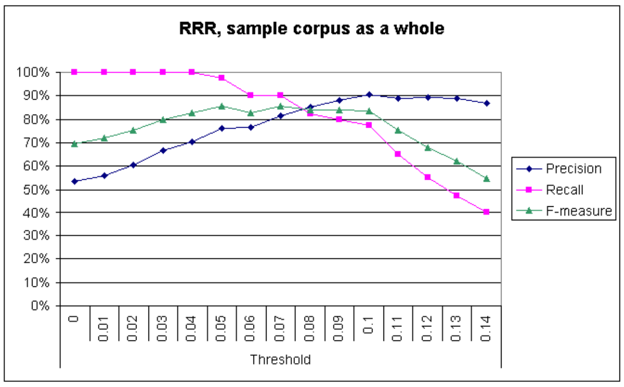


Figure 1: Results with RRR measure, taking the sample mini-corpus as a whole

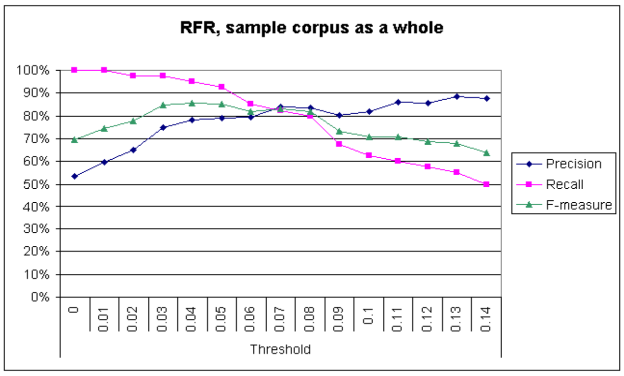


Figure 2: Results with RFR measure, taking the sample mini-corpus as a whole

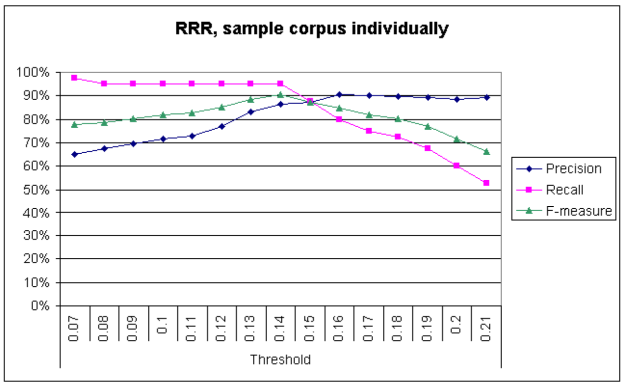


Figure 3: Results with RRR measure, taking each document of the sample mini-corpus individually

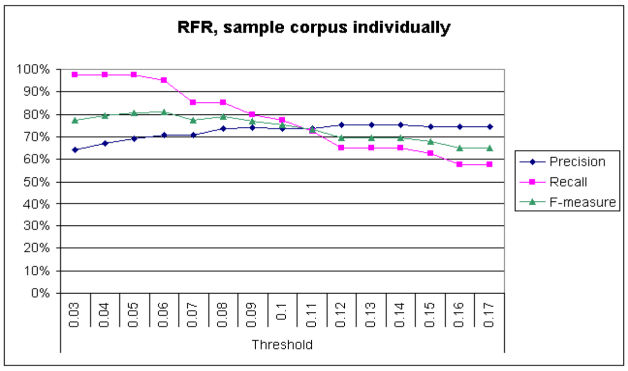


Figure 4: Results with RFR measure, taking each document of the sample mini-corpus individually

It is impossible to show here all the charts for all of the 18 corpora and the different averages. Instead, we will explain the conclusions we have drawn from their observation.

Since our primary objective is to improve topic precision, we are interested in finding a measure and a threshold that will maximise the F-measure but which will prime precision. This is usually obtained somewhere to the right and near the crossing point of the precision and recall series. On average, the highest crossing points are found with the RRR measure when compared with each document of the sample corpus individually.

We have also tried to improve the results by combining more than one of them. For example, we have tried first measuring the similarity with the whole sample mini-corpus and, if the measure is not above the threshold, trying again with the one-by-one comparison. But the only effect of this was that more documents were accepted, both good and bad ones, thus augmenting recall but at the cost of precision.

If we are to significantly improve the baseline of 66% topic precision, we would need a precision of 80% minimum, without a great loss in recall. The RRR-individual method can obtain precision and recall above 80% for most of the corpora, but with different thresholds. In other words, there is no threshold that maximises F-measure and obtains a precision above 80%, and which works for all of the corpora.

In any case, for higher thresholds we usually obtain a higher precision (at least until it falls at some point), so it is possible to assure high precision (80-90%) if recall is not an issue. This might not be the case of Basque, since, as we have observed before, some topics already yield very small corpora and a recall of 60-40% may not be acceptable. But for English or other bigger languages, with the RRR-individual method and a threshold from 0.18 to 0.20 we can obtain a topic precision of 80-90%.

8. Conclusions

The series of improvements to the standard method for collecting specialized corpora from the Internet that we propose, and which are intended to improve the otherwise disastrous performance when looking for documents in Basque, seem to achieve their purpose, since our results are similar to the baseline of other languages. We have also observed that, without any filtering, the best topic precision results are obtained, logically, with 4-word queries, but due to the reduced amount of Basque content on the Internet, corpora obtained on some topics are extremely small with these kinds of queries. And there is no way one can know *a priori* which topics will be affected, so it is better to use 3-word queries, even though the topic precision obtained will be a little lower.

We have also proposed a method for improving the topic precision for any language, based on a sample mini-corpus, automatic extraction of words for the queries and easily computable document similarity measures. In particular, we have shown that it is possible to attain a high precision (80-90%) using the RRR measure and the cosine to compare the documents of the corpus with each document of the sample corpus and taking the maximum, and applying a high enough threshold. But there is also a non-negligible loss in recall, which might be an issue at least for Basque. However, adding the initial word extraction and the final topic filtering as new optional modules to BootCaT could be very interesting.

However, there is an important aspect to point out regarding this method. Obtaining high topic precision does not imply that the corpus obtained will be highly representative of the universe. In fact, since we are filtering by applying similarity measures using the documents of the sample mini-corpus, if this is not wide enough, that is, if not all the subareas of the topic are represented there, we might be missing areas without ever knowing it. So the quality and heterogeneity (and also size) of the sample mini-corpus is a key issue in the method proposed. But it is not easy to say what is a minimum or optimum size of the sample mini-corpus to assure good representativeness, since it greatly depends on whether the topic is very specialised, or quite general, etc. This alone could be a matter for another paper.

9. References

Areta, N., Gurrutxaga, A., Leturia, I., Alegria, I., Artola, X., Diaz de Ilarraza, A., Ezeiza, N., Sologaistoa, A. (2007). ZT Corpus: Annotation and tools for Basque corpora. In Proceedings of *Corpus Linguistics 2007*. Birmingham, UK: University of Birmingham.

Bar-Ilan, J., Gutman, T. (2005). How the search engines respond to some non-English queries? *Journal of Information Science*, 31(1), pp. 13--28.

Baroni, M., Bernardini, S. (2004). BootCaT: Bootstrapping corpora and terms from the web. In *Proceedings of LREC 2004*. Lisbon, Portugal: ELDA, pp. 1313--1316.

Baroni, M., Chantree, F., Kilgarriff, A., Sharoff, S. (2008). Cleaneval: a competition for cleaning web pages. In *Proceedings of LREC 2008*. Marrakech, Morocco: ELDA.

Broder, A.Z. (2000). Identifying and filtering near-duplicate documents. In *Proceedings of Combinatorial Pattern Matching: 11th Annual Symposium*. Montreal, Canada: Springer, pp. 1--10.

Broder, A.Z. (1997). On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences 1997*. Los Alamitos, CA: IEEE Computer Society, pp. 21--29.

Chakrabarti, S., van der Berg, M., Dom, B. (1999). Focused crawling: a new approach to topic-specific web resource discovery. In *Proceedings of the 8th International WWW Conference*. Toronto, Canada: University of Toronto, pp. 545--562.

Damerau, F.J. (1993). Generating and evaluating domain-oriented multi-word terms from texts. *Information Processing & Management*, 29, pp. 433--447.

Fletcher, W.H. (2004). Making the web more useful as a source for linguistic corpora. In U. Connor & T. Upton (Eds.), *Corpus Linguistics in North America 2002*. Amsterdam, The Netherlands: Rodopi.

Kettunen, K. (2007). Managing keyword variation with frequency based generation of word forms in IR. In *Proceedings of NODALIDA Conference*. Tartu, Estonia: University of Tartu, pp. 318--323.

Kilgarriff, A., Rose, T. (1998). Measures for corpus similarity and homogeneity. In *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing*. Granada, Spain: ACL SIGDAT, pp. 46--52.

Lazarinis, F., Vilares, J., Tait, J. (2007). Improving non-English web searching (iNEWS07). *ACM SIGIR Forum*, 41(2), pp. 72--76.

Lee, M.D., Pincombe, B., Welsh, M. (2005) An empirical evaluation of models of text document similarity. In *Proceedings of CogSci2005*. Stresa, Italy: Earlbaum, pp. 1254--1259.

Leturia, I., Gurrutxaga, A., Alegria, I., Ezeiza, A. (2007). CorpEus, a 'web as corpus' tool designed for the agglutinative nature of Basque. In *Building and exploring web corpora, Proceedings of the 3rd Web as Corpus workshop*. Louvain-la-Neuve, Belgium: Presses Universitaires de Louvain, pp. 69--81.

Leturia, I., Gurrutxaga, A., Areta, A., Alegria, I., Ezeiza, A. (2007). EusBila, a search service designed for the agglutinative nature of Basque. In *Proceedings of Improving non-English web searching (iNEWS'07) workshop*. Amsterdam, The Netherlands: SIGIR, pp. 47--54.

Saralegi, X., Alegria, I. (2007). Similitud entre documentos multilingües de carácter científico-técnico en un entorno web. *Procesamiento del Lenguaje Natural*, 39, pp. 71--78.

Saralegi, X., Leturia, I. (2007). Kimatu, a tool for cleaning non-content text parts from HTML docs. In *Building and exploring web corpora, Proceedings of the 3rd Web as Corpus workshop*. Louvain-la-Neuve, Belgium: Presses universitaires de Louvain, pp. 163--167.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1), pp. 1--47.

Sharoff, S. (2006). Creating general-purpose corpora using automated search engine queries. In M. Baroni & S. Bernardini (Eds.), *WaCky! Working papers on the Web as Corpus*. Bologna, Italy: Gedit.

Zipf, G.K. (1949). *Human behavior and the principle of least effort*. Cambridge, MA: Addison-Wesley.

Introducing and evaluating ukWaC, a very large web-derived corpus of English

Adriano Ferraresi,* Eros Zanchetta,* Marco Baroni,† Silvia Bernardini*

* SITLeC – University of Bologna (Forlì)
Corso Diaz 64, 47100 Forlì – Italy
adriano@sslmit.unibo.it, eros@sslmit.unibo.it, silvia@sslmit.unibo.it

† CIMeC – University of Trento
Corso Bettini 31, 38068 Rovereto (TN) – Italy
marco.baroni@unitn.it

Abstract

In this paper we introduce ukWaC, a large corpus of English constructed by crawling the `.uk` Internet domain. The corpus contains more than 2 billion tokens and is one of the largest freely available linguistic resources for English. The paper describes the tools and methodology used in the construction of the corpus and provides a qualitative evaluation of its contents, carried out through a vocabulary-based comparison with the BNC. We conclude by giving practical information about availability and format of the corpus.

1. Introduction

This article introduces ukWaC, a very large (>2 billion words) corpus of English, and presents an evaluation of its contents. UkWaC was built by web crawling, contains basic linguistic annotation (part-of-speech tagging and lemmatization) and aims to serve as a general-purpose corpus of English, comparable in terms of document heterogeneity to traditional "balanced" resources. Since the aim was to build a corpus of British English, the crawl was limited to the `.uk` Internet domain. The corpus is, to the best of our knowledge, among the largest resources of its kind, and the only web-derived, freely available English resource with linguistic annotation. It was created in 2007 as part of the WaCky project, an informal consortium of researchers interested in the exploration of the web as a source of linguistic data.

The aims of this article are to introduce ukWaC to the community of linguistic researchers, to describe the procedure that was followed in constructing it and to provide some preliminary evaluation of its contents.

The article is structured as follows: in Section 2. we outline the corpus creation process. In Section 3. we carry out a vocabulary-based comparison between ukWaC and the British National Corpus (BNC), which sheds light on the main differences between the two corpora. Section 4. deals with issues related to format and availability of the corpus. Finally, Section 5. briefly discusses previous work on web corpora, and Section 6. hints at what we consider the most pressing next steps of the WaCky initiative.

2. Corpus construction

The procedure described in this Section was carried out on a server running Fedora Core 3 with 4 GB RAM, 2 Dual Xeon 4.3 GHz CPUs and 2.5 TB hard disk space. Data about corpus size and other relevant summary statistics for each step of the creation process are reported in Table 1 at the end of the Section.

2.1. Seed selection and crawling

Our aim was to set up a resource comparable to more traditional general language corpora, containing a wide range of text types and topics. These should include both 'pre-web' texts of a varied nature that can also be found in electronic format on the web (spanning from sermons to recipes, from technical manuals to short stories, and ideally including transcripts of spoken language as well), and texts representing web-based genres (Santini and Sharoff, 2007), like personal pages, blogs, or postings in forums. It should be noted that our goal here was for the corpus to be representative of the language of interest, rather than being representative of the language of the web. While the latter is a legitimate object for 'web linguistics' (Kilgarriff and Grefenstette, 2003), its pursuit is not among the priorities set out for the WaCky corpora.

The first step was identifying sets of seed URLs which ensured variety in terms of content and genre. In order to find these, random pairs of randomly selected content words in the target language were submitted to Google. The queries were formed by two-words tuples because preliminary experimentation found that single word queries tend to yield potentially undesirable documents (e.g., dictionary definitions of the queried words), whereas combining more than two words would often retrieve pages with lists of words, rather than connected text. Content- and genre-wise, previous research on the effects of seed selection upon the resulting web corpus (Ueyama, 2006) suggested that automatic queries to Google which include words sampled from traditional written sources such as newspapers and reference corpus materials tend to yield 'public sphere' documents, such as academic and journalistic texts addressing socio-political issues and the like. Issuing queries with words sampled from a basic vocabulary list, on the contrary, tends to produce corpora featuring 'personal interest' pages, like blogs or bulletin boards. Since it is desirable that both kinds of documents are included in the corpus, relevant sources have been chosen accordingly.

Three sets of queries were generated: the first set (1,000 word pairs) was obtained by combining mid-frequency content words randomly selected from the BNC; function words were excluded from the list, since search engines usually ignore them when submitted as part of a query. The second list of queries (500 word pairs) was obtained by randomly combining words sampled from the spoken section of the BNC, while the third list (500 word pairs) was generated from a vocabulary list for foreign learners of English[1] which (however counter-intuitively) contains rather formal vocabulary, possibly required for academic study in English. Once the various seed words had been identified, they were paired randomly before submission to Google.

A maximum of ten seed URLs were retrieved for each random seed pair query, and the retrieved URLs were collapsed in a single list. Duplicates were discarded and, to ensure maximal sparseness, only one (randomly selected) URL for each (normalized) domain name was kept. The remaining URLs were fed to a crawler in random order. The crawl was limited to pages in the `.uk` web domain whose URL does not end in a suffix cuing non-html data (`.wav`, `.jpg`, etc.). The rationale for the choice of limiting the crawl to `.uk` pages is that our goal was to construct a relatively homogeneous resource, comparable to the BNC, and because of practical issues arising when trying to define the country domains to crawl (i.e., including or excluding countries in which English is an official, though not a native language), as well as how to treat U.S. pages (relying on the `.us` domain would provide a very skewed sample of texts). Our strategy does not, of course, ensure that all the pages retrieved represent British English.

The crawl was performed using the Heritrix[2] crawler and was stopped after 10 days. The full seed pair and seed URL lists are available from the project page (see Section 4.).

2.2. Post-crawl cleaning

Using information in the Heritrix logs, we only preserved documents that were of mime type `text/html`, and between 5 and 200KB in size. As observed by Fletcher (2004) and confirmed by informal experimentation, very small documents tend to contain little genuine text (5KB counts as 'very small' because of the html code overhead) and very large documents tend to be lists of various sorts, such as library indices, store catalogs, etc.

We also identified and removed all documents that had perfect duplicates in the collection (i.e., we did not keep even one instance of a set of identical documents). Decision to apply this drastic policy followed inspection of about fifty randomly sampled documents with perfect duplicates: most of them turned out to be of limited or no linguistic interest (e.g., warning messages, copyright statements and the like). While in this way we might also have wasted relevant content, the guiding principle in our web-as-corpus construction approach is that of privileging precision over recall, given the vastness of the data source.[3]

The contents of all the documents that passed this pre-filtering stage underwent further cleaning based on their contents. First, we had to remove code (html and javascript), together with the so-called 'boilerplate', i.e., following Fletcher (2004), all those parts of web documents which tend to be the same across many pages (for instance disclaimers, navigation bars, etc.), and which are poor in human-produced connected text. From the point of view of our target user, boilerplate identification is critical, since too much boilerplate will invalidate statistics collected from the corpus and impair attempts to analyze the text by looking at KWiC concordances. Boilerplate stripping is a challenging task, since, unlike html and javascript, boilerplate is natural language text and it is not cued by special markup. We adapted and re-implemented the heuristic used in the Hyppia project BTE tool,[4] which is based on the observation that the content-rich section of a page has a low html tag density, whereas boilerplate text tends to be accompanied by a wealth of html (because of special formatting, many newlines, etc.). Thus, of all possible spans of text in a document, we pick the one for which the quantity $N(tokens) - N(tags)$ takes the highest value. After they are used for the count, all html tags and javascript code and comments are removed using regular expressions.

While resource-free and efficient, the proposed boilerplate stripping method has several limits. Most importantly, it cannot extract discontinuous fragments of connected text; thus, for pages with boilerplate in the middle, depending on the tag density of this middle part, we end up either with only one of the connected text fragments, or (worse) with both, but also the boilerplate in the middle. The heuristic also has problems with the margins of the extracted section, often including some boilerplate at one end and removing some connected text at the other. Recently, more sophisticated supervised boilerplate stripping methods have been proposed as part of the 2007 CLEANEVAL competition – see algorithms described in Fairon et al. (2007). However, the unsupervised, heuristic method we are using outperforms all the CLEANEVAL participants in the text-only task of the competition, with a score of 85.41 on average (the best competitor achieves a mean score of 84.07).[5]

Next in the pipeline, the cleaned documents were filtered based on a list of 151 function words. Connected text is known to reliably contain a high proportion of function words (Baayen, 2001), therefore documents not meeting certain minimal parameters – ten types and thirty tokens per page, with function words accounting for at least a quarter of all words – were discarded. The filter also works as a simple and effective language identifier.

Lastly, pornographic pages were identified and eliminated, since they tend to contain long machine-generated texts, probably used to 'trap' search engines. Lists were created of words that are highly frequent in ad-hoc crawls of pornography websites. A threshold was then set, such that documents containing at least 3 types or 10 tokens from this

[1]`http://wordlist.sourceforge.net/`

[2]`http://crawler.archive.org/`

[3]This is also the reason for excluding such documents as `.pdf` and `.doc` files from the crawl (cf. Section 2.1.). It is true that these documents may contain interesting linguistic materials, but, on the negative side, they require ad-hoc post-processing techniques, and we are not aware of reliable tools for converting them to text files. We plan to tackle this issue in future work.

[4]`http://www.smi.ucd.ie/hyppia/`

[5]These experiments were conducted by Jan Pomikálek, whose contribution we gratefully acknowledge.

list were discarded.

2.3. Near-duplicate detection and removal

The next step consisted in identifying near-duplicates, i.e., documents with substantial overlapping portions. There are several reasons to postpone this to after corpus cleaning, and in particular after boilerplate stripping. Boilerplate may create both false positives (different documents that share substantial amounts of boilerplate, thus looking like near-duplicates) and false negatives (documents with nearly identical contents that differ in their boilerplate). Also, near-duplicate spotting is computationally costly and hard to parallelize, as it requires comparison of all documents in the collection; thus it is wise to reduce the number and size of documents in the collection first.

We use a simplified version of the 'shingling' algorithm (Broder et al., 1997). For each document, after removing all function words, we took fingerprints of a fixed number `s` of randomly selected n-grams (sequences of `n` words; we counted types, not tokens – i.e., we only looked at distinct n-grams, and we did not take repetitions of the same n-gram into account). Then, for each pair of documents, we counted the number of shared n-grams, which can be taken to provide an unbiased estimate of the overlap between the two documents (Broder et al., 1997). For pairs of documents sharing more than `t` n-grams, one of the two was discarded. The pairs were ordered by document ID, and, to avoid inconsistencies, the second document of each pair was always removed. Thus, if the pairs A-B, B-C and C-D were in the list, only document A was kept; however, if the list contained the pairs A-C and B-C, both A and B were kept. The devising of efficient ways to identify *clusters* of near-duplicates, rather than pairs, is left to future work.

In constructing ukWaC, 25 5-grams were extracted from each document, based on preliminary experimentation. Near-duplicates are defined as documents sharing as few as two of these 5-grams. This threshold might sound surprisingly low, yet there are very low chances that, after boilerplate stripping, two unrelated documents will share two sequences of five content words. A quick sanity check conducted on a sample of twenty pairs of documents sharing two 5-grams confirmed that they all had substantial overlapping text. The near-duplicate detection phase took about four days.

2.4. Annotation

At this point, the surviving text could be enriched with different types of annotation. Part-of-speech tagging and lemmatization was performed by the TreeTagger.[6] The annotation phase took about five days.

In its final, annotated version, ukWaC contains 1.9 billion tokens, for a total of 12 GB of uncompressed data (30 GB with annotation). See Table 1 for detailed size information at the different stages.

[6]`http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`

n of seed word pairs	2,000
n of seed URLs	6,528
raw crawl size	351 GB
size after document filtering	19 GB
n of documents after filtering	5.69 M
size after near-duplicate cleaning	12 GB
n of documents after near-duplicate cleaning	2.69 M
size with annotation	30 GB
n of tokens	1,914,150,197
n of types	3,798,106

Table 1: Size data for ukWaC.

3. Evaluating ukWaC through wordlist comparisons

When corpora are built through automated procedures, as is the case for ukWaC, there is limited control over the contents that make up the final version of the corpus. Post-hoc evaluation is therefore needed to appraise actual corpus composition. Along the lines of Sharoff (2006) (cf. Section 5.), here we provide a qualitative evaluation of our web corpus based on a vocabulary comparison with the widely used BNC. A mostly quantitative evaluation of the overlap of ukWaC and the BNC in terms of lexis is presented in Baroni et al. (2008).

Separate wordlists of nouns, verbs and adjectives were created for the two corpora, which were then compared via the log-likelihood association measure.[7] This makes it possible to identify the words that are most characteristic of one corpus when compared to the other (Rayson and Garside, 2000). Since the procedure relies on the tagger's output, it should be noted that the version of the BNC used was retagged using the same tools as ukWaC, so as to minimize differences in the wordlists that would be due to different annotation procedures.

For each of the 50 words with the highest log-likelihood ratio, 250 randomly selected concordances were retrieved and analyzed. In the following Sections the results of the analysis are presented.

3.1. Nouns

The nouns most typical of ukWaC when compared to the BNC belong to three main semantic areas (see Table 2 for some examples), i.e., (a) computers and the web, (b) education, and (c) what may be called 'public sphere' issues.[8] Category (a) groups words like *website*, *email*, and *software*. If we analyse the contexts in which such words appear, it can be noticed that they are distributed across a wide variety of text types, ranging from online tutorials to promotional texts introducing, e.g. a web-based service. This

[7]Full lists are available from the 'download' section of the WaCky site (see Section 4.). For further details on the wordlist creation and for more detailed analysis see Ferraresi (2007).

[8]Here and below, the analysis does not take into account words typically featured in 'boilerplate' sections of web pages. An example of such words is *information*, which frequently occurs within expressions like "for more" or "for further information".

may be seen as a welcome finding, for at least two distinct reasons. First, no one-to-one correspondance is observed between a topic and a text typology (it could have been possible that, e.g., software instruction manuals emerged as a preponderant text type). Second, a corpus like ukWaC could be used to study the usage of relatively 'new' words, such as those produced within the constantly changing field of new technologies, and that are unattested in traditional corpora. As an example of this, a word like *website* does not appear at all in the BNC.

ukWaC		
Web and computers	Education	Public sphere issues
website	students	services
email	skills	organisations
link	project	nhs
software	research	support
BNC		
Imaginative	Spoken	Politics and economy
eyes	er	government
man	cos	recession
door	sort	plaintiff
house	mhm	party

Table 2: Examples of nouns typical of ukWaC and the BNC grouped according to their semantics.

The analysis of the concordances and associated URLs for nouns belonging to category (b) (e.g., *students*, *research*), and (c) (e.g., *organisations*, *nhs*, *support*) suggests that their (relatively) high frequency can be explained by the considerable presence in ukWaC of certain entities responsible for the publishing of web contents. These are universities – in the case of (b) – and non-governmental organizations or departments of the government – in the case of (c). Typical topics dealt with in these texts are on the one hand education and training and, on the other, public interest issues, such as assistance for citizens in need. The variety of the text genres which are featured is especially remarkable. As pointed out by Thelwall (2005), academic sites may contain very different types of texts, whose communicative intention and register can differ substantially. We find 'traditional' texts, like online prospectuses for students and academic papers, as well as 'new' web-based genres like homepages of research groups. In the same way, the concordances of a word like *nhs* reveal that the acronym is distributed across text types as diverse as newspaper articles regarding quality issues in the services for patients and forum postings on the treatment of diseases.

The nouns most typical of the BNC[9] compared to ukWaC can also be grouped into three macro-categories (examples are provided in Table 2), i.e., (a) nouns related to the description of people or objects, (b) markers of orality (or, more precisely, typical transcriptions of such words), and (c) words related to politics, economy and public institutions. The words included in category (a) are names of body parts, like *eyes*; words used to refer to people, such as *man*, and names of objects and places, like *door*, and *house*. All of these share the common feature of appearing in a clear majority of cases in texts classified by Lee (2001) as 'imaginative' or 'fiction/prose'. As an example, *eyes* appears 74% of the times in 'fiction/prose' texts, and *door* appears in this type of texts almost 62% of the times. In general, what can be inferred from the data is that, compared to ukWaC, the BNC seems to contain a higher proportion of narrative fiction texts, confirming that "texts aimed at recreation [such as fiction] are treated as an important category in traditional corpora" (Sharoff, 2006, p. 85), whereas they are rarer in web corpora. This may be due to the nature of the web itself, since copyright restrictions often prevent published fiction texts from being freely available online.

Category (b) includes expressions which are typically associated with spoken language, including graphical transcriptions of hesitations, backchannels and reduced forms. Among these we find *er*, *cos*, *mhm*, which appear most frequently in the spoken part of the BNC. These words are clearly not nouns. However, since the same tagging method was applied to the two corpora, it is likely that they really *are* more typical of the BNC, inasmuch as their relatively higher frequency cannot be accounted for by differences in tagger behavior. A noun like *sort* is also frequently featured in the spoken section of the BNC, being often found in the expression "sort of". As could be expected, spoken language is less well represented in ukWaC than in the BNC, since the latter was specifically designed to contain 10% transcribed speech.

The last group of words (c) which share important common traits in terms of their distribution across text genres and domains is that of words associated with politics, economy and public institutions. Examples of these nouns are *government*, *recession*, *plaintiff* and *party*. All of these are mainly featured in BNC texts that are classified as belonging to the domain "world affairs", "social sciences" or "commerce", and occur both in academic and non-academic texts. As a category, this seems to overlap with the group of words related to public sphere issues which are typical of ukWaC. However, the specific vocabulary differs because the texts dealing with politics and economy in ukWaC seem to share a broad operative function, e.g. offering guidance or promoting a certain governmental program, as in the following examples:

```
    OGC offers advice, guidance and support;

    Local business support services include the
recently established Sussex Business Link;

    ...use Choice Advisers to provide practical
support targeted at those parents most likely
to need extra-help.
```

Concordances reveal instead that in the BNC words like *government* or *recession* are more frequently featured in texts which *comment on* a given political or economic situation, as e.g., newspaper editorials would do, for example:

```
    ...is urging the government to release all
remaining prisoners of conscience;

    Despite assurances from government
officials that an investigation is underway;

    ...a crucial challenge to the cornerstone
of his government's economic policy.
```

[9]As can be noticed in Table 2, some of the words taken into account here are not nouns (e.g. *er*), but rather expressions which were erroneously recognized by the tagger as nouns.

3.2. Verbs

In Section 3.1. the nouns most characteristic of ukWaC and the BNC were grouped and analysed on the basis of the text domains and types they typically appear in. Identifying a similar relationship between textual domains and verb forms is somewhat less immediate, since verbs are often less easily associated with, e.g., a particular text topic. Alternatively, one can adopt a semantic classification based on their core meaning – here we follow that proposed by Biber et al. (1999) –, and assess whether verbs belonging to the same class show similar distributional patterns across, e.g., textual types. Another important aspect which is taken into account in the analysis of verbs is that of verb tenses, which, as we shall see, can provide further indications about the texts that characterize the two corpora.

ukWaC	
Activity verbs	Verbs of facilitation
use develop provided visit	help support improve ensure
BNC	
Activity verbs	Mental verbs
looked nodded go shrugged	know mean thought saw

Table 3: Examples of verbs typical of ukWaC and the BNC by semantic category.

A clear majority of the 50 verb forms most typical of ukWaC when compared to the BNC can be classified either as “activity verbs”, i.e. verbs which “denote actions and events that could be associated with choice” (*ibid.*, p. 361), such as *use*, *provide* and *work*, or as “verbs of facilitation or causation”, i.e. verbs that “indicate that some person or inanimate entity brings about a new state of affairs”, such as *help* and *allow* (other examples can be found in Table 3). Taken together, the verb forms belonging to these two categories account for almost 50% of the verbs most characteristic of ukWaC. Their distribution across text types, however, seems to differ.

Activity verbs are evenly distributed across the main text types identified in Section 3.1., i.e. promotional texts – issued both by private companies and governmental departments and universities –, “discussion texts”,[10] such as news articles and postings in forums, and “instruction texts”, like help pages and instruction manuals. Here are some examples:

```
    Specifically created to perform research
and to **develop** future leaders for aerospace
manufacturing;

    ...children are dying of AIDS. It
challenges all religions to **work** together
to reduce the stigma;
```

[10]The terminology used to classify web texts is taken from Sharoff (2006).

```
    When you **visit** a web page, a copy of that
page is placed in the cache.
```

As could be expected, verbs of causation, on the contrary, show a distinct tendency to appear in only two of these text types, i.e. instruction and promotional texts. In promotional texts, in particular, verbs of causation are used to convince readers that a certain product, service or idea can actually make a difference, as in the following sentence:

```
    Acas aims to **improve** organisations and
working life through better employment
relations.
```

It is interesting to notice in this respect that many texts in ukWaC are not easily classifiable as belonging to one single category. This is the case, e.g., for seemingly instructional texts, which actually also promote the product they are describing. Thus, a sentence like:

```
    Once again, we can **help** with any queries
you may have.  Products liability insurance
will cover...
```

published on the help page of an insurance company can hardly be seen as having a merely informative function. This corresponds to what Santini (2007) calls “genre hybridism”, which often makes it especially difficult to classify web text into clear-cut genre categories.

If verb tenses are taken into account, it can be noticed that most verbs in the list are in the present tense (or in their base form), and that those which could appear as past forms are, in fact, often used as past participles in passive forms. This could be due to the already noted considerable importance in ukWaC of discussion texts, which are typically concerned with current affairs, or of promotional and instruction texts, which often make use of the imperative form.

Verb forms in the BNC belong to two main semantic categories, i.e. activity verbs, like *looked* and *go*, and “mental verbs”, i.e. verbs that “denote a wide range of activities and states experienced by humans, [...] do not involve physical action and do not necessarily entail volition” (Biber et al., 1999, p. 362), like “know” and “thought” (see Table 3 for other examples).

The verbs belonging to these two categories show very similar distributional patterns: verbs in the past form occur most frequently in imaginative/fiction texts, whereas present tense forms are most frequently featured in the spoken section of the corpus. As regards this point, notice that activity verbs in the BNC – which usually indicate a physical action, e.g. of a character in fiction (cf. *nodded*, *shrugged*) – seem to be less evenly distributed across text types than activity verbs in ukWaC. As an example, the past tense form *looked* appears 67% of the times in fiction texts, and *nodded* 94% of the times. As already mentioned, present tense forms – which, however, are a minority in the list, accounting for less than 15% of the total number of verbs analysed – are instead most frequent in spoken language. The verb form *go*, e.g., appears 36% of the times in spoken texts (26% of the times in fiction texts), and the mental verb *know* occurs in such texts almost 50% of the times.

Summing up, the (relatively) high frequency of activity and mental verbs in the BNC can be explained by their being frequently used within two text types, i.e. fiction and spoken texts. Moreover, when verb tenses are also taken into account, the BNC, unlike ukWaC, seems to be characterized by past-oriented (narrative) language.

3.3. Adjectives

The adjectives most typical of ukWaC can be classified as belonging to four semantic areas, i.e. (a) web-related adjectives, (b) public sphere-related adjectives, (c) time-related adjectives, and (d) emphatic adjectives conveying a positive evaluation (see Table 4 for examples). As can be noticed classes (a) and (b) correspond to two of the main text topics identified in Section 3.1., thus confirming that such topics are well represented within ukWaC.

ukWaC			
Web	Public sphere	Present time	Emphatic
online digital mobile	sustainable global disabled	new current innovative	excellent fantastic unique
BNC			
Imaginative	Politics	Present time	Sciences
pale dark afraid	political soviet conservative	last former nineteenth	gastric colonic ulcerative

Table 4: Examples of adjectives typical of ukWaC and the BNC grouped according to their semantics.

Both adjectives belonging to class (a) and (b) show distributional patterns similar to those of their noun "counterparts". Adjectives like *online* and *digital* can be found in technical instruction texts, such as tutorials and user manuals; in discussion pages, like blogs, and in promotional texts about computing-related services. Similarly, adjectives like *sustainable* and *global* typically occur in texts created by departments within the government and NGOs, or in various kinds of promotional or discussion texts, such as texts promoting a political (or humanitarian) program, or news. Topics in ukWaC thus seem to correspond to a certain extent to current themes of discussion (such as "global economy" and "sustainable growth"). This, however, is also true for the BNC, in which two of the most typical adjectives compared to ukWaC are *soviet* and *cold*. Such datum is likely to reflect the importance that such themes as the "Soviet Union" and the "Cold War" – which are among the most frequent bigrams including these adjectives – had at the time of the corpus construction.
Category (c) includes adjectives referring to present time, or signalling a change with respect to the past, like, e.g., *new* and *current*. The presence of such adjectives may be seen as also connected with the high frequency of verbs in the present tense. Taken together, these two features seem to point at the fact that the web texts in ukWaC are typically both focused on the present time and willing to signal it explicitly. This is notably true for press releases and promotional pages. In the latter type of texts, adjectives which signal a radical change with respect to the past (e.g. *innovative*) are particularly used to display how original and innovative a service or product is.
The presence of a considerable number of promotional texts is also revealed by the high frequency of adjectives which are chiefly used to indicate positive characteristics (category (d)), like *excellent*, *fantastic*, and *unique*. All of these are mainly found, e.g., in descriptions of products, services or tourist attractions, as in the following example:

```
    ...your stay in Cornwall. Fantastic views
  across the ocean and countryside.
```

The adjectives most typical of the BNC when compared to ukWaC can also be classified into four main semantic areas, i.e. (a) adjectives used to describe people and objects, (b) politics-related adjectives, (c) adjectives related to past time, and (d) science-related adjectives (see examples in Table 4).
In the case of the BNC too, classes (a) and (b) correspond to two of the categories identified in Section 3.1. In category (a) we find adjectives that refer to physical characteristics of people (e.g. *pale*, *tall*), or of inanimate objects and settings in which an action takes place (e.g. *dark*, *thick*), and others that relate to people's temper (e.g. *anxious*, *angry*). As could be expected, all of these are most frequently found in imaginative texts. Adjectives belonging to category (b) include "general", hypernymic adjectives (e.g. *political*, *social*), and adjectives which designate national provenance (*soviet*, *french*) or refer to political parties (*conservative*). These are typically found in three domains, i.e. "world affairs", "social sciences" and "commerce" (Lee, 2001). As was noted in Section 3.1., this category of words seems to overlap with that of public-sphere issues identified in ukWaC. Concordances of politics-related adjectives, however, confirm that texts in which the two categories of adjectives occur differ: public sphere-related texts in ukWaC are often concerned with matter-of-fact issues (like, e.g., offering *support* to disabled people), and are mainly focused on the present (cf. Section 3.2.). Texts related to politics in the BNC, on the contrary, seem to describe events through general, abstract categories (e.g. *political*), and to report facts in the past time (cf. Section 3.2. and Section 3.1. for some examples).
In this regard, it is interesting to notice that, unlike in ukWaC, the adjectives most typical of the BNC relating to time refer to the past (category (c)), like, e.g., *last*, *former*, and *nineteenth* (whose most frequent collocate is *century*). These are mainly found in two text domains, i.e. world affairs and social sciences. Their frequency in these text types may be seen as confirming that texts about politics and economics in the BNC seem to adopt a retrospective, historical approach to facts, as is typical, e.g., of academic and journal articles.
Finally, adjectives belonging to category (d) are related to natural and applied sciences. Words like *gastric*, *colonic*, and *ulcerative* are often found in academic and non-academic essays which deal with anatomy or health problems (medicine). A closer look at the adjectives reveals that several refer to the digestive system. It seems therefore likely that the BNC contains a higher proportion of

essays on the specific topic of human or animal digestion than ukWaC (cf. also Kilgarriff and Grefenstette (2003)). In turn, this could be interpreted as a sign of the relative weight that even a few texts can have on a (not so small) corpus like the BNC.

3.4. Discussion

In the present Section a method was presented to provide an evaluation of ukWaC's contents. The method involved constructing different lists of nouns, verbs, and adjectives. The same procedure was carried out on the BNC, and the lists were subsequently compared across the two corpora via the log-likelihood association measure. This made it possible to find the words that are comparatively more frequent in either ukWaC or the BNC, i.e. the words that may be seen as being relatively typical of one corpus when compared to the other.

When two corpora are evaluated through word list comparisons, however, two points need to be remembered. The first is that all the words that appear in the lists should be taken as being indicators of relative typicality in one corpus or the other, and not as being absolutely typical of them. To give an example, the noun *eyes* appears as the $4th$ most typical noun of the BNC, even though its absolute frequency is nearly 15 times lower than in ukWaC. Thus, the fact that a word is typical of the BNC does not imply that it is not equally well represented in ukWaC. The second point is that the method is apt at highlighting strong asymmetries in the two corpora, but it conceals those features that make them similar (represented by words that have a log-likelihood value close to 0). In future work, we intend to determine what kinds of text types or domains do *not* turn up as typical of either ukWaC or the BNC, and assess whether there is ground to conclude that they are similarly represented in both corpora.

Moving on to the actual data analysis, it would seem that, compared to the BNC, ukWaC contains a higher proportion of texts dealing with three domains, i.e. the Web, education, and what were called "public sphere issues". These appear in a wide range of text types. Web-related issues, in particular, are found in almost all the text types identified by Sharoff (2006), i.e. discussion (e.g. online forums of discussion about a particular software or website), promotional (e.g. advertising of a traditional or web-based service) and instruction texts (e.g. tutorials). The presence of those among the most typical of ukWaC is unsurprising, insofar as they represent meta-references to the medium of communication that hosts them, and as the BNC was published at a time when the web was still in its infancy. Education and public service issues are also found in a great variety of text types, ranging from "traditional" texts like academic articles, to more recent web-based genres, like presentation pages detailing the activity, e.g., of a research or humanitarian group. Such heterogeneity of text types is a very positive feature in terms of the internal variety of ukWaC, since no one-to-one correspondence between a certain topic and a text type can be identified. This can be interpreted as confirming the soundness of the sampling strategy adopted.

In terms of domains, the BNC features a comparatively larger presence of narrative fiction texts. These are characterised by the frequent use of nouns and adjectives referring to physical characteristics or emotions, and by verbs (in the past tense) related to human actions. Moreover, the BNC seems to contain a higher proportion of spoken texts, whose presence is signalled by a number of discourse markers (e.g. *er*) and mental verbs in the present tense (e.g. *know*, *mean*). The third category of texts typical of the BNC is that of texts which deal with political and economic issues. Such texts differ from public service texts found in ukWaC, which are characterised by a stronger focus on practical issues (e.g. offering guidance to citizens), and on the present time. Politics- and economy-related texts in the BNC, on the contrary, are more concerned with describing events through abstract categories and using the past tense, as is typical, e.g., of non-fiction prose.

Differences in temporal deixis across the two corpora prove especially noteworthy. ukWaC seems to be characterised by a stronger concern with the present time, as is demonstrated, e.g., by the use of verbs in the present tense and of adjectives which refer to the present. This may be due, among other factors, to a considerable presence of advertising texts, which also display a number of causative verbs and of adjectives conveying a positive evaluation. One of the most interesting findings in this regard was that such advertising texts are featured not only in pages selling commercial products or services, but also in pages published by universities (e.g. inviting students to enrol), and governmental departments (e.g. promoting a political program). In the BNC, on the contrary, narrative language, characterised by past tense verbs and adjectives referring to the past, is more prominent.

4. Availability

UkWaC is available for download from the website of the Wacky initiative,[11] which also contains other data, such as frequency lists, seeds (words, tuples and URLs) as well as the lists used for the comparisons in Section 3. The customized tools used for corpus construction (duplicate detection, boilerplate stripping, etc.) are also available for download from the website. The corpus is available in two formats, as a plain text file (with no morphological annotation) and as a POS-tagged file encoded in a shallow XML format. This format is ready for indexing with the IMS Open Corpus Workbench (CWB),[12] a popular corpus processing tool. UkWaC is also available via the commercial "Sketch Engine".[13]

5. Related work

There is by now a large and growing literature on using the web for linguistic purposes, mostly via search engine queries or by crawling ad-hoc data – see for example the papers in Kilgarriff and Grefenstette (2003), Baroni and Bernardini (2006), Hundt et al. (2007), Fairon et al. (2007). On the other hand, we are not aware of much publicly documented work on developing large-scale, general-purpose web-derived corpora.

[11] `http://wacky.sslmit.unibo.it`

[12] `http://cwb.sourceforge.net`

[13] `http://www.sketchengine.co.uk`

The work most closely related to ours is that presented in Sharoff (2006). The author developed a collection of 'BNC-sized' corpora (around 100 M tokens) that, as of early 2008, include English, Chinese, Finnish, French, German, Italian, Japanese, Polish, Portuguese, Russian and Spanish, and that can be queried via an online interface.[14] The methodology followed (Sharoff, 2006) is similar to the one described here – indeed, many tools and ideas were developed jointly. The main differences are that Sharoff does not perform a true crawl (he retrieves and processes only the pages returned by random Google queries, rather than using them as seed URLs), nor does he perform near-duplicate detection. Evaluation of some of these corpora is carried out in Sharoff (2006), where a comparison is made with reference corpora in the same languages, in terms of domain analysis and comparing wordlists, similarly to what we did here. For a more systematic literature review, however, we invite the reader to refer to Baroni et al. (2008).

6. Further work

UkWaC is already being actively used in several projects, including simulations of human learning, lexical semantics and langage teaching. We hope that this article will encourage other researchers to adopt ukWaC as a research tool, and that these activities will give us a clearer idea of the corpus' strengths and limits.

We believe that the most pressing issue at this moment is the need to provide free access to the corpus, both through a web service that allows scripting access to remote corpora (to support linguists in doing extensive qualitative and quantitative research with the corpora) and via a web user interface that should allow user-friendly access to those without advanced technical skills (e.g., language learners, teachers and professionals). We are actively working in these areas.

A second important line of research pertains to automated cleaning of the corpora, and to the adaptation of tools such as POS taggers and lemmatizers – that are often based on resources derived from newspaper text and other traditional sources – to web data. Moreover, corpora should be enriched with further layers of linguistic annotation. To this effect, we recently finished parsing ukWaC with a dependency parser and we are currently investigating the best way to make these data available.

7. Acknowledgements

We would like to thank the members of the WaCky community for many useful interactions, in particular: S. Castagnoli, T. Emerson, S. Evert, W. Fletcher, F. Gaspari, A. Kilgarriff, J. Pomikálek and S. Sharoff. The WaCky project is partly funded by the University of Bologna through the LiMiNe funding scheme.

8. References

A. Baayen. 2001. *Word frequency distributions*. Kluwer, Dordrecht.

M. Baroni and S. Bernardini, editors. 2006. *Wacky! Working papers on the Web as Corpus*, Bologna. Gedit.

M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. 2008. The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora. submitted.

D. Biber, S. Johansson, G. Leech, S. Conrad, and E. Finegan. 1999. *Longman grammar of spoken and written English*. Harlow, London.

A. Broder, S. Glassman, M. Manasse, and G. Zweig. 1997. Syntactic clustering of the web. In *Proceedings of the Sixth International World Wide Web Conference*, pages 391–404, Santa Clara, California.

C. Fairon, H. Naets, A. Kilgarriff, and G.-M. de Schryver, editors. 2007. *Building and exploring web corpora – Proceedings of the 3rd Web as Corpus Workshop, incorporating Cleaneval*. Presses Universitaires de Louvain, Louvain.

A. Ferraresi. 2007. Building a very large corpus of English obtained by web crawling: ukWaC. Master's thesis, University of Bologna. Retrieved January 28, 2008 from `http://wacky.sslmit.unibo.it`.

W. Fletcher. 2004. Making the web more useful as a source for linguistic corpora. In U. Connor and T. Upton, editors, *Corpus Linguistics in North America 2002*, Amsterdam. Rodopi.

M. Hundt, N. Nesselhauf, and C. Biewer, editors. 2007. *Corpus linguistics and the web*. Rodopi, Amsterdam.

A. Kilgarriff and G. Grefenstette. 2003. Introduction to the special issue on the web as corpus. *Computational Linguistics*, 29(3):333–347.

D. Lee. 2001. Genres, registers, text types, domains, and styles: Clarifying the concepts and navigating a path through the BNC jungle. *Language Learning & Technology*, 5(3):37–72.

P. Rayson and R. Garside. 2000. Comparing corpora using frequency profiling. In *Proceedings of Workshop on Comparing Corpora of ACL 2000*, pages 1–6, Hong Kong, China.

M. Santini and S. Sharoff, editors. 2007. *Proceedings of the CL 2007 Colloquium: Towards a Reference Corpus of Web Genres*, Birmingham, UK.

M. Santini. 2007. Characterizing genres of web pages: genre hybridism and individualization. In *Proceedings of the 40th Hawaii International Conference on System Sciences, poster session*, pages 1–10, Waikoloa, Hawaii.

S. Sharoff. 2006. Creating general-purpose corpora using automated search engine queries. In M. Baroni and S. Bernardini, editors, *Wacky! Working papers on the Web as Corpus*, pages 63–98, Bologna. Gedit.

M. Thelwall. 2005. Creating and using web corpora. *International Journal of Corpus Linguistics*, 10(4):517–541.

M. Ueyama. 2006. Evaluation of japanese web-based reference corpora: Effects of seed selection and time interval. In M. Baroni and S. Bernardini, editors, *Wacky! Working papers on the Web as Corpus*, pages 99–126, Bologna. Gedit.

[14] `http://corpus.leeds.ac.uk/internet.html`

RoDEO: Reasoning over Dependencies Extracted Online

Reda Siblini and Leila Kosseim

CLaC Laboratory
Department of Computer Science and Software Engineering
Concordia University
1400 de Maisonneuve Blvd. West
Montreal, Quebec, Canada H3G 1M8

R_Sibl@encs.concordia.ca, Kosseim@encs.concordia.ca

Abstract

The web is the largest available corpus, which could be enormously valuable to many natural language processing applications. However it is becoming very difficult to identify relevant information from the web. We present a system for querying dependency tree collocations from the web. We show its usefulness in identifying relevant information by evaluating its accuracy in the task of extracting classes of named entities. The task achieved a general accuracy of 70%.

1. Introduction

(McEnery and Wilson, 2001) has described corpus linguistics as the study of language based on examples of 'real life' language use. And where can we find more examples of 'real life' language use than the web? With the growth of the World Wide Web, more and more researchers have been used it in their study of language. Collocations are a rich source of information that could be very useful in many natural language processing tasks. The richness of collocations is attributed to its relationship with meaning. This relationship has been noticed by many researchers, and probably two of the most quoted ones are (Firth, 1957) "you shall know the word by the company it keeps" and (Harris, 1968) distributional hypothesis that "words with similar meaning tend to appear together". The term collocation by itself has more than one definitions in the literature: (Firth, 1957) described it as "habitual" word combinations, (Manning and Sch'eutze, 1999) as some conventional way of saying things, (Bartsch, 2004) as a frequently recurrent, relatively fixed syntagmatic, combinations of two or more words, and (Evert, 2005) as word combination whose semantic and/or syntactic properties cannot be fully predicted from those of its components, and which therefore has to be listed in a lexicon.
Although the importance of collocation is obvious, it is not always easy to collect collocation information for words that are usually unavailable in a typical corpus; for example collocations of a specific proper noun. As the web contains the largest repository of text, it is seen as a solution of such a problem. However, using the web as a corpus poses its own set of challenges. In this paper we describe a system for extracting, representing, and querying collocations from the web, and we attempt to respond to some of those challenges posed by the web.
In the remainder of this paper, we first review related work, then describe a system for extracting collocations online titled RoDEO: for Reasoning over Dependencies Extracted Online. And then we evaluate it on the task of extracting classes of named entities.

2. Related Work

In this section we will present some of the related work in the literature that extract, represent, and query collocation information from the web.
(Kilgarriff et al., 2004)'s sketch engine, is a corpus tool that creates "Word sketches", or one-page summaries of a word's grammatical and collocational behavior, from a corpus, in addition to other functionalities. The corpus query language and search is based on regular expression. The tool also makes uses of a web service that produces corpora from the web, but the time needed to build a corpus from the web could varies between minutes and hours.
(Resnik and Elkiss, 2005) described the Linguist's Search Engine (LSE), a system that enables language researchers to query the web for examples based on syntactic and lexical criteria. The system allows users to create queries by example: by supplying a sample sentence the system parse the sentence, and find similar structure in the user selected corpus. To use the web as a corpus the user must supply a web query to a search engine from within the system, the system will extract, parse and index sentences from the web search results and the result can be used by the user as a new corpus. The user has to wait between minutes and hours for this process to be completed, however the user can begin querying the results as soon as they are collected. The indexing and search of the collected data is based on a method for querying XML Data by tree structures. It should be noted that parse trees data are not similar to the typical semi-structured data that is usually stored and queried in XML, as phrases structure are usually highly recursive.
(Renouf et al., 2007) presented WebCorp as a tool that helps corpus linguists in retrieving linguistic output from the web. The request for linguistic information is translated and feed to web search engine, the returned documents will be processed, and the concordance results are returned to the user. The author presented some of the current WebCorp problems, such as the performance issues, the need of a grammatical and better collocational analysis, in addition to a more sophisticated pattern matching.

Most of these systems are similar to the system we are presenting in this paper; however the differences and the advantage of our system will be evident in the following sections.

3. Extracting Collocations Online

Annotated corpora contain linguistic information that enables linguists to accurately search for the specific occurrences of linguistic information. The usefulness of the corpus is increased with the level of annotation or linguistic information explicitly present in it, and with the expressiveness of the search query formalism utilized to investigate these data.

Most of the related work presented in the previous section annotates the utilized corpus, or its extracted part, with syntactic structure. The syntactic structure used follows a certain syntactic theory, nevertheless most researches, justifiably, try not to select a theory specific structure. Two main categories of syntactic structure are usually described: phrase structure and dependency structure. Phrase structure combines words or phrases into syntactic categories or constituent parts. Dependency structure, on the other hand, is a representation of relationships between words. (Hays, 1964) define a dependency relationship as a binary relationship between a word called head and another called modifier. For example the sentence: *"Tom drives a car"* can be represented by the following set of dependency relationships: (Tom—subject—drives), (a—determiner—car), and (car—object—drives). Figure 1 shows a graphical

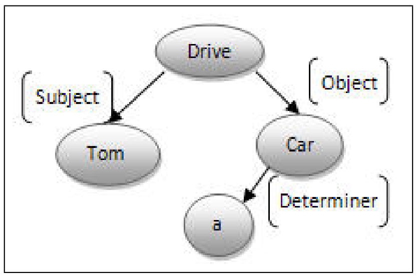


Figure 1: An Example Dependency Tree

representation of this dependency tree. The expressiveness of the dependency formalism is usually related to the different types of syntactic relations used to characterize the words relationships.

We prefer the use of dependency structure over phrasal structure since they represent syntactic relations explicitly; however these relationships are more implicit in a phrasal structure. For example, the actuality that *"Tom"* is the subject of *"drives"* in the above example can only be uttered in the phrasal structure as a sentence having noun phrase headed by the noun *"Tom"*, and related to a verb phrases headed by the verb *"drives"*. As such using a dependency structure allows us to directly represent and query syntactic relations. However selecting the syntactic structure is not the main issue in querying or representing syntactic collocations. The main issue is related to how to represent the created structure and how to query it. In the next section we are going to present the preferred annotation representation structure that will allows us, in addition to the incorporation of syntactic structure, the inclusion of semantic information. The selected representation is a scalable and tractable structure that will enable expressive query formalism in addition to an advanced reasoning capabilities based on the availability of syntactic and semantic information.

3.1. Corpora Representation

Before describing the preferred corpora representation, we will revisit the collocation definition in order to specify our preference, which correspond to our syntactic structure preference that we have described in the previous section.

(Lin, 1998b) defines a collocation as a dependency relationship between two words that occurs significantly more frequently than by chance. He also proposed a method for extracting dependency collocations from text corpora. His method involves the parsing of a corpus using the Minipar (Lin, 1998a) parser, and storing the resulted dependencies into a database. By using dependency frequencies and mutual information he separated collocations from dependency triples that occurred by coincidence. We would like to extend (Lin, 1998a)'s collocation definition to a dependency relationship between a word and one or more dependency trees. We refer to this type of collocations as Dependency Tree Collocation or (DTC). This extension of collocation would allow us to filter the extracted dependencies into very specific ones. This restriction is required as the most frequent collocations vary by context, and in a given context we are not usually talking about all the possibilities of a word collocating with another, but to a restricted subset that the context is related to. As such, this extension will allow us the restriction of the returned collocations to a selected context.

A dependency collocation between two words could be represented in the following linear form: [Word1] Dependency1—[Word2]; where Word1 collocates with Word2 via the dependency relationship Dependency1. On the other hand, a dependency collocation between a word and a dependency tree could be represented in the following form [Word1] Dependency1—[Word2] Dependency2—[Word3] Dependency3—[Word4]... or graphically as in Figure 2.

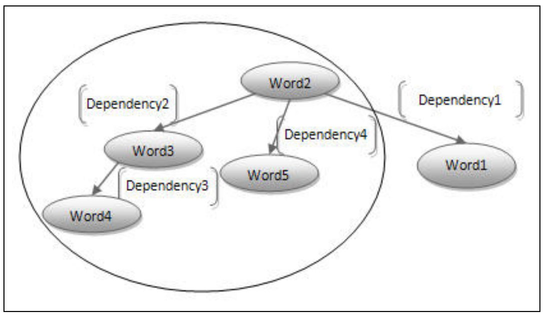


Figure 2: A Graphical Representation of a Dependency Tree Collocation

One of (Lin, 1998b)'s example of a collocation as a dependency relationship between two words is [Word]Object—[Drink], or finding the word that collocates with the word *"Drink"* through an object dependency relationship. We can extend this example to find [Word]Object—(Drink)—Subject(Child) or the word that collocates through an object dependency relationship with the dependency tree consisting of the verb *"Drink"* having a subject dependency relationship with the word *"Child"*. Although the top ranked collocation of (Lin, 1998b)'s example is *"Tap Water"* the top words for our DTC is *"Milk"*. In this sense we have restricted the retrieved collocations to the instances of the verb that collocate with a specific object. We could have different restriction types depending on the formulated DTC query.

Multiple method have been proposed in the literature to represent and query corpora annotated information, from a simple regular expression match such as the sketch engine of (Kilgarriff et al., 2004), to a more complex semi-structured way as in the linguistic Search Engine case of (Resnik and Elkiss, 2005). Nevertheless, what we are presenting here is our preferred annotation representation structure that will allows us, not only to represent syntactic structure, but also to include semantic information in a scalable, tractable structure. This structure will enable expressive query formalism, in addition to an advanced reasoning capabilities based on the availability of syntactic and semantic information. Our representation is based on a decidable logic representation of the DTC structure. Since dependency trees could be seen as a semantic network, using description logic that is equipped with a formal logic-based semantics fits exactly our needs. Especially when such a logic is a tractable, decidable subset of predicate logic, and has several well studied theorem provers to query it.

Description logic (DL) is a logical formalism for defining concepts and their relations (Terminologies) specifying properties of individuals (Assertions). Lately, description logics became the foundation of the semantic web. The Semantic web is a collaborative effort led by World Wide Web consortium (W3C) that provides a framework for making World Wide Web content processable by machines (Berners-Lee, 1998). W3C endorsed the web ontology language (OWL) as the language for the semantic web (Dean et al., 2004). OWL is a semantic markup language for defining and instantiating web ontologies, and it is based on description logic. It is a vocabulary extension of RDF (the Resource Description Framework), derived from the DAML+OIL (DARPA Agent Markup Language and Ontology Interchange Language), and based on XML (Extensible Markup Language). An OWL ontology may include descriptions of classes (or concepts), properties (relations between a main class called domain and another called range), and the classes instances (or individuals).

We have selected OWL-DL as the structure for representing corpora annotation, it is the subset of OWL supporting a decidable (SHOIN(D)) description logic (Horrocks and Patel-Schneider, 2004). The intuition behind this selection is the effortless mapping between a dependency relation and an OWL-DL property between two classes (the head of dependency relation and its modifier). In addition the lemma and part of speech information of each word are mapped to classes. For example, the DTC: (Drinks_Verb)—Subject(Child_Noun) will be mapped to the following OWL descriptions:

- "Subject" as an OWL property having as domain the class "Drink" and as range the class "Child" (note that the lemma of the word is used to describe the class) .
- "Drink" is subclass of the class "Verb" and "Child" is a subclass of the class "Noun".
- In addition to indexed instance of the class "Drink", such as: "Drinks_1", and an indexed instance of the class "Child" such as "Child_1".

The indexing is needed to relate sentence instances. The end result could be illustrated graphically as in figure 3.In this graph rectangles represent classes, ovals represent properties, and double brackets represent instances.

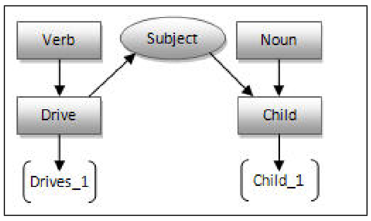


Figure 3: A Graphical Representation of a Dependency Tree Collocation ontology

This representation has a lot of advantages, some of these advantages:

1. Availability of well studied, powerful query formalism.
2. Possibility of applying rules to the created knowledge base using backward or forward chaining.
3. Ability to integrate multiple knowledge bases, some of which might include general semantic information.
4. The existence of well studied reasoners that can be used to answer queries over the created knowledge base schema and instances.
5. OWL is a standard meant to be used for the semantic web, in which semantics of information and services on the web are defined to be used and exchanged. As such, most of the available reasoner services are built for large scalability.

Investigating in details all of these advantages are beyond the scope of this paper. Nevertheless we are going to focus on the first advantage: a powerful query formalism and related answering mechanism, which will be introduced in the next section.

3.2. Query Formalism

The reasoner that we have selected for the query answering is the RACER reasoner (Haarslev and Moller, 2003). RACER (an acronym for Renamed ABox and Concept Expression Reasoner) is a reasoner that implements tableau calculus for description logic (DL) and supports the web ontology languages DAML+OIL, RDF, and OWL. nRQL (new RACER Query language) is an expressive DL-query language. An nRQL query consists of a query head and a query body. For example, the query (retrieve (?x) (?x Noun)) has the head (?x) and the body (?x Noun). It returns all *"Nouns"* from the ontology which is queried. A detailed description of nRQL is given in (Haarslev et al., 2004). Although nRQL is closely related to Horn logic query languages such as Datalog, it has been argued (Wessel and Moller, 2006) that nRQL is a more general query language framework that provides more flexibility and options for extensions than Datalog. In addition, it provides an optimized implementation by bounding the variables in a query to explicitly mentioned instances in the knowledge base.

The DTC query could be represented in nRQL, which enables us to formulate complex conjunctive queries in order to retrieve a specific DTC from the created dependency ontology structure. Then we can use the RACER conjunctive query answering to prove the nRQL, over the created DT ontology, and return the corresponding results.

For example, if we are looking for the dependency tree collocation [Noun]Subject—[Drive]—Object[Ball], that is the nouns that are subjects of the verb *"Drive"* and having *"Ball"* as its object, we can formulate it as the following nRQL query: (Retrieve (?x)(AND(?y ?x Subject)(?y ?z Object)(?y Drive)(?x Noun)(?z Ball)).

Although we are using nRQL as the query formalism of choice, other query languages could be easily used such as the OWL-QL (Fikes et al., 2004) query language of the semantic web.

Using RACER we can then run an nRQL query and return related Dependency Tree Collocations from the web corpora that would be presented as an OWL-DL ontology. Representing the whole World Wide Web as a well structured description-logic knowledge representation would be the ideal solution to query DTCs, however as it is not feasible for us to do so, we are going provide an alternative: a method to generate sentences from a DTC query, which will enable us to created related web queries and retrieve very specific and related documents that will represent a corpus related to the DTC in question. This method will be described in details in the next section.

3.3. From DTC to Sentences

Most web search engines provide an unstructured query language to query the web. Transforming a DTC conjunctive query to a web search query puts forward its own set of challenges. Using the DTC conjunctive query content words as web query keywords may return millions of documents. However, not all the returned documents are related to the dependencies collocation that we are looking for, but barely documents containing the supplied keywords. To narrow down the returned results to the dependency relationship that we are looking for, we formulate the web query as a specific search phrase. A search phrase is a sequence of words that must co-occur together.

For the DTC example query above: (Retrieve (?x)(AND(?y ?x Subject)(?y ?z Object)(?y Drive)(?x Noun)(?z Ball)). The content words of this DTC query are: "Drive" and "Ball". When we used the Google API to search for documents having the keywords "Drive" and "Ball", we obtained about 39 million documents. When we searched for the search phrase *"Drive.Ball"*, we found 167,000 documents. Still this search phrase is not the search phrase that returns the narrowed down documents to the DTC above. In addition, concatenating the content words from the DTC does not always match what we are looking for. So we need a way to know how we usually write sentences that match the dependency tree we are looking for.

Our solution to this problem is the creation of a dependency tree corpus from a subset of the open American National Corpus (ANC), which would act as a representation of the grammatical structure of the sentences in terms of dependency trees. The corpus is represented as an OWL-DL knowledge base in the same way described in section 3.1. We then use the reasoner over the create knowledge base to query for the subclasses of the DTC conjunctive query. For the example above the subclasses of DTC query is (Retrieve (?x)(AND(?y ?x Subject)(?y ?z Object)(?y Verb)(?x Noun)(?z Noun)). Running this query over the created tree ontology will help in the creation of search phrases that correspond to the DTC conjunctive query. The dependency tree corpus only contains dependency trees of grammatical categories and their relationships, but does not contain the actual words of sentences. This knowledge base is meant to be a representation of various grammatical phenomena, where each dependency tree represents a typical sentence in a text, and is ranked by its length and by the number of non-content words it contains. The reasoning behind the ranking is that the more non-content words in a search phrase the more specific the documents that will be returned from the search engine.

For example, one of the search phrases that matched the DTC query above in the created dependency tree knowledge base, and ranked high as having a length of 6 words, where 3 of them are non-content words, is [Determiner.Noun-Subject.Pronoun.Verb.Determiner.Noun-Object].

This search phrase contains a set of related grammatical categories, such as determiner, verb, noun.., which act as placeholders. Non-Content categories are then replaced by a non-content word, for example *"determiner"* could be replaced by *"a"*, and *"pronoun"* with *"that"*. Content words (Noun, Verb) are replaced by the content words of the DTC query; so *"Verb"* would be replaced by *"Drive"* in this example. The rest of the categories will be replaced by search query wildcards. So one of the resulted web search phrase from this example that is specific to the Google API is: [a.*.that.drive.*.ball]. This query with Google returned only 69 documents.

For each DTC conjunctive query we could automatically generate hundreds of similar search phrase queries. Then using a search engine we run the created search phrase queries. The resulted top 10 documents of each search query are downloaded, stripped from HTML, and a regular expression match is performed in order to extract the complete sentences that conform to the search phrase query. For the example query above some of top returned

sentences are:
"A golfer that drives a golf ball that..." —Rank = 8
"He has a swing that drives the ball..." —Rank = 6
"A batter that drives a ball forward..." —Rank = 4
"a force that drives your ball back..." —Rank = 2
...
The above ranks are the rank of the document returned by the search engine.

3.4. From Sentences to DTCs

In order to be able to execute the DTC conjunctive query over the extracted web sentences from the previous section, we need first to transform the extracted sentence to a dependency tree. To do so we use the Minipar dependency parser. But before transforming the extracted sentences to dependency trees, we first filter out interrogative, negated, and conditional sentences, in order to represent only factual and positive sentences. Minipar represents the grammar as a network of nodes representing grammatical categories and the links representing grammatical relationships. Minipar's lexicon is derived from WordNet. In addition to proper names, it contains about 130k entries in base forms. Lexical ambiguities are handled by the parser. Minipar constructs all possible parses for an input sentence, and outputs a single parse tree with the highest ranking. Parsing is based on a manually constructed grammar and is guided by statistical information obtained by parsing a 1GB corpus with Minipar. The resulted dependency trees will then be mapped into OWL-DL as we showed in the section 3.1. All the resulted sentences will be mapped into one knowledge base. Figure 4 shows a graphical representation of a part

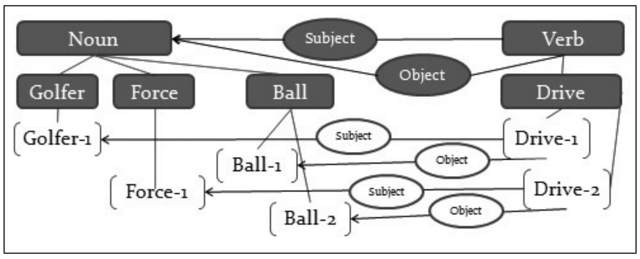


Figure 4: RoDEO Ontology Example

of the dependency ontology created by RoDEO for some of the returned sentences of the example query above. In this graph rectangles represent classes, ovals represent properties, and double brackets represent instances.

3.5. Reasoning over Dependencies Extracted Online

In order to achieve high precision is answering the DTC nRQL query, the query and the resulted OWL-DL knowledge base will be supplied to the reasoner RACER. RACER will try to prove the query over the created knowledge base and return any instances that conform to it.
For example, some of the nouns that were returned by RoDEO for the nRQL query (Retrieve (?x)(AND(?y ?x Subject)(?y ?z Object)(?y Drive)(?x Noun)(?z Ball)) are: *"Golfer, swing, batter, stroke..."*. Notice that introducing the object *"Ball"* that is modifying the verb *"Drive"* returns subjects that are related to one sense of *"Drive"*, that is *"Hit very hard, as by swinging a bat horizontally"*, which is the only sense related to the word *"Ball"*.
The reasoning however is not only in nRQL answering, but also in the possibility of reasoning over word semantic relationship. Such semantic information could be easily integrating from general available ontologies. For example, if a sports ontology has been added to the created ontology of the previous example, the object *"Ball"* will also match to instances under its subclasses, such as the subclasses: *"baseball, basketball, or even a marble..."*
Inferring new information from the available one is also possible by running rules over the created knowledge base. For example, if we create and executed a rule saying that: "if an object is round and is hit or thrown or kicked in games then it is a subclass of the class ball", we will be able to reason over the new inferred information that has been added to the ontology by this rule.
The following section will describe our results in using RoDEO on the task of extracting classes of named entities.

4. Evaluation Application: Extracting classes of Named Entities

In this section we evaluate the extracted collocations using an application of the RoDEO system. The application that we developed over RoDEO is the application of extracting classes of named entities.

4.1. Named Entity Recognition

Named Entity Recognition (NER) as described by the Message Understanding Conferences (MUC)-7 (Chinchor, 1998) is the task consisting of identifying and classifying entities that are considered to belong to one of the following classes: person, location, organization, temporal entities and numeric quantities. Different approaches have been introduced to deal with NER, however two approaches are mainly adopted. The first uses resources, such as gazetteers, and handcrafted rules to match the term to the resources, and the other use machine learning techniques on a tagged corpus in order to learn a set of patterns or to train some sort of a supervised learning algorithm such as the work of (Bikel et al., 1999).

4.2. Extracting classes of Named Entities with RoDEO

Our aim is to automatically extract the most specific class(es) of a selected named entity. For example, we need the ability to extract the class that *"Paul Krugman"* belongs to, in this case a general class would be a "person", but a more specific one would be a "columnist". To accommodate the utmost coverage in selecting fine-grained classes of named entities, many researchers have used the web. Most of the techniques used rely on a set of pattern, and the main difference between one technique and the other is usually the type of patterns used. Some used text patterns such as the work of (Etzioni et al., 2005), other used wrapper or HTML patterns such as the work of (Nadeau et al., 2006). (Etzioni et al., 2005) KNOWITALL system aims to automate the extraction of instances of classes such as the names of scientist from the web by using a set of patterns.

We will be building upon (Etzioni et al., 2005)'s work, however instead of using a set of text patterns over the web, we will be using dependency tree conjunctive query patterns, and instead of learning instances of classes, we will be learning classes of instances. By creating the dependency tree collocation patterns we can then use the RoDEO system to extract dependencies that conform to the selected dependency pattern from the web in order to extract classes of named entities.

One of the dependency tree collocation patterns that we will introduce here is the **predicate noun** pattern. Grammatically, a predicate-noun follows a form of the verb *to be*, like in the sentence: *"Margaret Thatcher was the Prime Minister"*. In this example *"Margaret Thatcher"* is the subject of verb *to be* and *"Prime Minister"* is its predicate noun. As a result, it would be an appropriate pattern to extract classes of named entities. The predicate-noun DTC query pattern is of the form: (Retrieve (?x)(Verb-toBe(?z) AND Predicate(?z ?x) AND Subject(?z ?y) AND Named-Entity(?y))), that is a pattern that looks for ?x having a dependency relationship of type predicate with ?z an instance of verb to be that is having a subject relationship to the named entity in question.

Another pattern is the **appositive pattern**. Appositive is a word that usually describes another word, as in: *"Rudy Giuliani, New York City Mayor, is..."*. The noun *"Mayor"* is an appositive to *"Rudy Giuliani"*. The corresponding dependency tree collocation pattern would be: ((Retrieve (?x) (Noun(?y) AND Named-Entity(?x) AND Appositive(?x ?y)))).

We have also derived other patterns from (Hearst, 1992)'s lexico-syntactic patterns, such as:

1. NP such as NP, (or/and) NP.
   Where NP stands for noun phrase. Example: *Columnist, such as Paul Krugman.*

2. Such NP as NP, (or/and) NP.
   Example: *Work by such columnist as Paul Krugman, and Paul Romer.*

3. NP, or other NP.
   Example: *Paul Krugman, or any other columnist in the N.Y. Times.*

4. NP, and other NP.
   Example: *Read Paul Krugman and other economists and healthcare experts....*

The first of Hearst's patterns, for example is translated to the following DTC query: ((Retrieve (?x)((?x Noun) AND Named-Entity(?y) AND Modify(?x such-as) AND Pcomp-n(such-as ?y)))), where Pcomp-n stands for a nominal complement of a preposition.

Using RoDEO, a dependency tree collocation pattern will return a list of nouns that conforms to a selected DTC query from the web. First we replace the named entity into the Named-Entity DTC patterns, then using RoDEO we run the created DTC pattern that will collect related dependencies and store them into an ontology. We then Match the DTC query using the Reasoner over the created ontology and we count the resulted dependency tree collocations. If the resulted collocation is less than a certain threshold t, we run the next DTC patterns until we have enough collocations returned. The returned collocations are all ranked by Google's document ranking. In order to select the most appropriate noun corresponding to the term in question, we first cluster the returned collocations, and then rank the clusters by collocation frequency. The clustering is simply based on the semantic relatedness of nouns as defined by (Miller, 1995)'s WordNet's hypernym relations. In addition, we filter out the nouns that belong to certain classes that could not represent a named entity class, such as the nouns belonging to the following hypernyms: *"feeling, psychological, status..."*.

Table 1 shows an example of the top 5 returned classes with their ranks for the named entity *"Al Franken"*, grouped by clusters.

Cluster	Class	Rank
Cluster 1	Guy, man, adult male, male...	25
Cluster 1	Author, communicator, person...	12
Cluster 1	Candidate, politician, politico...	7
Cluster 1	Comedian, performer, artist...	5
Cluster 2	Dog, canine, carnivore...	2

Table 1: Named Entity Classes Returned By RoDEO For "Al Franken"

Only the classes of the cluster with the highest frequency are considered as possible types of the named entity, so in this example only cluster 1 is returned.

4.3. Evaluation Results

As we classify named entities into very specific types, we evaluated the application of extracting classes of named entities over a set of 1019 named entities extracted from a shared online database of structured knowledge called FreeBase (Bollacker et al., 2007). FreeBase contains named entities with their general and specific types. For example, according to FreeBase, the named entity: *"Al Franken"* belongs to the following types *"Person, author, writer, and actor"*. The evaluation scoring has been done by comparing our extracted types to the FreeBase types. As the system returned classes do not have to exactly match the FreeBase types, we used the WordNet::Similarity (Pedersen et al., 2004) Path Length method in comparing two types. The path length method is a simple node-counting scheme, which returns a relatedness score between two concepts. The score is inversely proportional to the number of nodes along the shortest path between the synsets in WordNet. The shortest possible path occurs when the two synsets are the same, in which case the length is 1. If the compared types had a relatedness score that is over a threshold t, t=0.21, we considered that it as correct. The threshold has been selected after manually comparing a set of 50 classes. For example, if the returned class is an *"Actor"* for a named entity, and its FreeBase corresponding type is an *"Artist"*, the WordNet::Similarity Path Length method returns a relatedness of 0.25 for the two concepts. As such we assume that the returned class is correct. We have evaluated a total of 1019 named entities. The total number of

different FreeBase types, that these entities belong to is 69 types. The total number of classes returned by our system for the 1019 named entity is 678 types. That shows that our system is returning far more specific results than the FreeBase types. For example, the *"Athlete"* FreeBase type has been matched to *"Blocker, bowler, boxer, cornerback, cricketer, footballer, keeper, receiver, scorer, skater, swimmer, tackle..."*.
To compute the accuracy of the extracted classes of a single named entity we use the following:

$$\text{Accuracy} = \frac{\text{Number of correct types}}{\text{Total number of types}}$$

The total accuracy of the system is computed as the average of the accuracy for all the evaluated named entities.
Overall, the application achieved an accuracy of 0.7. Table 2 shows some of the accuracy results grouped by types and sub-types. For example, for the high level *"Person"* type the accuracy achieved is up to 0.87, whereas the type *"Company"* achieved an accuracy of 0.62. The person type can be subdivided into several subtypes, for example the *"Actor"* type achieved an accuracy of 0.78.

Types	Accuracy	Subtypes	Accuracy
Person	0.87	...	...
		Actor	0.78
		Athlete	0.76
		Author	0.75
		...	...
		Publisher	0.14
		...	...
Company	0.62	...	...
		Airline	0.66
		Employer	0.34
		Owner	0.31
		...	...
		Chain	0.16
		...	...

Table 2: Sample Of The Evaluation Results

There are many related work in the named entity recognition and classification field; however most of the available work fall under the initially task set at the MUC conference for identifying and classifying named entities into five classes, which is much easier than classifying named entities into more fine grained classes. Most methods that classifies named entities into five classes achieved an accuracy of well above 90%. However, this has not been the case when classifying named entities into more fine grained classes. As such, we are going to focus our comparison to some of the approaches that classify named entities into more than just five classes. Table 3 shows a comparison of some of these approaches ordered by the number of classes they consider. (Cimiano and Staab, 2004)'s PANKOW system is a leixco-syntactic pattern based system that uses the web frequency to select the appropriate class from a set of 59 classes. The PANKOW system achieved an accuracy of 24.9%. (Nadeau, 2007)'s BaLIE system uses semi-supervised machine learning and the web to classify named entities into 100 classes. It achieved an accuracy of 57.4%. BaLIE creates large gazetteers of named entities, using a hand crafted HTML markup in web pages and a seed of named entities, and then uses a simple heuristic to identify and classify named entities. (Sekine, 2004)'s system achieved 72% by classifying named entities into 200 classes, however they used about 1,400 handcrafted rules and a dictionary of 130,000 instances that are classified into the 200 classes. The last system is (Alfonseca and Manandhar, 2002)'s system that adopted a vector space model having syntactic dependencies as vector features, and compared the named entity vector into the most similar vector. They had considered 1200 classes and achieved an accuracy of 17.39% using the verb/object dependencies as a feature.

Systems	Types	Accuracy
MUC	5	>90%
PANKOW	59	24.9%
BaLIE	100	57.4%
Sekine's tagger	200	72%
RoDEO	678	70%
Alfonseca's system	1200	17.39%

Table 3: Comparison Table

Although we are extracting a large number of fine grained classes, we are not classifying the named entities into these set of classes, but extracting the most frequent classes associated with each named entities. We notice from this comparison that RoDEO's accuracy is comparable to the system using hand crafted rules, although we are extracting a much larger number of classes.
While analyzing the system results, we noticed that some of the low scores are a result of the restriction that we have set on the RoDEO system regarding the total number of returned collocations. It seems that the number is too low, and increasing it would probably boost the overall system accuracy. In addition, the similarity path length method that was used for the scoring is not very adequate. For example the *"Chain"* concept relatedness score to the *"Company"* concept is 0.2, which is less than the threshold set. As such a *"Chain"* is not a treated as of type *"Company"*. At the same time, if we lower the threshold to less than 0.21, then concepts such as a *"Person* and a *"Set"* would be related.
It should be noted that comparing the results of extracting classes of named entity is also an issue by itself. As many techniques have been proposed for the ranking of named entity recognition and classification task, a recent survey of named entity recognition and classification systems (Nadeau and Sekine, 2007) has showed that a score of a simple example made of only five named entities, varied between 20 and 40 %, using three scoring techniques that have been used in the major conferences related to named entity recognition and classification.

5. Conclusion and Future Work

In this paper we have presented a system for extracting dependency tree collocations online, using a dependency parser, an advanced representation based on description logic, and a reasoner. We showed the usefulness of such a system in extracting classes of named entities, and the usefulness of using the web as a corpus. Without the quantity of the text that the web provides, such an application would not have been possible.

In our future work, we plan to enhance the extraction of classes of named entities, mainly by increasing the threshold of returned collocations, and integrating the resulted ontology with a named entity types ontology which will automatically reason over these types without relying on WordNet Similarity. In addition, we would like to show the usefulness of the RoDEO system in other NLP applications, such as in semantic relations of noun compounds, in commonsense rule discovery, and in other applications.

6. References

E. Alfonseca and S. Manandhar. 2002. Extending a lexical ontology by a combination of distributional semantics signatures. *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2002)*, pages 1–7.

S. Bartsch. 2004. *Structural and Functional Properties of Collocations in English.: a corpus study of lexical and pragmatic constraints on lexical co-occurrence*. Gunter Narr Verlag.

T. Berners-Lee. 1998. Semantic Web Road Map. *World Wide Web Consortium (W3C). http://www.w3.org/DesignIssues/Semantic.html*.

D.M. Bikel, R. Schwartz, and R.M. Weischedel. 1999. An Algorithm that Learns What's in a Name. *Machine Learning*.

K. Bollacker, R. Cook, and P. Tufts. 2007. Freebase: A shared database of structured general human knowledge. *Proceedings of the National Conference on Artificial Intelligence*, 22(2).

N. Chinchor. 1998. Muc-7 named entity task definition. *Proceedings of the 7th Message Understanding Conference (MUC-7)*.

P. Cimiano and S. Staab. 2004. Learning by googling. *ACM SIGKDD Explorations Newsletter*, 6(2):24–33.

M. Dean, G. Schreiber, et al. 2004. OWL Web Ontology Language Reference. *W3C Recommendation*, 10.

O. Etzioni, M. Cafarella, D. Downey, A.M. Popescu, T. Shaked, S. Soderland, D.S. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

Stefan Evert. 2005. *The Statistics of Word Cooccurrences Word Pairs and Collocations*. Ph.D. thesis, University of Stuttgart.

R. Fikes, P. Hayes, and I. Horrocks. 2004. OWL-QLa language for deductive query answering on the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1):19–29.

J.R. Firth. 1957. *Papers in linguistics 1934-1951*. Oxford University Press New York.

V. Haarslev and R. Moller. 2003. Racer: A core inference engine for the semantic web. *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools*, pages 27–36.

V. Haarslev, R. Moller, and M. Wessel. 2004. Querying the Semantic Web with Racer+ nRQL. *Proceedings of the KI-04 Workshop on Applications of Description Logics*.

Z.S. Harris. 1968. *Mathematical Structures of Language*. Interscience Publishers New York.

D.G. Hays. 1964. Dependency theory: A formalism and some observations. *Language*, 40(4):511–525.

M.A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. *In Proceedings of the 14th International Conference on Computational Linguistics*.

I. Horrocks and P. Patel-Schneider. 2004. Reducing OWL entailment to description logic satisfiability. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(4):345–357.

A. Kilgarriff, P. Rychly, P. Smrz, and D. Tugwell. 2004. The Sketch Engine. *Proceedings of Euralex*, pages 105–116.

D. Lin. 1998a. Dependency-based evaluation of minipar. *Workshop on the Evaluation of Parsing Systems*, pages 317–330.

D. Lin. 1998b. Extracting collocations from text corpora. *First Workshop on Computational Terminology*, pages 57–63.

C.D. Manning and H. Sch'eutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

T. McEnery and A. Wilson. 2001. *Corpus Linguistics*. Edinburgh University Press.

G.A. Miller. 1995. WordNet: A Lexical Database for English. *COMMUNICATIONS OF THE ACM*, 38(11):39.

D. Nadeau and S. Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes*.

D. Nadeau, P.D. Turney, and S. Matwin. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. *19th Canadian Conference on Artificial Intelligence*.

David Nadeau. 2007. *Semi-Supervised Named Entity Recognition: Learning to Recognize 100 Entity Types with Little Supervision*. Ph.D. thesis, University of Ottawa, November.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. WordNet::Similarity-Measuring the Relatedness of Concepts. *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*.

A. Renouf, A. Kehoe, and J. Banerjee. 2007. WebCorp: an integrated system for web text search. *Corpus Linguistics and the Web*.

P. Resnik and A. Elkiss. 2005. The Linguists Search Engine: An Overview. *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 33–36.

S. Sekine. 2004. Definition, dictionaries and tagger for Extended Named Entity Hierarchy. *Actes LREC*.

M. Wessel and R. Moller. 2006. A Flexible DL-based Architecture for Deductive Information Systems. *IJCAR Workshop on Empirically Successful Computerized Reasoning (ESCoR)*, pages 92–111.