HLT-NAACL 2007

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing

Proceedings of the Workshop

26 April, 2007
Rochester, NY, USA

Production and Manufacturing by
Omnipress Inc.
Post Office Box 7214
Madison, WI 53707-7214

TextGraphs-2 Workshop at HLT-NAACL 2007
was sponsored by the University of Leipzig, Germany

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
75 Paterson Street, Suite 9
New Brunswick, NJ 08901
USA
Tel: +1-732-342-9100
Fax: +1-732-342-9339
acl@aclweb.org

PREFACE

Recent years have shown an increased interest in bringing the field of graph theory into Natural Language Processing. In many NLP applications entities can be naturally represented as nodes in a graph and relations between them can be represented as edges. Recent research has shown that graph-based representations of linguistic units as diverse as words, sentences and documents give rise to novel and efficient solutions in a variety of NLP tasks, ranging from part of speech tagging, word sense disambiguation and parsing to information extraction, semantic role assignment, summarization and sentiment analysis.

This volume contains papers accepted for presentation at the TextGraphs-2 2007 Workshop on Graph-Based Algorithms for Natural Language Processing. This event took place on April 26, 2007, in Rochester, NY, USA, immediately following the HLT-NAACL Human Language Technologies Conference. It was the second workshop on this topic, building on the success of the first TextGraphs workshop at HLT-NAACL 2006. The workshop aimed at bringing together researchers working on problems related to the use of graph-based algorithms for Natural Language Processing and on the theory of graph-based methods. It addressed a broad spectrum of research areas to foster exchange of ideas and help to identify principles of using the graph notions that go beyond an ad-hoc usage. Unveiling these principles will give rise to applying generic graph methods to many new problems that can be encoded in this framework.

We issued calls for both regular and short, late-breaking papers. In total, ten regular and three short papers were accepted for presentation, considering the careful reviews of our program committee. We are indebted to all program committee members for their thoughtful, high quality and elaborate reviews, especially considering our extremely tight time frame for reviewing. The papers appearing in this volume have surely benefited from their expert feedback.

This year's workshop attracted papers employing graphs in a wide range of settings. While some contributions focus on analyzing the structure of graphs induced by language data or the interaction of processes on various levels, others use graphs as a means for data representation to solve NLP tasks, sometimes involving transformations on the graph structure.

H. F. Witschel introduces a new graph based meta model for Information Retrieval that subsumes many previous retrieval models and supports different forms of search. Improved unigram language models by a smoothing technique that accounts for word similarities are constructed by B. Jedynak and D. Karakos. Unsupervised grammar induction using latent semantics is the topic of A. M. Olney's research. V. Jijkoun and M. de Rijke view NLP tasks as graph transformations of labelled, directed graphs and experiment with tasks involving syntax and semantics. Syntactic dependency trees as a basis for semantic similarity are applied to textual entailment by D. Micol et al. D. Leite et al. find in their graph-based automatic summarization experiments that linguistic knowledge is necessary to improve automatic extracts. For multi-document summarization, evolving timestamped graphs are employed in the contribution of Z. Lin and M.-Y. Kan. A graph for the extraction of patterns combined with an extension of chance discovery is applied by C. S. Montero and K. Araki to human-computer dialogue mining. The small-world and scale-free property of linguistic graphs that go in hand with power-law distributions on entity and entity pair frequencies are examined in four papers: T. Zesch and I. Gurevych analyze the article and category graph of Wikipedia and measure correlation with WordNet. R. Ferrer

i Cancho et al. find correlations in the organization of syntactic dependency networks for a wide range of languages. Co-occurrence degree distributions are examined in a comparative study of Russian and English by V. Kapustin and A. Jamsen. In the setting of spell checking, M. Choudhury et al. find that spelling error probabilities for different languages are proportional to the average weighted degree of the corresponding SpellNet. A transductive classification algorithm based on graph clustering is described by K. Ganchev and F. Pereira, and tested on various NLP tasks.

Finally, having a prominent researcher as an invited speaker greatly contributes to the quality of the workshop. We thank Andrew McCallum for his talk and for the support that his prompt acceptance provided to the workshop.

Chris Biemann, Irina Matveeva, Rada Mihalcea and Dragomir Radev
April 2007

CHAIRS:

Chris Biemann, University of Leipzig, Germany
Irina Matveeva, University of Chicago, USA
Rada Mihalcea, University of North Texas, USA
Dragomir Radev, University of Michigan, USA

PROGRAM COMMITTEE:

Eneko Agirre, University of the Basque Country, Spain
Monojit Choudhury, Indian Institute of Technology
Diane Cook, Washington State University
Hal Daumé III, University of Utah
Gael Dias, Beira Interior University, Portugal
Güneş Erkan, University of Michigan
Michael Gamon, Microsoft Research
Bruno Gaume, IRIT, France
Andrew Goldberg, University of Wisconsin
Samer Hassan, University of North Texas
Hany Hassan, IBM, Egypt
Rosie Jones, Yahoo Research
Fabio Massimo Zanzotto, University of Rome, Italy
Andrew McCallum, University of Massachusetts Amherst
Ani Nenkova, Stanford University
Patrick Pantel, USC Information Sciences Institute
Uwe Quasthoff, University of Leipzig
Aitor Soroa, University of the Basque Country, Spain
Simone Teufel, Cambridge University, UK
Kristina Toutanova, Microsoft Research
Lucy Vanderwende, Microsoft Research
Dominic Widdows, Maya Design
Florian Wolf, F-W Consulting
Xiaojin Zhu, University of Wisconsin

INVITED SPEAKER:

Andrew McCallum, University of Massachusetts Amherst

WEBSITE:

http://www.textgraphs.org/ws07

Table of Contents

Conference Program

Thursday, April 26, 2007

8:45–9:00 Opening Remarks

Session 1: Session One

09:00–10:00 Invited Talk by Andrew McCallum

10:00–10:25 *Analysis of the Wikipedia Category Graph for NLP Applications*
Torsten Zesch and Iryna Gurevych

10:30–11:00 Coffee Break

Session 2: Session Two

11:00–11:25 *Multi-level Association Graphs - A New Graph-Based Model for Information Retrieval*
Hans Friedrich Witschel

11:25–11:50 *Extractive Automatic Summarization: Does more Linguistic Knowledge Make a Difference?*
Daniel S. Leite, Lucia H. M. Rino, Thiago A. S. Pardo and Maria das Graças V. Nunes

11:50–12:15 *Timestamped Graphs: Evolutionary Models of Text for Multi-Document Summarization*
Ziheng Lin and Min-Yen Kan

12:15–12:30 *Unigram Language Models using Diffusion Smoothing over Graphs*
Bruno Jedynak and Damianos Karakos

12:30–14:00 Lunch Break

Thursday, April 26, 2007 (continued)

Session 3: Session Three

14:00–14:25 *Transductive Structured Classification through Constrained Min-Cuts*
Kuzman Ganchev and Fernando Pereira

14:25–14:50 *Latent Semantic Grammar Induction: Context, Projectivity, and Prior Distributions*
Andrew M Olney

14:50–15:15 *Learning to Transform Linguistic Graphs*
Valentin Jijkoun and Maarten de Rijke

15:15–15:30 *Semi-supervised Algorithm for Human-Computer Dialogue Mining*
Calkin S. Montero and Kenji Araki

15:30–16:00 Coffee Break

Session 4: Session Four

16:00–16:25 *Correlations in the Organization of Large-Scale Syntactic Dependency Networks*
Ramon Ferrer i Cancho, Alexander Mehler, Olga Pustylnikov and Albert Diaz-Guilera

16:25–16:50 *DLSITE-2: Semantic Similarity Based on Syntactic Dependency Trees Applied to Textual Entailment*
Daniel Micol, Óscar Ferrández, Rafael Muñoz and Manuel Palomar

16:50–17:15 *How Difficult is it to Develop a Perfect Spell-checker? A Cross-Linguistic Analysis through Complex Network Approach*
Monojit Choudhury, Markose Thomas, Animesh Mukherjee, Anupam Basu and Niloy Ganguly

17:15–17:30 *Vertex Degree Distribution for the Graph of Word Co-Occurrences in Russian*
Victor Kapustin and Anna Jamsen

Analysis of the Wikipedia Category Graph for NLP Applications

Torsten Zesch and Iryna Gurevych
Ubiquitous Knowledge Processing Group
Telecooperation Division
Darmstadt University of Technology, Hochschulstraße 10
D-64289 Darmstadt, Germany
{zesch,gurevych} (at) tk.informatik.tu-darmstadt.de

Abstract

In this paper, we discuss two graphs in Wikipedia (i) the article graph, and (ii) the category graph. We perform a graph-theoretic analysis of the category graph, and show that it is a scale-free, small world graph like other well-known lexical semantic networks. We substantiate our findings by transferring semantic relatedness algorithms defined on WordNet to the Wikipedia category graph. To assess the usefulness of the category graph as an NLP resource, we analyze its coverage and the performance of the transferred semantic relatedness algorithms.

1 Introduction

Wikipedia[1] is a free multi-lingual online encyclopedia that is constructed in a collaborative effort of voluntary contributors and still grows exponentially. During this process, Wikipedia has probably become the largest collection of freely available knowledge. A part of this knowledge is encoded in the network structure of Wikipedia pages. In particular, Wikipedia articles form a network of semantically related terms, while the categories are organized in a taxonomy-like structure called **Wikipedia Category Graph** (**WCG**).

In this paper, we perform a detailed analysis of the WCG by computing a set of graph-theoretic parameters, and comparing them with the parameters reported for well-known graphs and classical lexical semantic networks. We show that the WCG, which is constructed collaboratively, shares many properties with other lexical semantic networks, such as WordNet (Fellbaum, 1998) or Roget's Thesaurus[2] that are constructed by expert authors. This implies that the WCG can be used as a resource in NLP applications, where other semantic networks have been traditionally employed.

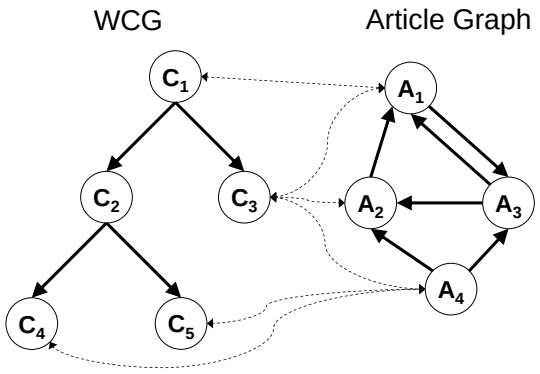


Figure 1: Relations between article graph and WCG.

To further evaluate this issue, we adapt algorithms for computing semantic relatedness on classical semantic networks like WordNet to the WCG. We evaluate their performance on the task of computing semantic relatedness using three German datasets, and show that WCG based algorithms perform very well.

Article graph Wikipedia articles are heavily linked, as links can be easily inserted while editing an article. If we treat each article as a node, and each link between articles as an edge running from one node to another, then Wikipedia articles form a directed graph (see right side of Figure 1). The article graph has been targeted by numerous studies, and is not addressed in this paper. Buriol et al. (2006) analyze the development of the article graph over time, and find that some regions are fairly stable, while others are advancing quickly. Zlatic et al.

[1] http://www.wikipedia.org

[2] http://thesaurus.reference.com

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 1–8,
Rochester, April 2007

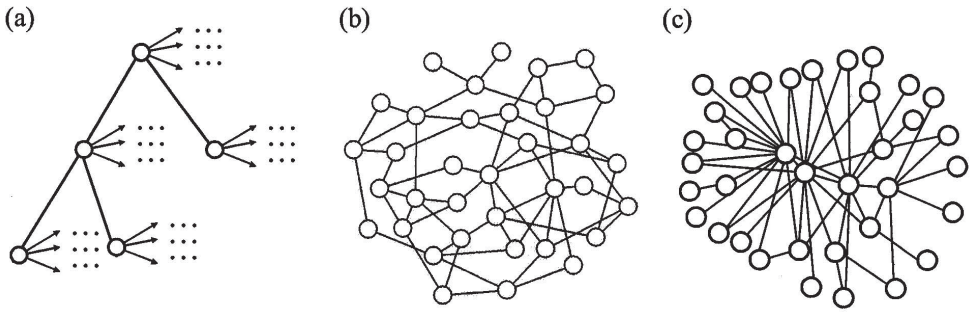


Figure 2: Structures of semantic networks after Steyvers and Tenenbaum (2005). a) a taxonomy, b) an arbitrary graph, c) scale-free, small-world graph.

(2006) give a comprehensive overview of the graph parameters for the largest languages in Wikipedia. Capocci et al. (2006) study the growth of the article graph and show that it is based on preferential attachment (Barabasi and Albert, 1999). Voss (2005) shows that the article graph is scale-free and grows exponentially.

Category graph Categories in Wikipedia are organized in a taxonomy-like structure (see left side of Figure 1 and Figure 2-a). Each category can have an arbitrary number of subcategories, where a subcategory is typically established because of a hyponymy or meronymy relation. For example, a category *vehicle* has subcategories like *aircraft* or *watercraft*. Thus, the WCG is very similar to semantic wordnets like WordNet or GermaNet (Kunze, 2004). As Wikipedia does not strictly enforce a taxonomic category structure, cycles and disconnected categories are possible, but rare. In the snapshot of the German Wikipedia[3] from May 15, 2006, the largest connected component in the WCG contains 99,8% of all category nodes, as well as 7 cycles.

In Wikipedia, each article can link to an arbitrary number of categories, where each category is a kind of semantic tag for that article. A category backlinks to all articles in this category. Thus, article graph and WCG are heavily interlinked (see Figure 1), and most studies (Capocci et al., 2006; Zlatic et al., 2006) have not treated them separately. However, the WCG should be treated separately, as it differs from the article graph. Article links are established because of any kind of relation between articles, while links between categories are typically established because of hyponymy or meronymy relations.

Holloway et al. (2005) create and visualize a category map based on co-occurrence of categories. Voss (2006) pointed out that the WCG is a kind of thesaurus that combines collaborative tagging and hierarchical indexing. Zesch et al. (2007a) identified the WCG as a valueable source of lexical semantic knowledge, but did not analytically analyze its properties. However, even if the WCG seems to be very similar to other semantic wordnets, a graph-theoretic analysis of the WCG is necessary to substantiate this claim. It is carried out in the next section.

2 Graph-theoretic Analysis of the WCG

A graph-theoretic analysis of the WCG is required to estimate, whether graph based semantic relatedness measures developed for semantic wordnets can be transferred to the WCG. This is substantiated in a case study on computing semantic relatedness in section 4.

For our analysis, we treat the directed WCG as an undirected **graph** $G := (V, E)$,[4] as the relations connecting categories are reversible. V is a set of vertices or nodes. E is a set of unordered pairs of distinct vertices, called edges. Each page is treated as a **node** n, and each link between pages is modeled as an **edge** e running between two nodes.

Following Steyvers and Tenenbaum (2005), we characterize the graph structure of a lexical semantic resource in terms of a set of graph parameters: The

[3] Wikipedia can be downloaded from `http://download.wikimedia.org/`

[4] Newman (2003) gives a comprehensive overview about the theoretical aspects of graphs.

PARAMETER	*Actor*	*Power*	*C.elegans*	*AN*	*Roget*	*WordNet*	$Wiki_{Art}$	**WCG**
$\|V\|$	225,226	4,941	282	5,018	9,381	122,005	190,099	27,865
D	-	-	-	5	10	27	-	17
$\overline{k}$	61.0	2.67	14.0	22.0	49.6	4.0	-	3.54
γ	-	-	-	3.01	3.19	3.11	2.45	2.12
$\overline{L}$	3.65	18.7	2.65	3.04	5.60	10.56	3.34	7.18
$\overline{L}_{random}$	2.99	12.4	2.25	3.03	5.43	10.61	~ 3.30	~ 8.10
C	0.79	0.08	0.28	0.186	0.87	0.027	~ 0.04	0.012
C_{random}	0.0003	0.005	0.05	0.004	0.613	0.0001	~ 0.006	0.0008

Table 1: Parameter values for different graphs.

Values for *Actor* (collaboration graph of actors in feature films), *Power* (the electrical power grid of the western United States) and *C.elegans* (the neural network of the nematode worm C. elegans) are from Watts and Strogatz (1998). Values for AN (a network of word associations by Nelson et al. (1998)), Roget's thesaurus and WordNet are from Steyvers and Tenenbaum (2005). Values for $Wiki_{art}$ (German Wikipedia article graph) are from Zlatic et al. (2006). We took the values for the page set labelled M on their website containing 190,099 pages for German, as it comes closest to a graph of only articles. Values marked with '-' in the table were not reported in the studies. The values for the WCG are computed in this study.

degree k of a node is the number of edges that are connected with this node. Averaging over all nodes gives the **average degree** $\overline{k}$. The degree distribution $P(k)$ is the probability that a random node will have degree k. In some graphs (like the WWW), the degree distribution follows a power law $P(k) \approx k^{-\gamma}$ (Barabasi and Albert, 1999). We use the **power law exponent** γ as a graph parameter.

A **path** $p_{i,j}$ is a sequence of edges that connects a node n_i with a node n_j. The **path length** $l(p_{i,j})$ is the number of edges along that path. There can be more than one path between two nodes. The **shortest path length** L is the minimum of all these paths, i.e. $L_{i,j} = \min l(p_{i,j})$. Averaging over all nodes gives the **average shortest path length** $\overline{L}$. The **diameter** D is the maximum of the shortest path lengths between all pairs of nodes in the graph.

The **cluster coefficient** of a certain node n_i can be computed as

$$C_i = \frac{T_i}{\frac{k_i(k_i-1)}{2}} = \frac{2T_i}{k_i(k_i-1)}$$

where T_i refers to the number of edges between the neighbors of node n_i and $k_i(k_i-1)/2$ is the maximum number of edges that can exist between the k_i neighbors of node n_i.[5] The cluster coefficient C for the whole graph is the average of all C_i. In a fully connected graph, the cluster coefficient is 1.

[5]In a social network, the cluster coefficient measures how many of my friends (neighboring nodes) are friends themselves.

For our analysis, we use a snapshot of the German Wikipedia from May 15, 2006. We consider only the largest connected component of the WCG that contains 99,8% of the nodes. Table 1 shows our results on the WCG as well as the corresponding values for other well-known graphs and lexical semantic networks. We compare our empirically obtained values with the values expected for a random graph. Following Zlatic et al. (2006), the cluster coefficient C for a random graph is

$$C_{random} = \frac{(\overline{k}^2 - \overline{k})^2}{|V|\overline{k}}$$

The average path length for a random network can be approximated as $\overline{L}_{random} \approx \log|V| \ / \ \log\overline{k}$ (Watts and Strogatz, 1998).

From the analysis, we conclude that all graphs in Table 1 are small world graphs (see Figure 2-c). Small world graphs (Watts and Strogatz, 1998) contain local clusters that are connected by some long range links leading to low values of $\overline{L}$ and D. Thus, small world graphs are characterized by (i) small values of $\overline{L}$ (typically $\overline{L} \gtrsim \overline{L}_{random}$), together with (ii) large values of C ($C \gg C_{random}$).

Additionally, all semantic networks are scale-free graphs, as their degree distribution follows a power law. Structural commonalities between the graphs in Table 1 are assumed to result from the growing process based on preferential attachment (Capocci et al., 2006).

Our analysis shows that $WordNet$ and the WCG are (i) scale-free, small world graphs, and (ii) have a very similar parameter set. Thus, we conclude that algorithms designed to work on the graph structure of WordNet can be transferred to the WCG.

In the next section, we introduce the task of computing semantic relatedness on graphs and adapt existing algorithms to the WCG. In section 4, we evaluate the transferred algorithms with respect to correlation with human judgments on SR, and coverage.

3 Graph Based Semantic Relatedness Measures

Semantic similarity (**SS**) is typically defined via the lexical relations of synonymy ($automobile$ – car) and hypernymy ($vehicle$ – car), while **semantic relatedness** (**SR**) is defined to cover any kind of lexical or functional association that may exist between two words (Budanitsky and Hirst, 2006). Dissimilar words can be semantically related, e.g. via functional relationships ($night$ – $dark$) or when they are antonyms ($high$ – low). Many NLP applications require knowledge about semantic relatedness rather than just similarity (Budanitsky and Hirst, 2006).

We introduce a number of competing approaches for computing semantic relatedness between words using a graph structure, and then discuss the changes that are necessary to adapt semantic relatedness algorithms to work on the WCG.

3.1 Wordnet Based Measures

A multitude of semantic relatedness measures working on semantic networks has been proposed.

Rada et al. (1989) use the path length (**PL**) between two nodes (measured in edges) to compute semantic relatedness.

$$dist_{PL} = l(n_1, n_2)$$

Leacock and Chodorow (1998, ***LC***) normalize the path-length with the depth of the graph,

$$sim_{LC}(n_1, n_2) = -\log \frac{l(n_1, n_2)}{2 \times depth}$$

where $depth$ is the length of the longest path in the graph.

Wu and Palmer (1994, ***WP***) introduce a measure that uses the notion of a lowest common subsumer of two nodes $lcs(n_1, n_2)$. In a directed graph, a lcs is the parent of both child nodes with the largest depth in the graph.

$$sim_{WP} = \frac{2\, depth(lcs)}{l(n_1, lcs) + l(n_2, lcs) + 2\, depth(lcs)}$$

Resnik (1995, ***Res***), defines semantic similarity between two nodes as the information content (IC) value of their lcs. He used the relative corpus frequency to estimate the information content value.

Jiang and Conrath (1997, ***JC***) additionally use the IC of the nodes.

$$dist_{JC}(n_1, n_2) = IC(n_1) + IC(n_2) - 2IC(lcs)$$

Note that JC returns a distance value instead of a similarity value.

Lin (1998, ***Lin***) defined semantic similarity using a formula derived from information theory.

$$sim_{Lin}(n_1, n_2) = 2 \times \frac{IC(lcs)}{IC(n_1) + IC(n_2)}$$

Because polysemous words may have more than one corresponding node in a semantic wordnet, the resulting semantic relatedness between two words w_1 and w_2 can be calculated as

$$SR = \begin{cases} \min\limits_{n_1 \in s(w_1), n_2 \in s(w_2)} dist(n_1, n_2) & \text{path} \\ \max\limits_{n_1 \in s(w_1), n_2 \in s(w_2)} sim(n_1, n_2) & \text{IC} \end{cases}$$

where $s(w_i)$ is the set of nodes that represent senses of word w_i. That means, the relatedness of two words is equal to that of the most related pair of nodes.

3.2 Adapting SR Measures to Wikipedia

Unlike other wordnets, nodes in the WCG do not represent synsets or single terms, but a generalized concept or category. Therefore, we cannot use the WCG directly to compute SR. Additionally, the WCG would not provide sufficient coverage, as it is relatively small. Thus, transferring SR measures to the WCG requires some modifications. The task of estimating SR between terms is casted to the task of SR between Wikipedia articles devoted to these terms. SR between articles is measured via the categories assigned to these articles.

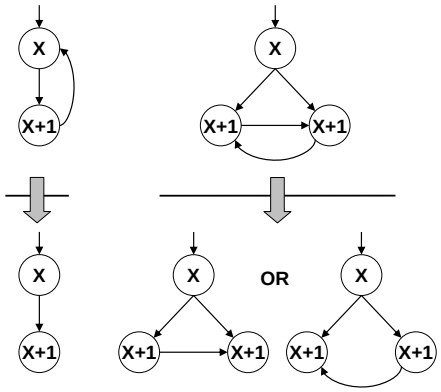


Figure 3: Breaking cycles in the WCG.

We define C_1 and C_2 as the set of categories assigned to article a_i and a_j, respectively. We then determine the SR value for each category pair (c_k, c_l) with $c_k \in C_1$ and $c_l \in C_2$. We choose the best value among all pairs (c_k, c_l), i.e. the minimum for path based and the maximum for information content based measures.

$$SR_{best} = \begin{cases} \min\limits_{c_k \in C_1, c_l \in C_2} (sr(c_k, c_l)) & \text{path based} \\ \max\limits_{c_k \in C_1, c_l \in C_2} (sr(c_k, c_l)) & \text{IIC based} \end{cases}$$

See (Zesch et al., 2007b) for a more detailed description of the adaptation process.

We substitute Resnik's information content with the **intrinsic information content** (**IIC**) by Seco et al. (2004) that is computed only from structural information of the underlying graph. It yields better results and is corpus independent. The IIC of a node n_i is computed as a function of its hyponyms,

$$IIC(n) = 1 - \frac{\log(hypo(n_i) + 1}{\log(|C|)}$$

where $hypo(n_i)$ is the number of hyponyms of node n_i and $|C|$ is the number of nodes in the taxonomy.

Efficiently counting the hyponyms of a node requires to break cycles that may occur in a WCG. We perform a colored depth-first-search to detect cycles, and break them as visualized in Figure 3. A link pointing back to a node closer to the top of the graph is deleted, as it violates the rule that links in the WCG typically express hyponymy or meronymy relations. If the cycle occurs between nodes on the same level, we cannot decide based on that rule and simply delete one of the links running on the same level. This strategy never disconnects any nodes from a connected component.

4 Semantic Relatedness Experiments

A commonly accepted method for evaluating SR measures is to compare their results with a gold standard dataset based on human judgments on word pairs.[6]

4.1 Datasets

To create gold standard datasets for evaluation, human annotators are asked to judge the relatedness of presented word pairs. The average annotation scores are correlated with the SR values generated by a particular measure.

Several datasets for evaluation of semantic relatedness or semantic similarity have been created so far (see Table 2). Rubenstein and Goodenough (1965) created a dataset with 65 English noun pairs (**RG65** for short). A subset of RG65 has been used for experiments by Miller and Charles (1991, **MC30**) and Resnik (1995, **Res30**).

Finkelstein et al. (2002) created a larger dataset for English containing 353 pairs (**Fin353**), that has been criticized by Jarmasz and Szpakowicz (2003) for being culturally biased. More problematic is that Fin353 consists of two subsets, which have been annotated by a different number of annotators. We performed further analysis of their dataset and found that the inter-annotator agreement[7] differs considerably. These results suggest that further evaluation based on this data should actually regard it as two independent datasets.

As Wikipedia is a multi-lingual resource, we are not bound to English datasets. Several German datasets are available that are larger than the existing English datasets and do not share the problems of the Finkelstein datasets (see Table 2). Gurevych (2005) conducted experiments with a German translation of an English dataset (Rubenstein and Goodenough, 1965), but argued that the dataset is too small and only contains noun-noun pairs connected

[6] Note that we do not use multiple-choice synonym question datasets (Jarmasz and Szpakowicz, 2003), as this is a different task, which is not addressed in this paper.

[7] We computed the correlation for all annotators pairwise and summarized the values using a Fisher Z-value transformation.

Dataset	Year	Language	# Pairs	POS	Type	Scores	# Subjects	Correlation r Inter	Correlation r Intra
RG65	1965	English	65	N	SS	continuous 0–4	51	-	.850
MC30	1991	English	30	N	SS	continuous 0–4	38	-	-
Res30	1995	English	30	N	SS	continuous 0–4	10	.903	-
Fin353	2002	English	353	N, V, A	SR	continuous 0–10	13/16	-	-
			153				13	.731	-
			200				16	.549	-
Gur65	2005	German	65	N	SS	discrete {0,1,2,3,4}	24	.810	-
Gur350	2006	German	350	N, V, A	SR	discrete {0,1,2,3,4}	8	.690	-
ZG222	2006	German	222	N, V, A	SR	discrete {0,1,2,3,4}	21	.490	.647

Table 2: Comparison of German datasets used for evaluating semantic relatedness.

by either synonymy or hyponymy. Thus, she created a larger German dataset containing 350 word pairs (**Gur350**). It contains nouns, verbs and adjectives that are connected by classical and non-classical relations (Morris and Hirst, 2004). However, word pairs for this dataset are biased towards strong classical relations, as they were manually selected. Thus, Zesch and Gurevych (2006) used a semi-automatic process to create word pairs from domain-specific corpora. The resulting **ZG222** dataset contains 222 word pairs that are connected by all kinds of lexical semantic relations. Hence, it is particularly suited for analyzing the capability of a measure to estimate SR.

4.2 Results and Discussion

Figure 4 gives an overview of our experimental results of evaluating SR measures based on the WCG on three German datasets. We use Pearson's product moment correlation r to compare the results with human judgments. From each dataset, we only use word pairs where Wikipedia articles corresponding to these words are available (see section 4.3 for a detailed discussion of word pair coverage). For comparison, we give the best results obtained by GermaNet based measures (abbreviated as **GN**).[8]

Our results show that the graph-based SR measures have been successfully transferred to the WCG. Results on the Gur65 dataset (containing only word pairs connected by strong classical relations) are lower than values computed using GermaNet. This is to be expected, as the WCG is created collaboratively without strictly enforcing a certain type

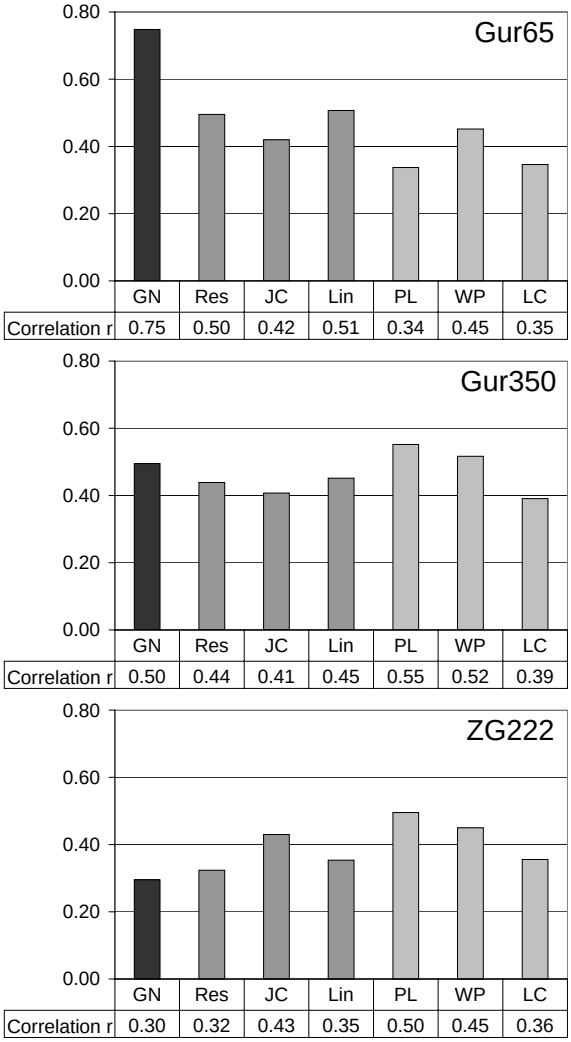


Figure 4: Correlations on different datasets.

[8] Additionally, Table 2 gives the inter annotator agreement for each subset. It constitutes an upper bound of a measure's performance on a certain dataset.

of semantic relation between categories, while GermaNet is carefully modelled to represent the strong classical relations captured by Gur65. Results on the two other datasets, which contain a majority of word pairs connected by non-classical semantic relations, show that the WCG is better suited than GermaNet to estimate SR.

Performance of WCG based measures depends on the dataset and the kind of knowledge used. IIC based measures (Res, JC and Lin) outperform path based measures (PL, LC and WP) on the Gur65 dataset, while path based measures are clearly better on SR datasets (Gur350 and ZG222). The impressive performance of the simple PL measure on the SR datasets cannot be explained with the structural properties of the WCG, as they are very similar to those of other semantic networks. Semantically related terms are very likely to be categorized under the same category, resulting in short path lengths leading to high SR. The generalization process that comes along with classification seems to capture the phenomenon of SR quite well. As each article can have many categories, different kinds of semantic relations between terms can be established, but the type of relation remains unknown.

4.3 Coverage of Word Pairs

If the WCG is to be used as a lexical semantic resource in large scale NLP applications, it should provide broad coverage. As was described in section 3.2, computing SR using the WCG relies on categories assigned to articles. Thus, we consider a word to be covered by the WCG, if there is a categorized article with matching title.

Table 3 gives an overview of the number of word pairs covered in GermaNet or the WCG. Only few words from Gur65 were not found in one of the resources. This proportion is much higher for Gur350 and ZG222, as these datasets contain many domain specific terms that are badly covered in GermaNet, and many word pairs containing verbs and adjectives that are badly covered in the WCG.[9] A number of word pairs (mostly containing combinations of verbs or adjectives) were found neither in GermaNet nor Wikipedia (see GN $\cup$ WCG). If we consider only noun-noun pairs (NN), the coverage of Wikipedia exceeds that of GermaNet. The high proportion of word pairs that are either only found in GermaNet or in the WCG indicates that they are partially complementary with respect to covered vocabulary.

[9]Resulting from an editorial decision, Wikipedia only contains articles devoted to terms of encyclopedic interest - mainly nouns. Adjectives and verbs redirect to their corresponding nouns, if they are covered at all.

5 Conclusion

In this paper, we performed a graph-theoretic analysis of the Wikipedia Category Graph and showed that it is a scale-free, small-world graph, like other semantic networks such as WordNet or Roget's thesaurus. From this result, we concluded that the WCG can be used for NLP tasks, where other semantic networks have been traditionally employed. As Wikipedia is a multi-lingual resource, this enables the transfer of NLP algorithms to languages that do not have well-developed semantic wordnets.

To substantiate this claim, we described how measures of semantic relatedness operating on semantic wordnets, like WordNet or GermaNet, can be adapted to work on the WCG. We showed that the WCG is well suited to estimate SR between words. This is due to the categorization process that connects terms which would not be closely related in a taxonomic wordnet structure. Consequently, GermaNet outperforms the WCG on the task of estimating semantic similarity. Furthermore, the WCG cannot be used for tasks that require knowledge about the exact type of semantic relation.

We performed an analysis of the coverage of Wikipedia. It covers nouns very well, but is less suited to compute semantic relatedness across parts-of-speech. In this case, conventional semantic wordnets are likely to provide a better knowledge source. In Zesch et al. (2007b), we show that knowledge from wordnets and from Wikipedia is complementary, and can be combined to improve the performance on the SR task. As the simple PL measure performs remarkably well on the SR datasets, in our future work, we will also consider computing SR using the path length on the Wikipedia article graph rather than on the WCG.

Acknowledgments

This work was supported by the German Research Foundation under the grant "Semantic Information

DATASET	# PAIRS	GN	WCG	GN ∪ WCG	GN \ WCG	WCG \ GN	GN ∩ WCG
Gur65	65	57	61	65	4	8	53
Gur350	350	208	161	248	87	40	121
Gur350 NN	173	109	115	129	14	20	95
ZG222	222	86	86	118	32	30	56
ZG222 NN	119	57	61	73	12	16	45

Table 3: Number of covered word pairs based on GermaNet (GN) and the WCG on different datasets.

Retrieval from Texts in the Example Domain Electronic Career Guidance" (SIR), GU 798/1-2.

References

A. Barabasi and R. Albert. 1999. Emergence of scaling in random networks. *Science*, 286:509–512.

A. Budanitsky and G. Hirst. 2006. Evaluating WordNet-based Measures of Semantic Distance. *Computational Linguistics*, 32(1).

L. Buriol, C. Castillo, D. Donato, S. Leonardi, and S. Millozzi. 2006. Temporal Analysis of the Wikigraph. In *Proc. of Web Intelligence*, Hong Kong.

A. Capocci, V. D. P. Servedio, F. Colaiori, L. S. Buriol, D. Donato, S. Leonardi, and G. Caldarelli. 2006. Preferential attachment in the growth of social networks: The internet encyclopedia Wikipedia. *Physical Review E*, 74:036116.

C. Fellbaum. 1998. *WordNet An Electronic Lexical Database*. MIT Press, Cambridge, MA.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, and G. Wolfman. 2002. Placing Search in Context: The Concept Revisited. *ACM TOIS*, 20(1):116–131.

I. Gurevych. 2005. Using the Structure of a Conceptual Network in Computing Semantic Relatedness. In *Proc. of IJCNLP*, pages 767–778.

T. Holloway, M. Bozicevic, and K. Börner. 2005. Analyzing and Visualizing the Semantic Coverage of Wikipedia and Its Authors. *ArXiv Computer Science e-prints*, cs/0512085.

M. Jarmasz and S. Szpakowicz. 2003. Roget's thesaurus and semantic similarity. In *Proc. of RANLP*, pages 111–120.

J. J. Jiang and D. W. Conrath. 1997. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *Proc. of the 10th International Conference on Research in Computational Linguistics*.

C. Kunze, 2004. *Computerlinguistik und Sprachtechnologie*, chapter Lexikalisch-semantische Wortnetze, pages 423–431. Spektrum Akademischer Verlag.

C. Leacock and M. Chodorow, 1998. *WordNet: An Electronic Lexical Database*, chapter Combining Local Context and WordNet Similarity for Word Sense Identification, pages 265–283. Cambridge: MIT Press.

D. Lin. 1998. An Information-Theoretic Definition of Similarity. In *Proc. of ICML*.

G. A. Miller and W. G. Charles. 1991. Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes*, 6(1):1–28.

J. Morris and G. Hirst. 2004. Non-Classical Lexical Semantic Relations. In *Proc. of the Workshop on Computational Lexical Semantics, NAACL-HLT*.

D. L. Nelson, C. L. McEvoy, and T. A. Schreiber. 1998. The University of South Florida word association, rhyme, and word fragment norms. Technical report, U. of South Florida.

M. E. J. Newman. 2003. The structure and function of complex networks. *SIAM Review*, 45:167–256.

R. Rada, H. Mili, E. Bicknell, and M. Blettner. 1989. Development and Application of a Metric on Semantic Nets. *IEEE Trans. on Systems, Man, and Cybernetics,*, 19(1):17–30.

P. Resnik. 1995. Using Information Content to Evaluate Semantic Similarity. In *Proc. of IJCAI*, pages 448–453.

H. Rubenstein and J. B. Goodenough. 1965. Contextual Correlates of Synonymy. *Communications of the ACM*, 8(10):627–633.

N. Seco, T. Veale, and J. Hayes. 2004. An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In *Proc. of ECAI*.

M. Steyvers and J. B. Tenenbaum. 2005. The Large-Scale Structure of Semantic Networks: Statistical Analyses and a Model of Semantic Growth. *Cognitive Science*, 29:41–78.

J. Voss. 2005. Measuring Wikipedia. In *Proc. of the 10th International Conference of the International Society for Scientometrics and Informetrics*, Stockholm, Sweden.

J. Voss. 2006. Collaborative thesaurus tagging the Wikipedia way. *ArXiv Computer Science e-prints*, cs/0604036.

D. J. Watts and S. H. Strogatz. 1998. Collective Dynamics of Small-World Networks. *Nature*, 393:440–442.

Z. Wu and M. Palmer. 1994. Verb Semantics and Lexical Selection. In *Proc. of ACL*, pages 133–138.

T. Zesch and I. Gurevych. 2006. Automatically Creating Datasets for Measures of Semantic Relatedness. In *Proc. of the Workshop on Linguistic Distances, ACL*, pages 16–24.

T. Zesch, I. Gurevych, and M. Mühlhäuser. 2007a. Analyzing and Accessing Wikipedia as a Lexical Semantic Resource. In *Proc. of Biannual Conference of the Society for Computational Linguistics and Language Technology*, pages 213–221.

T. Zesch, I. Gurevych, and M. Mühlhäuser. 2007b. Comparing Wikipedia and German Wordnet by Evaluating Semantic Relatedness on Multiple Datasets. In *Proc. of NAACL-HLT*, page (to appear).

V. Zlatic, M. Bozicevic, H. Stefancic, and M. Domazet. 2006. Wikipedias: Collaborative web-based encyclopedias as complex networks. *Physical Review E*, 74:016115.

Multi-level Association Graphs – A New Graph-Based Model for Information Retrieval

Hans Friedrich Witschel
NLP department
University of Leipzig
04009 Leipzig, P.O. Box 100920
witschel@informatik.uni-leipzig.de

Abstract

This paper introduces *multi-level association graphs* (MLAGs), a new graph-based framework for information retrieval (IR). The goal of that framework is twofold: First, it is meant to be a meta model of IR, i.e. it subsumes various IR models under one common representation. Second, it allows to model different forms of search, such as feedback, associative retrieval and browsing at the same time. It is shown how the new integrated model gives insights and stimulates new ideas for IR algorithms. One of these new ideas is presented and evaluated, yielding promising experimental results.

1 Introduction

Developing formal models for information retrieval has a long history. A model of information retrieval "predicts and explains what a user will find relevant given the user query" (Hiemstra, 2001). Most IR models are firmly grounded in mathematics and thus provide a formalisation of ideas that facilitates discussion and makes sure that the ideas can be implemented. More specifically, most IR models provide a so-called *retrieval function* $f(q, d)$, which returns – for given representations of a document d and of a user information need q – a so-called retrieval status value by which documents can be ranked according to their presumed relevance w.r.t. to the query q.

In order to understand the commonalities and differences among IR models, this paper introduces the notion of *meta modeling*. Since the word "meta model" is perhaps not standard terminology in IR, it should be explained what is meant by it: a meta model is a model or framework that subsumes other IR models, such that they are derived by specifying certain parameters of the meta model.

In terms of IR theory, such a framework conveys what is common to all IR models by subsuming them. At the same time, the differences between models are highlighted in a conceptually simple way by the different values of parameters that have to be set in order to arrive at this subsumption. It will be shown that a graph-based representation of IR data is very well suited to this problem.

IR models concentrate on the *matching* process, i.e. on measuring the degree of overlap between a query q and a document representation d. On the other hand, there are the problems of finding suitable representations for documents (*indexing*) and for users' information needs (*query formulation*). Since users are often not able to adequately state their information need, some interactive and associative procedures have been developed by IR researchers that help to overcome this problem:

- *Associative retrieval*, i.e. retrieving information which is associated to objects known or suspected to be relevant to the user – e.g. query terms or documents that have been retrieved already.
- *Feedback*, another method for boosting recall, either relies on relevance information given by the user (relevance feedback) or assumes

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 9–16,
Rochester, April 2007

top-ranked documents to be relevant (pseudo feedback) and learns better query formulations from this information.

- *Browsing*, i.e. exploring a document collection interactively by following links between objects such as documents, terms or concepts.

Again, it will be shown that – using a graph-based representation – these forms of search can be subsumed easily.

2 Related work

2.1 Meta modeling

In the literal sense of the definition above, there is a rather limited number of meta models for IR, the most important of which will be described here very shortly.

Most research about how to subsume various IR models in a common framework has been done in the context of Bayesian networks and *probabilistic inference* (Turtle and Croft, 1990). In this approach, models are subsumed by specifying certain probability distributions. In (Wong and Yao, 1995), the authors elaborately show how all major IR models known at that time can be subsumed using probabilistic inference. Language modeling, which was *not* known then was later added to the list by (Metzler and Croft, 2004).

Another graph-based meta modeling approach uses the paradigm of *spreading activation* (SA) as a simple unifying framework. Given semantic knowledge in the form of a (directed) graph, the idea of spreading activation is that a measure of relevance – w.r.t. a current focus of attention – is spread over the graph's edges in the form of *activation energy*, yielding for each vertex in the graph a degree of relatedness with that focus (cf. (Anderson and Pirolli, 1984)). It is easy to see how this relates to IR: using a graph that contains vertices for both terms and documents and appropriate links between the two, we can interpret a query as a focus of attention and spread that over the network in order to rank documents by their degree of relatedness to that focus.

A very general introduction of spreading activation as a meta model is given in the early work by (Preece, 1981) All later models are hence special cases of Preece's work, including the multi-level association graphs introduced in section 3. Preece's model subsumes the Boolean retrieval model, coordination level matching and vector space processing.

Finally, an interesting meta model is described by (van Rijsbergen, 2004) who uses a Hilbert space as an information space and connects the geometry of that space to probability and logics. In particular, he manages to give the familiar dot product between query and document vector a probabilistic interpretation.

2.2 Graph-based models for associative retrieval and browsing

The spreading activation paradigm is also often used for associative retrieval. The idea is to reach vertices in the graph that are not necessarily directly linked to query nodes, but are reachable from query nodes via a large number of short paths along highly weighted edges.

Besides (Preece, 1981), much more work on SA was done, a good survey of which can be found in (Crestani, 1997). A renewed interest in SA was later triggered with the advent of the WWW where hyperlinks form a directed graph. In particular, variants of the PageRank (Brin and Page, 1998) algorithm that bias a random searcher towards some starting nodes (e.g. an initial result set of documents) bear close resemblance to SA (Richardson and Domingos, 2002; White and Smyth, 2003).

Turning to browsing, we can distinguish three types of browsing w.r.t. to the vertices of the graph: index term browsing, which supports the user in formulating his query by picking related terms (Doyle, 1961; Beaulieu, 1997), document browsing which serves to expand result sets by allowing access to similar documents or by supporting web browsing (Smucker and Allan, 2006; Olston and Chi, 2003) and combined approaches where both index terms and documents are used simultaneously for browsing.

In this last category, many different possibilities arise for designing interfaces. A common guiding principle of many graph-based browsing approahces is that of interactive spreading activation (Oddy, 1977; Croft and Thompson, 1987). Another approach, which is very closely related to MLAGs, is a multi-level hypertext (MLHT), as proposed in

(Agosti and Crestani, 1993) – a data structure consisting of three levels, for documents, index terms and concepts. Each level contains objects and links among them. There are also connections between objects of two adjacent levels. An MLHT is meant to be used for interactive query formulation, browsing and search, although (Agosti and Crestani, 1993) give no precise specification of the processing procedures.

2.3 Contribution of this work

Compared to Preece's work, the MLAG framework makes two sorts of modifications in order to reach the goals formulated in the introduction: in order to subsume more IR models, the flexibility and power of Preece's model is increased by adding real-valued edge weights. On the other hand, a clearer distinction is made between local and global information through the explicit introduction of "level graphs".

With the introduction of *levels*, the MLAG data structure becomes very closely related to the MLHT paradigm of (Agosti and Crestani, 1993), MLAGs, however, generalise MLHTs by allowing arbitrary types of levels, not only the three types proposed in (Agosti and Crestani, 1993). Additionally, links in MLAGs are weighted and the spreading activation processing defined in the next section makes extensive use of these weights.

All in all, the new model combines the data structure of multi-level hypertexts (Agosti and Crestani, 1993) with the processing paradigm of spreading activation as proposed by Preece (Preece, 1981), refining both with an adequate edge weighting. The framework is an attempt to be *as general as necessary* for subsuming all models and allowing for different forms of search, while at the same time being *as specific as possible* about the things that are really common to all IR models.

3 The MLAG model

3.1 Data structure

Formally, the basis of a multi-level association graph (MLAG) is a union of n level graphs $L_1, ..., L_n$. Each of these n directed graphs $L_i = G(VL_i, EL_i, WL_i)$ consists of a set of vertices VL_i, a set $EL_i \subseteq VL_i \times VL_i$ of edges and a function $WL_i : EL_i \to \mathbb{R}$ returning edge weights.

In order to connect the levels, there are $n - 1$ connecting bipartite graphs (or inverted lists) $I_{1,2}, ..., I_{n-1,n}$ where each inverted list $I_{j,j+1}$ consists of vertices $VI_{j,j+1} = VL_j \cup VL_{j+1}$, edges $EI_{j,j+1} \subseteq (VL_j \times VL_{j+1}) \cup (VL_{j+1} \times VL_j)$ and weights $WI_{j,j+1} : EI_{j,j+1} \to \mathbb{R}$. Figure 1 depicts a simple example multi-level association graph with two levels L_d and L_t for documents and terms.

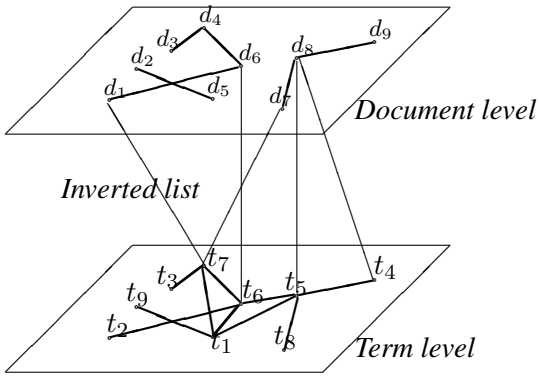


Figure 1: A simple example MLAG

Assuming that the vertices on a given level correspond to objects of the same type and vertices in different levels to objects of different types, this data structure has the following general interpretation: Each level represents associations between objects of a given type, e.g. term-term or document-document similarities. The inverted lists, on the other hand, represent associations between different types of objects, e.g. occurrences of terms in documents.

3.2 Examples

3.2.1 Standard retrieval

The simplest version of a multi-level association graph consists of just two levels – a term level L_t and a document level L_d. This is the variant depicted in figure 1.

The graph I_{td} that connects L_t and L_d is an inverted list in the traditional sense of the word, i.e. a term is connected to all documents that it occurs in and the weight $WI(t, d)$ of an edge (t, d) connecting term t and document d conveys the degree to which t is representative of d's content, or to which d is about t.

The level graphs L_t and L_d can be computed in various ways. As for documents, a straight-forward way would be to calculate document similarities, e.g. based on the number of terms shared by two documents. However, other forms of edges are possible, such as hyperlinks or citations – if available. Term associations, on the other hand, can be computed using co-occurrence information. An alternative would be to use relations from manually created thesauri or ontologies.

3.2.2 More levels

In order to (partly) take document structure into account, it can be useful to introduce a level for document parts (e.g. headlines and/or passages) in between the term and the document level. This can be combined with text summarisation methods (cf. e.g. (Brandow et al., 1995)) in order to give higher weights to more important passages in the inverted list connecting passages to documents.

In distributed or peer-to-peer environments, databases or peers may be modeled in a separate layer above the document level, inverted lists indicating where documents are held. Additionally, a peer's neighbours in an overlay network may be modeled by directed edges in the peer level graph. More extensions are possible and the flexibility of the MLAG framework allows for the insertion of arbitrary layers.

3.3 Processing paradigm

The operating mode of an MLAG is based on the spreading activation principle. However, the spread of activation between nodes of two *different* levels is not iterated. Rather, it is carefully controlled, yet allowing non-linear modifications at some points.

In order to model spreading activation in an MLAG, we introduce an *activation function* $A_i : VL_i \rightarrow \mathbb{R}$ which returns the so-called *activation energies* of vertices on a given level L_i. The default value of the activation function is $A_i(v) = 0$ for all vertices $v \in VL_i$.

In the following, it is assumed that the MLAG processing is invoked by *activating* a set of vertices A on a given level L_i of the MLAG by modifying the activation function of that level so that $A_i(v) = w_v$ for each $v \in A$.

A common example of such activation is a query being issued by a user. The initial activation is the result of the query formulation process, which selects some vertices $v \in A$ and weights them according to their presumed importance w_v. This weight is then the initial activation energy of the vertex.

Once we have an initial set of activated vertices, the following general procedure is executed until some stopping criterion is met:

1. Collect activation values on current level L_i, i.e. determine $A_i(u)$ for all $u \in VL_i$.
2. (Optionally) apply a transformation to the activation energies of L_i-nodes, i.e. alter $A_i(u)$ by using a – possibly non-linear – transformation function or procedure.
3. Spread activation to the next level L_{i+1} along the links connecting the two levels:

$$A_{i+1}(v) = \sum_{(u,v) \in I_{i,i+1}} A_i(u) \cdot WI(u,v) \quad (1)$$

4. Set $A_i(u) = 0$ for all $u \in VL_i$, i.e. "forget" about the old activation energies
5. (Optionally) apply a transformation to the activation energies of L_{i+1}-nodes (see step 2).
6. Go to 1, increment i (or decrement, depending on its value and the configuration of the MLAG)

If we take a vector space view of this processing mode and if we identify level L_i with terms and level L_{i+1} with documents, we can interpret the activation energies $A_i(u)$ as a query vector and the edge weights $WI(u,v)$ of edges arriving at vertex $v \in VL_{i+1}$ as a document vector for document v.

This shows that the basic retrieval function realised by steps 1, 3 and 4 of this process is a simple dot product. We will later see that retrieval functions of most IR models can actually be written in that form, provided that the initial activation of query terms and the edge weights of $I_{i,i+1}$ are chosen correctly (section 4).

For some models, however, we additionally need the possibility to perform nonlinear transformations on result sets in order to subsume them. Steps 2 and 5 of the algorithm allow for arbitrary modifications

of the activation values based on whatever evidence may be available on the current level or globally – but *not* in the inverted list. This will later also allow to include feedback and associative retrieval techniques.

4 The MLAG as a meta model

In this section, examples will be shown that demonstrate how existing IR models of ranked retrieval [1] can be subsumed using the simple MLAG of figure 1 and the processing paradigm from the last section. This is done by specifying the following parameters of that paradigm:

1. How nodes are activated in the very first step
2. How edges of the inverted list are weighted
3. Which transformation is used in 2 and 5.

For each model, the corresponding retrieval function will be given and the parameter specification will be discussed shortly. The specification of the above parameters will be given in the form of triplets $\langle activation_{init}, edge\ weights, transform \rangle$.

4.1 Vector space model

In the case of the vector space model (Salton et al., 1975), the retrieval function to be mimicked is as follows:

$$f(q,d) = \sum_{t \in q \cap d} w_{tq} w_{td} \tag{2}$$

where w_{tq} and w_{td} are a term's weight in the query q and the current document d, respectively. This can be achieved by specifying the parameter triplet $\langle w_{tq}, w_{td}, none \rangle$. This simple representation reflects the closeness of the MLAG paradigm to the vector space model that has been hinted at above.

4.2 Probabilistic model

For the probabilistic relevance model (Robertson and Sparck-Jones, 1976), the MLAG has to realise the following retrieval function

$$f(q,d) = \sum_i d_i \log \frac{p_i(1-r_i)}{r_i(1-p_i)} \tag{3}$$

[1] This excludes the Boolean model, which can, however, also be subsumed as shown in section 5.5 of (Preece, 1981)

where $d_i \in \{0,1\}$ indicates whether term i is contained in document d, p_i is the probability that a relevant document will contain term i and r_i is the probability that an irrelevant document will contain it. This retrieval function is realised by the parameter triplet $\langle \log \frac{p_i(1-r_i)}{r_i(1-p_i)}, d_i, none \rangle$.

Now there is still the question of how the estimates of p_i and r_i are derived. This task involves the use of relevance information which can be gained via feedback, described in section 6.1.

4.3 Language models

The general language modeling retrieval function (cf. e.g. (Zhai and Lafferty, 2001)) is – admittedly – not in the linear form of equation 1. But using logarithms, products can be turned into sums without changing the ranking – the logarithm being a monotonic function (note that this is what also happened in the case of the probabilistic relevance models).

In particular, we will use the approach of comparing query and document language models by Kullback-Leibler divergence (KLD) (Lafferty and Zhai, 2001) which results in the equation

$$\begin{aligned} KLD(M_q||M_d) &= \sum_{t \in q} P(t|M_q) \log \frac{P(t|M_q)}{P(t|M_d)} \\ &\propto -\sum_{t \in q} P(t|M_q) \log P(t|M_d) \end{aligned}$$

where $P(t|M_q)$ and $P(t|M_d)$ refer to the probability that term t will be generated by the unigram language model of query q or document d, respectively. Note that we have simplified the equation by dropping a term $\sum_t P(t|M_q) \log P(t|M_q)$, which depends only on the query, not on the documents to be ranked.

Now, the triplet $\langle P(t|M_q), -\log P(t|M_d), t \rangle$ can be used to realise this retrieval function where t stands for a procedure that adds $-P(t|M_q) \log P(t|M_d)$ to the document node's activation level for terms t not occurring in d and sorts documents by *increasing* activation values afterwards.

5 Combining IR models

As can be seen from the last equation above, the language model retrieval function sums over all terms in the query. Each term – regardless of whether it

appears in the document d or not – contributes something that may be interpreted as a "penalty" for the document. The magnitude of this penalty depends on the smoothing method used (cf. (Zhai and Lafferty, 2001)). A popular smoothing method uses so-called Dirichlet priors to estimate document language models:

$$P(t|M_d) = \frac{tf + \mu p(t|C)}{\mu + |d|} \quad (4)$$

where tf is t's frequency in d, $p(t|C)$ is the term's relative frequency in the whole collection and μ is a free parameter. This indicates that if a rare term is missing from a document, the penalty will be large, $P(t|M_d)$ being very small because $tf = 0$ and $p(t|C)$ small.

Conceptually, it is unproblematic to model the retrieval function by making I_{td} a complete bipartite graph, i.e. specifying a (non-zero) value for $P(t|M_d)$, even if t does not occur in d. In a practical implementation, this is not feasible, which is why we add the contribution of terms not contained in a document, i.e. $-P(t|M_q) \log P(t|M_d)$, for terms that do not occur in d. [2]

This transformation indicates an important difference between language modeling and all other IR models: language models *penalise* documents for the *absence* of rare (i.e. informative) terms whereas the other models *reward* them for the *presence* of these terms.

These considerations suggest a combination of both approaches: starting with an arbitrary "presence rewarding" model – e.g. the vector space model – we may integrate the "absence penalising" philosophy by subtracting from a document's score, for each missing term, the contribution that one occurrence of that term would have earned (cf. (Witschel, 2006)).

For the vector space model, this yields the following retrieval function:

$$\begin{aligned} f(q,d) &= \sum_{t \in q \cap d} w_{tq} w_{td} \\ &- \frac{\alpha}{|q|} \sum_{t \in q \setminus d} w_{td}(tf = 1) w_{tq} \end{aligned}$$

where α is a free parameter regulating the relative influence of penalties, comparable to the μ parameter of language models above.

5.1 Experimental results

Table 2 shows retrieval results for combining two weighting schemes, *BM25* (Robertson et al., 1992) and *Lnu.ltn* (Singhal et al., 1996), with penalties. Both of them belong to the family of $tf.idf$ weighting schemes and can hence be regarded as representing the vector space model, although BM25 was developed out of the probabilistic model.

Combining them with the idea of "absence penalties" works as indicated above, i.e. weights are accumulated for each document using the $tf.idf$-like retrieval functions. Then, from each score, the contributions that one occurrence of each missing term would have earned is subtracted. More precisely, what is subtracted consists of the usual $tf.idf$ weight for the missing term, where $tf = 1$ is substituted in the tf part of the formula.

Experiments were run with queries from TREC-7 and TREC-8. In order to study the effect of query length, very short queries (using only the title field of TREC queries), medium ones (using title and description fields) and long ones (using all fields) were used. Table 1 shows an example TREC query.

< top>
< num> Number: 441
< title> Lyme disease
< desc> Description:
How do you prevent and treat Lyme disease?
< narr> Narrative:
Documents that discuss current prevention and treatment techniques for Lyme disease are relevant [...]
< /top>

Table 1: A sample TREC query

Table 2 shows that both weighting schemes can be significantly improved by using penalties, especially for short queries, reaching and sometimes surpassing the performance of retrieval with language models. This holds even when the parameter α is not tuned and confirms that interesting insights are gained from a common representation of IR models in a graph-based environment. [3]

[2] In order to do this, we only need to know $|d|$ and the relative frequency of t in the collection $p(t|C)$, i.e. information that is available *outside* the inverted list.

[3] Note that these figures were obtained without any refinements such as query expansion and are hence substantially

	TREC-7			TREC-8		
Weighting	very short	medium	long	very short	medium	long
BM25	0.1770	0.2120	0.2141	0.2268	0.2514	0.2332
+ P ($\alpha = 1$)	0.1867*	0.2194*	0.2178*	0.2380*	0.2593*	0.2335
+ P (best α)	**0.1896***	**0.2220***	**0.2185***	0.2411*	**0.2625***	**0.2337**
best α value	2	2	1.5	2	2	0.25
Lnu.ltn	0.1521	0.1837	0.1920	0.1984	0.2226	0.2013
+ P ($\alpha = 1$)	0.1714*	0.1972*	0.1946*	0.2176*	0.2305*	0.2040
+ P (best α)	0.1873*	0.2106*	0.1977	0.2394*	0.2396*	0.2064*
best α value	5	5	3	5	4	1.5
LM	0.1856	0.2163	0.2016	**0.2505**	0.2578	0.2307

Table 2: Mean average precision of BM25 and Lnu.ltn and their corresponding penalty schemes (+ P) for TREC-7 and TREC-8. Asterisks indicate statistically significant deviations (using a paired Wilcoxon test on a 95% confidence level) from each baseline, whereas the best run for each query length is marked with **bold** font. Performance of language models (LM) is given for reference, where the value of the smoothing parameter μ was set to the average document length.

6 Different forms of search with MLAGs

In order to complete the goals stated in the introduction of this paper, this section will briefly explain how feedback, associative retrieval and browsing can be modeled within the MLAG framework.

6.1 Feedback

Using the simple term-document MLAG of figure 1, feedback can be implemented by the following procedure:

1. Perform steps 1 – 4 of the basic processing.
2. Apply a transformation to the activation values of L_d-nodes, e.g. let the user pick relevant documents and set their activation to some positive constant β.
3. Perform step 3 of the basic processing with $L_i = L_d$ and $L_{i+1} = L_t$, i.e. let activation flow back to term level.
4. Forget about activation levels of documents.
5. Apply transformation on the term level L_t, e.g. apply thresholding to obtain a fixed number of expansion terms.
6. Spread activation back to the document level to obtain the final retrieval status values of documents.

lower than MAP scores achieved by systems actually participating in TREC.

In order to instantiate a particular feedback algorithm, there are three parameters to be specified:

- The transformation to be applied in step 2
- The weighting of document-term edges (if different from term-document edges) and
- The transformation applied in step 5.

Unfortunately, due to space constraints, it is out of the scope of this paper to show how different specifications lead to well-known feedback algorithms such as Rocchio (Rocchio, 1971) or the probabilistic model above.

6.2 Associative retrieval

Associative retrieval in MLAGs exploits the information encoded in level graphs: expanding queries with related terms can be realised by using the term level graph L_t of a simple MLAG (cf. figure 1) in step 2 of the basic processing, whereas the expansion of document result sets takes place in step 5 on the document level L_d. In order to exploit the relations encoded in the level graphs, one may again use spreading activation, but also simpler mechanisms. Since relations are used directly, dimensionality reduction techniques such as LSI cannot and need not be modeled.

6.3 Browsing

Since the MLAG framework is graph-based, it is easy to grasp and to be visualised, which makes it

a suitable data structure for browsing. The level graphs can be used as a flat graphical representation of the data, which can be exploited directly for browsing. Depending on their information need, users can choose to browse either on the term level L_t or on the document level L_d and they can switch between both types of levels at any time using the inverted list I_{td}. This applies, of course, also to passage or any other type of levels if they exist.

7 Conclusions

In this paper, a new graph-based framework for information retrieval has been introduced that allows to subsume a wide range of IR models and algorithms. It has been shown how this common representation can be an inspiration and lead to new insights and algorithms that outperform the original ones. Future work will aim at finding similar forms of synergies for the different forms of search, e.g. new combinations of feedback and associative retrieval algorithms.

References

M. Agosti and F. Crestani. 1993. A methodology for the automatic construction of a hypertext for information retrieval. In *Proceedings of SAC 1993*, pages 745–753.

J. R. Anderson and P. L. Pirolli. 1984. Spread of activation. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 10:791–799.

M. Beaulieu. 1997. Experiments of interfaces to support query expansion. *Journal of Documentation*, 1(53):8–19.

R. Brandow, K. Mitze, and L. F. Rau. 1995. Automatic condensation of electronic publications by sentence selection. *Information Processing and Management*, 31(5):675–685.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of WWW7*, pages 107–117.

F. Crestani. 1997. Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 11(6):453–482.

W. B. Croft and R. H. Thompson. 1987. I3R : a new approach to the design of document retrieval systems. *Journal of the american society for information science*, 38(6):389–404.

L. B. Doyle. 1961. Semantic Road Maps for Literature Searchers. *Journal of the ACM*, 8(4):553–578.

D. Hiemstra. 2001. *Using language models for information retrieval*. Ph.D. thesis, University of Twente.

J. Lafferty and C. Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR 2001*, pages 111–119.

D. Metzler and W. B. Croft. 2004. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40(5):735–750.

R. N. Oddy. 1977. Information retrieval through man-machine dialogue. *Journal of Documentation*, 33(1):1–14.

C. Olston and E. H. Chi. 2003. ScentTrails: Integrating browsing and searching on the Web. *ACM Transactions on Computer-Human Interaction*, 10(3):177–197.

S. E. Preece. 1981. *A spreading activation network model for information retrieval*. Ph.D. thesis, Universtiy of Illinois at Urbana-Champaign.

M. Richardson and P. Domingos. 2002. The intelligent surfer: Probabilistic combination of link and content information in pagerank. In *Proceedings of Advances in Neural Information Processing Systems*.

S. E. Robertson and K. Sparck-Jones. 1976. Relevance Weighting of Search Terms. *JASIS*, 27(3):129–146.

S. E. Robertson, S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. 1992. Okapi at TREC-3. In *Proceedings of TREC*, pages 21–30.

J.J. Rocchio. 1971. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System : Experiments in Automatic Document Processing*. Prentice Hall Inc., Englewood Cliffs, New Jersey.

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

A. Singhal, C. Buckley, and M. Mitra. 1996. Pivoted document length normalization. In *Proceedings of SIGIR 1996*, pages 21–29.

M. D. Smucker and J. Allan. 2006. Find-similar: similarity browsing as a search tool. In *Proceedings of SIGIR 2006*, pages 461–468.

H. Turtle and W. B. Croft. 1990. Inference networks for document retrieval. In *Proceedings of SIGIR 1990*, pages 1–24.

C. J. van Rijsbergen. 2004. *The Geometry of Information Retrieval*. Cambridge University Press.

Scott White and Padhraic Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proceedings of KDD 2003*, pages 266–275.

H. F. Witschel. 2006. Carrot and stick: combining information retrieval models. In *Proceedings of DocEng 2006*, page 32.

S. K. M. Wong and Y. Y. Yao. 1995. On modeling information retrieval with probabilistic inference. *ACM Transactions on Information Systems*, 13(1):38–68.

C. Zhai and J. Lafferty. 2001. A study of smoothing methods for language models applied to Ad Hoc information retrieval. In *Proceedings of SIGIR 2001*, pages 334–342.

Extractive Automatic Summarization: Does more linguistic knowledge make a difference?

Daniel S. Leite[1], Lucia H. M. Rino[1], Thiago A. S. Pardo[2], Maria das Graças V. Nunes[2]
Núcleo Interinstitucional de Lingüística Computacional (NILC)
http://www.nilc.icmc.usp.br
[1]Departamento de Computação, UFSCar
CP 676, 13565-905 São Carlos - SP, Brazil
[2]Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo
CP 668, 13560-970 São Carlos - SP, Brazil
{daniel_leite; lucia}@dc.ufscar.br , {taspardo,gracan}@icmc.usp.br

Abstract

In this article we address the usefulness of linguistic-independent methods in extractive Automatic Summarization, arguing that linguistic knowledge is not only useful, but may be necessary to improve the informativeness of automatic extracts. An assessment of four diverse AS methods on Brazilian Portuguese texts is presented to support our claim. One of them is Mihalcea's TextRank; other two are modified versions of the former through the inclusion of varied linguistic features. Finally, the fourth method employs machine learning techniques, tackling more profound and language-dependent knowledge.

1 Introduction

Usually, automatic summarization involves producing a condensed version of a source text through selecting or generalizing its relevant content. As a result, either an extract or an abstract will be produced. An extract is produced by copying text segments and pasting them into the final text preserving the original order. An abstract instead is produced by selecting and restructuring information from the source text. The resulting structure is thus linguistically realized independently of the surface choices of the source text. This comprises, thus, a rewriting task.

This article focuses solely on extracts of source texts written in Brazilian Portuguese. For extractive Automatic Summarization (AS), several methods have been suggested that are based upon statistics or data readily available in the source text. Word frequency (Luhn, 1958) and sentence position (Edmundson, 1969) methods are classic examples of that. Usually, extractive AS does not take into account linguistic and semantic knowledge in order to be portable to distinct domains or languages (Mihalcea, 2005). Graph-based methods aim at the same and have been gaining a lot of interest because they usually do not rely on any linguistic resource and run pretty fast. Exemplars of those are LexRank (Erkan and Radev, 2004) and TextRank (Mihalcea and Tarau, 2004). In spite of their potentialities, we claim that there is a compromise in pursuing a language-free setting: however portable a system may be, it may also produce extracts that lack the degree of informativeness needed for use. Informativeness, in the current context, refers to the ability of an automatic summarizer to produce summaries that convey most information of reference, or ideal, summaries. Our assessment thus aimed at verifying if parsimonious use of linguistic knowledge could improve extractive AS.

We argue that the lack of linguistic knowledge in extractive AS can be the reason for weak performance regarding informativeness. This argument follows from acknowledging that improvements on the scores usually obtained in that field have not been expressive lately. The most common metrics used to date, precision and recall, signal average results, suggesting that it is not enough to pursue completely language-free systems, no matter the current demands for portability in the global communication scenario. We focus here on TextRank, which can be used for summa-

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 17–24,
Rochester, April 2007

rizing Brazilian Portuguese texts due to its language independence. To show that linguistic knowledge does make a difference in extractive AS, we compared four automatic summarizers: TextRank itself, two other modified versions of that, and SuPor-2 (Leite and Rino, 2006). TextRank works in a completely unsupervised way. Our two variations, although still unsupervised, include diverse linguistic knowledge in the preprocessing phase. SuPor-2 is the only machine learning-based system amongst the four ones, and it was built to summarize texts in Brazilian Portuguese, although it may be customized to other languages. Unlike the others, it embeds more sophisticated decision features that rely on varied linguistic resources. Some of them correspond to full summarization methods by themselves: *Lexical Chaining* (Barzilay and Elhadad, 1997), *Relationship Mapping* (Salton et al., 1997), and *Importance of Topics* (Larocca Neto et al., 2000). This is its unique and distinguishing characteristic.

In what follows we first review the different levels of processing in extractive AS (Section 2), then we describe TextRank and its implementation to summarize Brazilian Portuguese texts (Section 3). Our suggested modifications of TextRank are presented in Section 4, whilst SuPor-2 is described in Section 5. Finally, we compare the results of the four automatic summarizers when running on Brazilian Portuguese texts (Section 6), and make some remarks on linguistic independence for extractive AS in Section 7.

2 A Review of Automatic Summarization

Mani (2001) classifies AS methods based upon three levels of linguistic processing to summarize a text, namely:

- **Shallow level**. At this level only features at the surface of the text are explored. For example, location (Edmunson, 1969), sentence length and presence of signaling phrases (e.g., Kupiec et al., 1995). Combined, such features may yield a salience function that drives selection of sentences of the source text to include in a summary.
- **Entity level**. The aim here is to build an internal representation of the source text that conveys its entities and corresponding relationships. These amount to the information that allows identifying important text segments. Examples of such relations are word cooccurrence (e.g., Salton et al., 1997), synonyms and antonyms (e.g., Barzilay and Elhadad, 1997), logical relations, such as concordance or contradiction, and syntactic relations.
- **Discourse level**. At this level the whole structure of the source text is modeled, provided that its communicative goals can be grasped from the source text. The discourse structure is intended to help retrieving, e.g., the main topics of the document (e.g, Barzilay and Elhadad, 1997; Larocca Neto et al., 2000) or its rhetorical structure (e.g., Marcu, 1999), in order to provide the means for AS.

In this work we mainly focus on the entity level. Special entities and their relations thus provide the means to identify important sentences for building an extract. In turn, there is a loss of independence from linguistic knowledge, when compared to shallower approaches. Actually, apart from TextRank, the other systems described in this paper target entity level methods, as we shall see shortly.

3 The TextRank Method

The unsupervised TextRank method (Mihalcea and Tarau, 2004) takes after Google's PageRank (Brin and Page, 1998), a graph-based system that helps judge the relevance of a webpage through incoming and outgoing links. PageRank directed graphs represent webpages as nodes and their linking to other webpages as edges. A random walk model is thus applied to build a path between the nodes, in order to grade the importance of a webpage in the graph.

Similarly to grading webpages through traversing a graph, TextRank attempts to weight sentences of a text by building an undirected graph. Nodes are now sentences, and edges express their similarity degrees to other sentences in the text. Actually, the degree of similarity is based upon content overlap. As such, similarity degrees help assess the overall cohesive structure of a text. The more content overlap a sentence has with other sentences, the more important it is and more likely it is to be included in the extract.. Similarity is calculated through equation [1] (Mihalcea and Tarau, 2004), where S_i and S_j are sentences and w_k is a common token between

them. The numerator is the sum of common words between S_i and S_j. To reduce bias, normalization of the involved sentences length takes place, as shows the denominator.

$$Sim(S_i, S_j) = \frac{|\{w_k \mid w_k \in S_i \wedge w_k \in S_j\}|}{\log(|S_i|) + \log(|S_j|)} \quad [1]$$

Once the graph and all similarity degrees are produced, sentence importance is calculated by the random walk algorithm shown in equation [2]. $TR(V_i)$ signals sentence importance, d is an arbitrary parameter in the interval [0,1], and N is the number of sentences in the text. Parameter d integrates the probability of jumping from one vertex to another randomly chosen. Thus, it is responsible for random walking. This parameter is normally set to 0.85 (this value is also used in TextRank).

$$TR(V_i) = (1-d) + d \times \sum_{j=0}^{N-1} \left(TR(V_j) \times \frac{Sim(S_i, S_j)}{\sum_{k=0}^{N-1} Sim(S_j, S_k)} \right) \quad [2]$$

Initial TR similarity values are randomly set in the [0,1] interval. After successive calculations, those values converge to the targeted importance value. After calculating the importance of the vertices, the sentences are sorted in reverse order and the top ones are selected to compose the extract. As usual, the number of sentences of the extract is dependent upon a given compression rate.

Clearly, TextRank is not language dependent. For this reason Mihalcea (2005) could use it to evaluate AS on texts in Brazilian Portuguese, besides reporting results on texts in English. She also explored distinct means of representing a text without considering linguistic knowledge, emphasizing TextRank language and domain independence. She varies, e.g., the ways the graphs could be traversed using both directed and undirected graphs. Once a sentence is chosen to compose an extract, having undirected graphs makes possible, to look forward – from the sentence to its outgoing edges (i.e., focusing on the set of its following sentences in the text) – or to look backward, considering that sentence incoming edges and, thus, the set of its preceding sentences in the text.

Another variation proposed by Mihalcea is to replace the PageRank algorithm (Equation [2]) by HITS (Kleinberg, 1999). This works quite similarly to PageRank. However, instead of aggregating the scores for both incoming and outgoing links of a node in just one final score, it produces two independent scores. These are correspondingly named "authority" and "hub" scores.

4 Improving TextRank through variations on linguistic information

To improve the similarity scores between sentences in TextRank we fed it with more linguistic knowledge, yielding its two modified versions. The first variation focused just upon basic preprocessing; the second one, on the use of a thesaurus to calculate semantic similarity to promote AS decisions. However, we did not modify the main extractive algorithm of TextRank: we kept the graph undirected and used PageRank as the score determiner. Actually, we modified only the method of computing the edges weights.

4.1 Using Basic Preprocessing Methods

In applying Equation 1 for similarity scores, only exact matches between two words are allowed. Since in Brazilian Portuguese there are many morphological and inflexional endings for most words, this process becomes troublesome: important matches may be ignored. To overcome that, we used a stemmer for Brazilian Portuguese (Caldas Jr. et al., 2001) based upon Porter's algorithm (1980). We also removed stopwords from the source text, because they are not useful in determining similarity. The resulting version of TextRank is named hereafter 'TextRank+Stem+StopwordsRem'.

4.2 Using a Thesaurus

Our second TextRank variation involved plugging into the system a Brazilian Portuguese thesaurus (Dias-da-Silva et al., 2003). Our hypothesis here is that semantic similarity of the involved words is also important to improve the informativeness of the extracts under production. Thus, an extractive summarizer should consider not only word repetition in the source text, but also synonymy and antonymy.

Although plugging the thesaurus into the automatic summarizer did not imply changing its main method of calculating similarity, there were some obstacles to overcome concerning the following:

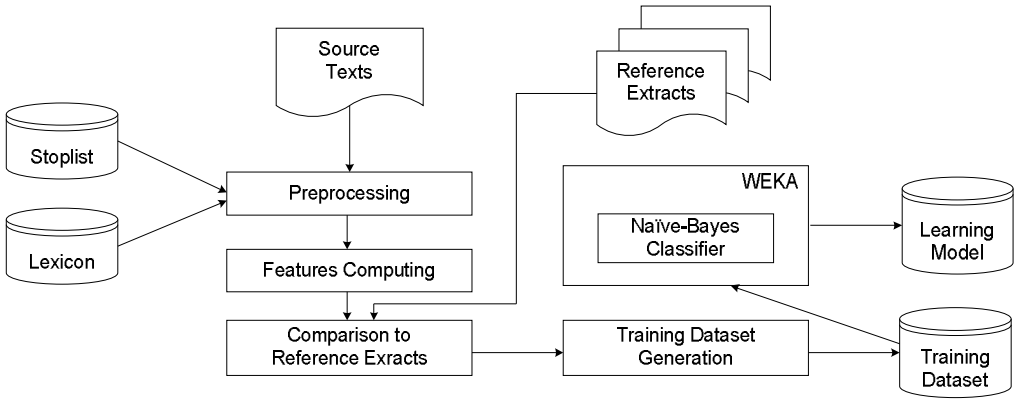


Figure 1. SuPor-2 training phase

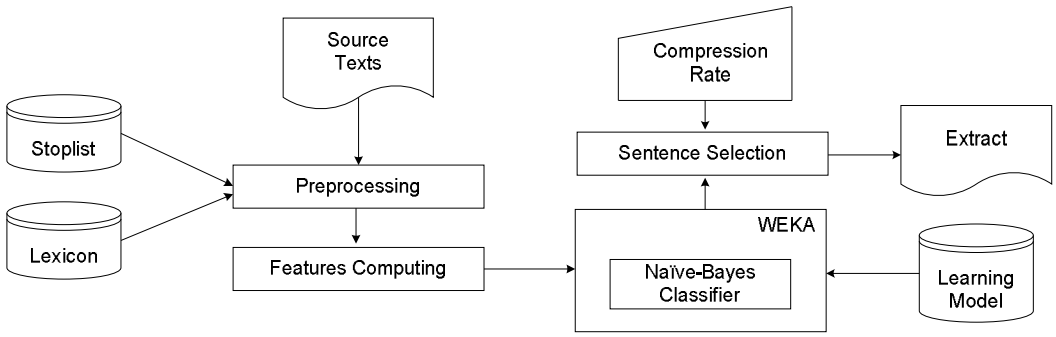


Figure 2. SuPor-2 extraction phase

a) Should we consider only synonyms or both synonyms and antonyms in addition to term repetition (reiteration)?

b) How to acknowledge, and disentangle, semantic similarity, when polissemy, for example, is present?

c) Once the proper relations have been determined, how should they be weighted? Just considering all thesaural relations to be equally important might not be the best approach.

Concerning (a), synonyms, antonyms, and term repetition were all considered, as suggested by others (e.g., Barzilay and Elhadad, 1997). We did not tackle (b) to choose the right sense of a word because of the lack of an effective disambiguation procedure for Brazilian Portuguese. Finally, in tackling (c) and, thus, grading the importance of the relations for sentence similarity, we adopted the same weights proposed by Barzilay and Elhadad (1997) in their lexical chaining method, which is discussed in more detail below. For both reiteration and synonymy, they assume a score of 10 for the considered lexical chain; for antonymy, they suggest a score of 7. The resulting version of TextRank is named here 'TextRank+Thesaurus'.

5 The SuPor-2 System

SuPor-2 is an extractive summarizer built from scratch for Brazilian Portuguese. It embeds different features in order to identify and extract relevant sentences of a source text. To configure SuPor-2 for an adequate combination of such features we employ a machine learning approach. Figures 1 and 2 depict the training and extraction phases, respectively.

For training, machine learning is carried out by a Naïve-Bayes classifier that employs Kernel methods for numeric feature handling, known as *Flexible Bayes* (John and Langley, 1995). This environment is provided by WEKA[1] (Witten and Frank, 2005), which is used within SuPor-2 itself. The training corpus comprises both source texts and corresponding reference extracts. Every sentence from a source text is represented in the train-

[1] Waikato Environment for Knowledge Analysis. Available at http://www.cs.waikato.ac.nz/ml/weka/ (December, 2006)

ing dataset as a tuple of the considered features. Each tuple is labeled with its class, which signals if the sentence appears in a reference extract. The class label will be true if the sentence under focus matches a sentence of the reference extract and false otherwise.

Once produced, the training dataset is used by the Bayesian classifier to depict the sentences that are candidates to compose the extract (Figure 2). In other words, the probability for the "true" class is computed and the top-ranked sentences are selected, until reaching the intended compression rate.

When computing features, three full methods (M) and four corpus-based parameters (P) are considered. Both methods and parameters are mapped onto the feature space and are defined as follows:

(M) Lexical Chaining (Barzilay and Elhadad, 1997). This method computes the connectedness between words aiming at determining lexical chains in the source text. The stronger a lexical chain, the more important it is considered for extraction. Both an ontological resource and WordNet (Miller et al., 1990) are used to identify different relations, such as synonymy or antonym, hypernymy or hyponymy, that intervene to compute connectedness. The lexical chains are then used to produce three sets of sentences. To identify and extract sentences from those sets, three heuristics are made available, namely: (H1) selecting every sentence *s* of the source text based on each member *m* of every strong lexical chain of the text. In this case, *s* is the sentence that contains the first occurrence of *m*; (H2) this heuristics is similar to the former one, but instead of considering all the members of a strong lexical chain, it uses only the representative ones. A representative member is one whose frequency is greater than the average frequency of all words in the chain; (H3) a sentence *s* is chosen by focusing only on representative lexical chains of every topic of the source text. In SuPor-2, the mapping of this method onto a nominal feature is accomplished by signaling which heuristics have recommended the sentence. Thus, features in the domain may range over the values {'None', 'H1', 'H2', 'H3', 'H1H2', 'H1H3', 'H2H3', 'H1H2H3'}.

(M) Relationship Mapping (Salton et al., 1997). This method performs similarly to the previous one and also to TextRank in that it builds up a graph interconnecting text segments. However, it considers paragraphs instead of sentences as vertices. Hence, graph edges signal the connectiveness of the paragraphs of the source text. Similarity scores between two paragraphs are thus related to the degree of connectivity of the nodes. Similarly to Lexical Chaining, Salton et al. also suggest three different ways of producing extracts. However, they now depend on the way the graph is traversed. The so-called dense or bushy path (P1), deep path (P2), and segmented path (P3) aim at tackling distinct textual problems that may damage the quality of the resulting extracts. The dense path considers that paragraphs are totally independent from each other, focusing on the top-ranked ones (i.e., the ones that are denser). As a result, it does not guarantee that an extract will be cohesive. The deep path is intended to overcome the former problem by choosing paragraphs that may be semantically inter-related. Its drawback is that only one topic, even one that is irrelevant, may be conveyed in the extract. Thus, it may lack proper coverage of the source text. Finally, the segmented path aims at overcoming the limitations of the former ones, addressing all the topics at once. Similarly to Lexical Chaining, features in the Relationship method range over the set {'None','P1','P2','P3', 'P1P2', 'P1P3', 'P2P3', 'P1P2P3'}.

(M) Importance of Topics (Larocca Neto et al., 2000). This method also aims at identifying the main topics of the source text, however through the TextTiling algorithm (Hearst, 1993). Once the topics of the source text have been determined, the first step is to select sentences that better express the importance of each topic. The amount of sentences, in this case, is proportional to the topic importance. The second step is to determine the sentences that will actually be included in the extract. This is carried out by measuring their similarity to their respective topic centroids (Larocca Neto et al., 2000). The method thus signals how relevant a sentence is to a given topic. In SuPor-2 this method yields a numeric feature whose value conveys the harmonic mean between the sentence similarity to the centroid of the topic in which it appears and the importance of that topic.

(P) Sentence Length (Kupiec et al., 1995). This parameter just signals the normalized count of words of a sentence.

(P) Sentence Location (Edmundson, 1969). This parameter takes into account the position of a sentence in the text. It is valued, thus, in {'II','IM','IF','MI','MM','MF','FI','FM','FF'}. In this set the first letter of each label signals the position of the sentence within a paragraph (Initial, Medium, or Final). Similarly, the second letter signals the position of the paragraph within the text.

(P) Occurrence of proper nouns (e.g., Kupiec et al., 1995). This parameter accounts for the number of proper nouns in a sentence.

(P) Word Frequency (Luhn, 1958). This parameter mirrors the normalized sum of the word frequency in a sentence.

SuPor-2 provides a flexible way of combining linguistic and non-linguistic features for extraction. There are profound differences from TextRank. First, it is clearly language-dependent. Also, its graph-based methods do not assign weights to their vertices in order to select sentences for extraction. Instead, they traverse a graph in very specific and varied ways that mirror both linguistic interdependencies and important connections between the nodes.

6 Assessing the Four Systems

To assess the degree of informativeness of the systems previously described, we adopt ROUGE[2] (Lin and Hovy, 2003), whose recall rate mirrors the informativeness degree of automatically generated extracts by correlating automatic summaries with ideal ones.

The two modified versions of TextRank require linguistic knowledge but at a low cost. This is certainly due to varying only preprocessing, while the main decision procedure is kept unchanged and language-independent. Those three systems do not need training, one of the main arguments in favor of TextRank (Mihalcea and Tarau, 2004). In contrast, SuPor-2 relies on training and this is certainly one of its main bottlenecks. It also employs linguistic knowledge for both preprocessing and extraction, which TextRank purposefully avoids. However, using WEKA has made its adjustments less demanding and more consistent, indicating that scaling up the system is feasible.

[2] Recall-Oriented Understudy for Gisting Evaluation. Available at http://haydn.isi.edu/ROUGE/ (January, 2007).

In our assessment, the same single-document summarization scenario posed by Mihalcea (2005) was adopted, namely: (a) we considered the Brazilian Portuguese TeMário corpus (Pardo and Rino, 2003); (b) we used the same baseline, which selects top-first sentences to include in the extract; (c) we adopted a 70-75% compression rate, making it compatible with the compression rate of the reference summaries; and (d) ROUGE was used for evaluation in its Ngram(1,1) 95% confidence rate setting, without stopwords removal. TeMário comprises 100 newspaper articles from online Brazilian newswire. A set of corresponding manual summaries produced by an expert in Brazilian Portuguese is also included in TeMário. These are our reference summaries.

For training and testing SuPor-2, we avoided building an additional training corpus by using a 10-fold cross-validation procedure. Finally, we produced three sets of extracts using 'TextRank + Stem + StopwordsRem', 'TextRank + Thesaurus', and SuPor-2 on the TeMário source texts. Results for informativeness are shown in Table 1. Since Mihalcea's setting was kept unchanged, we just included in that table the same results presented in (Mihalcea, 2005), i.e., we did not run her systems all over again. We also reproduced for comparison the TextRank variations reported by Mihalcea, especially regarding graph-based walks by PageRank and HITS. Shaded lines correspond to our suggested methods presented in Sections 4 and 5, which involve differing degrees of dependence on linguistic knowledge.

It can be seen that 'TextRank+Thesaurus' and 'TextRank+Stem+StopwordsRem' considerably outperformed all other versions of TextRank. Compared with Mihalcea's best version, i.e., with 'TextRank (PageRank - backward)', those two methods represented a 6% and 9% improvement, respectively. We can conclude that neither the way the graph is built nor the choice of the graph-based ranking algorithm affects the results as significantly as do the linguistic-based methods. Clearly, both variations proposed in this paper signal that linguistic knowledge, even if only used at the preprocessing stage, provides more informative extracts than those produced when no linguistic knowledge at all is considered. Moreover, at that stage little modeling and computational effort is demanded, since lexicons, stoplists, and thesauri

are quite widely available nowadays for several Romance languages.

Even the baseline outperformed most versions of TextRank, showing that linguistic independence in a random walk model for extractive AS should be reconsidered. Actually, this shows that linguistic knowledge *does make a difference*, at least for summarizing newswire texts in Brazilian Portuguese.

In addition, SuPor-2 performance exceeds the best version of TextRank that uses no linguistic knowledge – 'TextRank (PageRank - backward)' – by about 14%.

System	ROUGE NGram(1,1)
SuPor-2	0,5839
TextRank+Thesaurus	0,5603
TextRank+Stem+StopwordsRem	0,5426
TextRank (PageRank - backward)	0,5121
TextRank (HIT hub - forward)	0,5002
TextRank (HITS authority - backward)	0,5002
Baseline	0,4963
TextRank (PageRank - undirected)	0,4939
TextRank (HITS authority - forward)	0,4834
TextRank (HIT hub - backward)	0,4834
TextRank (HITS authority -undirected)	0,4814
TextRank (HIT hub - undirected)	0,4814
TextRank (PageRank - forward)	0,4574

Table 1. Informativeness comparison between extractive summarizers

7 Final Remarks

A critical issue in the comparison presented above is the contrast between having an unsupervised or supervised summarizer, which is related to the issue on having linguistic-independent extractive summarizers. Perhaps the question that we should pose here is how interesting and useful an extractive automatic summarizer that is totally independent from linguistic knowledge can actually be. To our view, the more non-informative an extract, the less useful it may be. So, summarizers that do not reach a minimum threshold concerning informativeness are deemed to failure nowadays. Clearly, SuPor-2 requires language-dependent resources, but its main extraction procedure is still general enough to make it portable and adaptable to new domains and languages. Hence, SuPor-2 assessment suggests that it may be interesting to scale up SuPor-2.

Considering that SuPor-2 is one of the best extractive summarizers for Brazilian Portuguese texts (Leite and Rino, 2006) and 'TextRank+Thesaurus' performed only 4% below it, we can also argue in favor of providing even simple linguistic procedures for extractive AS. The latter system shows that TextRank can yield extracts nearly as informative as those produced by the former, when embedding stemming and stopwords removal. It can also perform AS with little computational effort and no training, when compared to the supervised SuPor-2. As a conclusion, we see that some linguistic knowledge may boost TextRank performance without too much effort, since language-dependent resources for preprocessing texts in natural language are usually available and easy to handle, concerning our addressed approach.

There are many experiments that may be derived from our discussion in this paper (1) Although the reported results suggest that linguistic knowledge does make a difference when embedded in language-free extractive summarizers, the performance of the top systems assessed through ROUGE should be more comprehensively licensed through additional assessment tasks. (2) These could also incorporate other graph-based algorithms than TextRank, such as the LexRank one, aiming at reassuring our claim and scaling up graph-based approaches. (3) Since we addressed language-independence (thus portability) versus language-dependence for informativeness, it would also be interesting to explore other domains or languages to support our claim or, at least, to look for other findings to confirm if linguistic knowledge indeed makes a difference. (4) Other TextRank variations could also be explored, to see if adding more features would make TextRank closer to SuPor-2.

Acknowledgements

This work has been supported by the Brazilian research funding agencies CNPq, CAPES and FAPESP.

References

B. C. Dias-da-Silva, M. F. Oliveira, H. R. Moraes, C. Paschoalino, R. Hasegawa, D. Amorin and A. C. Nascimento. 2000. Construção de um Thesaurus Eletrônico para o Português do Brasil. In *Proceedings of the V Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR 2000*), São Carlos, Brasil , 1-11.

C. Lin and E. H. Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-occurrence Statistics. In *Proceedings of Language Technology Conference (HLT-NAACL 2003),* Edmonton, Canada.

D. Marcu. 1999. Discourse Trees Are Good Indicators of Importance in Text. In Mani, I., Maybury, M. T. (Eds.). 1999. *Advances in Automatic Text Summarization*. MIT Press.

D. S. Leite and L. H. M. Rino. 2006. Selecting a Feature Set to Summarize Texts in Brazilian Portuguese. In J. S. Sichman et al. (eds.): *Proceedings of 18th. Brazilian Symposium on Artificial Intelligence (SBIA'06) and 10th. Ibero-American Artificial Intelligence Conference (IBERAMIA'06)*. Lecture Notes on Artificial Intelligence, No. 4140, Springer-Verlag, 462-471.

G. Erkan and D R. Radev. 2004. LexRank: Graph-based Lexical Centrality as Salience in Text Summarization. *Journal of Artificial Intelligence Research* 22:457-479

G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross and K. Miller. 1990. Introduction to WordNet: An Online Lexical Database. *International Journal of Lexicography* 3(4):235-244

G. Salton, and C. Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24 : 513-523.. Reprinted in: K. Sparck-Jones and P. Willet (eds.). 1997. *Readings in Information Retrieval*, Morgan Kaufmann, 323-328.

H. Luhn. 1958. The automatic creation of literature abstracts. *IBM Journal of Research and Development* 2:159-165

H. P. Edmundson. 1969. New methods in automatic extracting. *Journal of the Association for Computing Machinery* 16:264-285.

I. Witten and E. Frank. 2005. *Data Mining: Practical machine learning tools and techniques*, 2nd ed. Morgan Kaufmann, San Francisco.

I. Mani. 2001. *Automatic Summarization*. John Benjamin's Publishing Company.

I. Mani and M. T. Maybury. 1999. *Advances in Automatic Text Summarization*. MIT Press.

J. Caldas Junior, C. Y. M. Imamura and S. O. Rezende. Avaliação de um Algoritmo de Stemming para a Língua Portuguesa. In *Proceedings of the 2nd Congress of Logic Applied to Technology (LABTEC'2001)*, vol. II. Faculdade SENAC de Ciências Exatas e Tecnologia, São Paulo, Brasil (2001), 267-274.

J. M. Kleinberg. 1999. Authoritative sources in hyperlinked environment. *Journal of the ACM*, 46(5):604-632.

J. Kupiec, J. Pedersen and F. Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th ACM-SIGIR Conference on Research & Development in Information Retrieval*, 68-73.

J. Larocca Neto, A. D. Santos, C. A. A. Kaestner and A. A. Freitas. 2000. Generating Text Summaries through the Relative Importance of Topics. *Lecture Notes in Artificial Intelligence*, No. 1952. Springer-Verlag, 200-309

M. A. Hearst. 1993. TextTiling: A Quantitative Approach to Discourse Segmentation. Technical Report 93/24. University of California, Berkeley.

M. F. Porter. 1980. An Algorithm for Suffix Stripping. *Program*, 14 (3) : 130-137

R. Mihalcea and P. Tarau. 2004. TextRank: Bringing Order into Texts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, Barcelona, Spain, July.

R. Mihalcea. 2005. Language Independent Extractive Summarization. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics*, Companion Volume (ACL2005), Ann Arbor, MI, June.

R. Barzilay and M. Elhadad. 1997. Using lexical chains for text summarization. In *Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97)*, ACL, Madrid, Spain.

S. Brin and L. Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems* 30:1-7.

T. A. S. Pardo and L.H.M. Rino. 2003. TeMário: A corpus for automatic text summarization (in Portuguese). NILC Tech. Report NILC-TR-03-09

Timestamped Graphs: Evolutionary Models of Text for Multi-document Summarization

Ziheng Lin, Min-Yen Kan
School of Computing
National University of Singapore
Singapore 177543
{linzihen, kanmy}@comp.nus.edu.sg

Abstract

Current graph-based approaches to automatic text summarization, such as LexRank and TextRank, assume a static graph which does not model how the input texts emerge. A suitable evolutionary text graph model may impart a better understanding of the texts and improve the summarization process. We propose a timestamped graph (TSG) model that is motivated by human writing and reading processes, and show how text units in this model emerge over time. In our model, the graphs used by LexRank and TextRank are specific instances of our timestamped graph with particular parameter settings. We apply timestamped graphs on the standard DUC multi-document text summarization task and achieve comparable results to the state of the art.

1 Introduction

Graph-based ranking algorithms such as Kleinberg's HITS (Kleinberg, 1999) or Google's PageRank (Brin and Page, 1998) have been successfully applied in citation network analysis and ranking of webpages. These algorithms essentially decide the weights of graph nodes based on global topological information. Recently, a number of graph-based approaches have been suggested for NLP applications. Erkan and Radev (2004) introduced LexRank for multi-document text summarization. Mihalcea and Tarau (2004) introduced TextRank for keyword and sentence extractions. Both LexRank and TextRank assume a fully connected, undirected graph, with text units as nodes and similarity as edges. After graph construction, both algorithms use a random walk on the graph to redistribute the node weights.

Many graph-based algorithms feature an evolutionary model, in which the graph changes over timesteps. An example is a citation network whose edges point backward in time: papers (usually) only reference older published works. References in old papers are static and are not updated. Simple models of Web growth are exemples of this: they model the chronological evolution of the Web in which a new webpage must be linked by an incoming edge in order to be publicly accessible and may embed links to existing webpages. These models differ in that they allow links in previously generated webpages to be updated or rewired. However, existing graph models for summarization – LexRank and TextRank – assume a static graph, and do not model how the input texts evolve. The central hypothesis of this paper is that modeling the evolution of input texts may improve the subsequent summarization process. Such a model may be based on human writing/reading process and should show how just composed/consumed units of text relate to previous ones. By applying this model over a series of timesteps, we obtain a representation of how information flows in the construction of the document set and leverage this to construct automatic summaries.

We first introduce and formalize our timestamped graph model in next section. In particular, our formalization subsumes previous works: we show in Section 3 that the graphs used by LexRank and TextRank are specific instances of our timestamped graph. In Section 4, we discuss how the resulting graphs are applied to automatic multi-document text summarization: by counting node in-degree or applying a random walk algorithm to smooth the information flow. We apply these models to create an extractive summarization program

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 25–32,
Rochester, April 2007

and apply it to the standard Document Understanding Conference (DUC) datasets. We discuss the resulting performance in Section 5.

2 Timestamped Graph

We believe that a proper evolutionary graph model of text should capture the writing and reading processes of humans. Although such human processes vary widely, when we limit ourselves to expository text, we find that both skilled writers and readers often follow conventional rhetorical styles (Endres-Niggemeyer, 1998; Liddy, 1991). In this work, we explore how a simple model of evolution affects graph construction and subsequent summarization. In this paper, our work is only exploratory and not meant to realistically model human processes and we believe that deep understanding and inference of rhetorical styles (Mann and Thompson, 1988) will improve the fidelity of our model. Nevertheless, a simple model is a good starting point.

We make two simple assumptions:

1: Writers write articles from the first sentence to the last;

2: Readers read articles from the first sentence to the last.

The assumptions suggest that we add sentences into the graph in chronological order: we add the first sentence, followed by the second sentence, and so forth, until the last sentence is added.

These assumptions are suitable in modeling the growth of individual documents. However when dealing with multi-document input (common in DUC), our assumptions do not lead to a straightforward model as to which sentences should appear in the graph before others. One simple way is to treat multi-document problems simply as multiple instances of the single document problem, which evolve in parallel. Thus, in multi-document graphs, we add a sentence from each document in the input set into the graph at each timestep. Our model introduces a skew variable to model this and other possible variations, which is detailed later.

The pseudocode in Figure 1 summarizes how we build a timestamped graph for multi-document input set. Informally, we build the graph iteratively, introducing new sentence(s) as node(s) in

Input:	**M**, a cluster of m documents relating to a common event;
Let:	**i** = index to sentences, initially 1; **G** = the timestamped graph, initially empty.
Step 1:	Add the **i**th sentence of all documents into **G**.
Step 2:	Let each existing sentence in **G** choose and connect to **one** other existing sentence in **G**. The chosen sentence must be sentence which has not been previously chosen by this sentence in previous iterations.
Step 3:	**if** there are no new sentences to add, **break;** **else** i++, goto Step 1.
Output:	**G**, a timestamped graph.

Figure 1: Pseudocode for a specific instance of a timestamped graph algorithm

the graph at each timestep. Next, all sentences in the graph pick other previously unconnected ones to draw a directed edge to. This process continues until all sentences are placed into the graph.

Figure 2 shows this graph building process in mid-growth, where documents are arranged in columns, with d_x represents the x^{th} document and s_y represents the y^{th} sentence of each document. The bottom shows the n^{th} sentences of all m documents being added simultaneously to the graph. Each new node can either connect to a node in the existing graph or one of the other m-1 new nodes. Each existing node can connect to another existing node or to one of the m newly-introduced nodes. Note that this model differs from the citation networks in such that new outgoing edges are introduced to old nodes, and differs from previous models for Web growth as it does not require new nodes to have incoming edges.

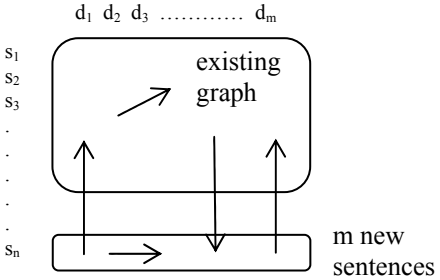
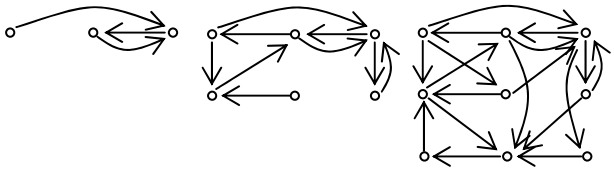


Figure 2: Snapshot of a timestamped graph.

Figure 3 shows an example of the graph building process over three timesteps, starting from an empty graph. Assume that we have three documents and each document has three sentences. Let d_xs_y indicate the y^{th} sentence in the x^{th} document. At timestep 1, sentences d_1s_1, d_2s_1 and d_3s_1 are

added to the graph. Three edges are introduced to the graph, in which the edges are chosen by some strategy; perhaps by choosing the candidate sentence by its maximum cosine similarity with the sentence under consideration. Let us say that this process connects $d_1s_1 \rightarrow d_3s_1$, $d_2s_1 \rightarrow d_3s_1$ and $d_3s_1 \rightarrow d_2s_1$. At timestep 2, sentences d_1s_2, d_2s_2 and d_3s_2 are added to the graph and six new edges are introduced to the graph. At timestep 3, sentences d_1s_3, d_2s_3 and d_3s_3 are added to the graph, and nine new edges are introduced.



(a) Timestep 1 (b) Timestep 2 (c) Timestep 3

Figure 3: An example of the growth of a timestamped graph.

The above illustration is just one instance of a timestamped graph with specific parameter settings. We generalize and formalize the timestamped graph algorithm as follows:

Definition: A timestamped graph algorithm *tsg*(*M*) is a 9-tuple (*d, e, u, f, σ, t, i, s, τ*) that specifies a resulting algorithm that takes as input the set of texts *M* and outputs a graph *G*, where:

d specifies the direction of the edges, $d \in \{f, b, u\}$;
e is the number of edges to add for each vertex in *G* at each timestep, $e \in \mathbb{Z}^+$;
u is 0 or 1, where 0 and 1 specifies unweighted and weighted edges, respectively;
f is the inter-document factor, $0 \leq f \leq 1$;
σ is a vertex selection function $\sigma(u, G)$ that takes in a vertex *u* and *G*, and chooses a vertex $v \in G$;
t is the type of text units, $t \in \{$*word, phrase, sentence, paragraph, document*$\}$;
i is the node increment factor, $i \in \mathbb{Z}^+$;
s is the skew degree, $s \geq -1$ and $s \in \mathbb{Z}$, where -1 represent free skew and 0 no skew;
τ is a document segmentation function $\tau(\bullet)$.

In the TSG model, the first set of parameters *d*, *e*, *u*, *f* deal with the properties of edges; *σ*, *t*, *i*, *s* deal with properties of nodes; finally, *τ* is a function that modifies input texts. We now discuss the first eight parameters; the relevance of *τ* will be expanded upon later in the paper.

2.1 Edge Settings

We can specify the direction of information flow by setting different *d* values. When a node v_1 chooses another node v_2 to connect to, we set *d* to *f* to represent a forward (outgoing) edge. We say that v_1 propagates some of its information into v_2. When letting a node v_1 choose another node v_2 to connect to v_1 itself, we set *d* to *b* to represent a backward (incoming) edge, and we say that v_1 receives some information from v_2. Similarly, $d = u$ specifies undirected edges in which information propagates in both directions. The larger amount of information a node receives from other nodes, the higher the importance of this node.

Our toy example in Figure 3 has small dimensions: three sentences for each of three documents. Experimental document clusters often have much larger dimensions. In DUC, clusters routinely contain over 25 documents, and the average length for documents can be as large as 50 sentences. In such cases, if we introduce one edge for each node at each timestep, the resulting graph is loosely connected. We let *e* be the number of outgoing edges for each sentence in the graph at each timestep. To introduce more edges into the graph, we increase *e*.

We can also incorporate unweighted or weighted edges into the graph by specifying the value of *u*. Unweighted edges are good when ranking algorithms based on in-degree of nodes are used. However, unlike links between webpages, edges between text units often have weights to indicate connection strength. In these cases, unweighted edges lose information and a weighted representation may be better, such as in cases where PageRank-like algorithms are used for ranking.

Edges can represent information flow from one node to another. We may prefer intra-document edges over inter-document edges, to model the intuition that information flows within the same document more likely than across documents. Thus we introduce an inter-document factor *f*, where $0 \leq f \leq 1$. When this feature is smaller than 1, we replace the weight *w* for inter-document edges by *fw*.

2.2 Node Settings

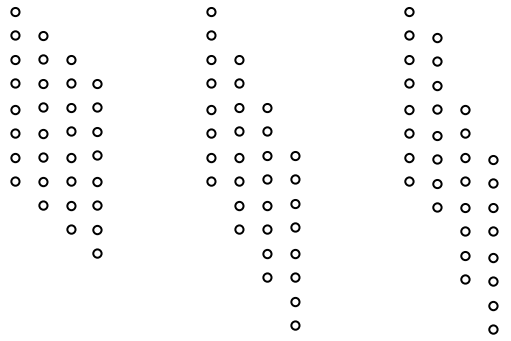
In Figure 1 Step 2, every existing node has a chance to choose another existing node to connect to. Which node to choose is decided by the selection strategy σ. One strategy is to choose the node with the highest similarity. There are many similarity functions to use, including token-based Jaccard similarity, cosine similarity, or more complex models such as concept links (Ye et al., 2005).

t controls the type of text unit that represents nodes. Depending on the application, text units can be words, phrases, sentences, paragraphs or even documents. In the task of automatic text summarization, systems are conveniently assessed by letting text units be sentences.

i controls the number of sentences entering the graph at every iteration. Certain models, such as LexRank, introduce all of the input sentences in one time step (i.e., $i = L_{max}$, where L_{max} is the maximum length of the input documents), completing the construction of *G* in one step. However, to model time evolution, *i* needs to be set to a value smaller than this.

Most relevant to our study is the skew parameter *s*. Up to now, the TSG models discussed all assume that authors start writing all documents in the input set at the same time. It is reflected by adding the first sentences of all documents simultaneously. However in reality, some documents are authored later than others, giving updates or reporting changes to events reported earlier. In DUC document clusters, news articles are typically taken from two or three different newswire sources. They report on a common event and thus follow a storyline. A news article usually gives summary about what have been reported in early articles, and gives updates or changes on the same event.

To model this, we arrange the documents in accordance with the publishing time of the documents. The earliest document is assigned to column 1, the second earliest document to column 2, and so forth, until the latest document is assigned to the last column. The graph construction process is the same as before, except that we delay adding the first sentences of later documents until a proper iteration, governed by *s*. With $s = 1$, we delay the addition of the first sentence of column 2 until the second timestep, and delay the addition of the first sentence of column 3 until the third timestep. The resulting timestamped graph is skewed by 1 timestep (Figure 4 (a)). We can increase the skew degree *s* if the time intervals between publishing time of documents are large. Figure 4 (b) shows a timestamped graph skewed by 2 timesteps. We can also skew a graph freely by setting *s* to -1. When we start to add the first sentence d_is_1 of a document d_i, we check whether there are existing sentences in the graph that want to connect to d_is_1 (i.e., that $\sigma(\bullet, G) = d_is_1$). If there is, we add d_is_1 to the graph; else we delay the addition and reassess again in next timestep. The result is a freely skewed graph (Figure 4 (c)). In Figure 4 (c), we start adding the first sentences of documents d_2 to d_4 at timesteps 2, 5 and 7, respectively. At timestep 1, d_1s_1 is added into the graph. At timestep 2, an existing node (d_1s_1 in this case) wants to connect to d_2s_1, so d_2s_1 is added. d_3s_1 is added at timestep 5 as no existing node wants to connect to d_3s_1 until timestep 5. Similarly, d_4s_1 is added until some nodes choose to connect to it at timestep 7. Notice that we hide edges in Figure 4 for clarity.



(a) Skewed by 1 (b) Skewed by 2 (c) Freely skewed

Figure 4: Skewing the graphs. Edges are hidden for clarity. For each graph, the leftmost column is the earliest document. Documents are then chronologically ordered, with the rightmost one being the latest.

3 Comparison and Properties of TSG

The TSG representation generalizes many possible specific algorithm configurations. As such, it is natural that previous works can be cast as specific instances of a TSG. For example, we can succinctly represent the algorithm used in the running example in Section 2 as the tuple (*f, 1, 0, 1, max-cosine-based, sentence, 1, 0, null*). LexRank and TextRank can also be cast as TSGs: (*u, N, 1, 1, cosine-based, sentence,* L_{max}*, 0, null*) and (*u, L, 1, 1, modified-co-occurrence-based, sentence, L, 0,*

null). As LexRank is applied in multi-document summarizations, *e* is set to the total number of sentences in the cluster, N, and *i* is set to the maximum document length in the cluster, L_{max}. TextRank is applied in single-document summarization, so both its *e* and *i* are set to the length of the input document, *L*. This compact notation emphasizes the salient differences between these two algorithm variants: namely that, *e*, σ and *i*.

Despite all of these possible variations, all timestamped graphs have two important features, regardless of their specific parameter settings. First, nodes that were added early have more chosen edges than nodes added later, as visible in Figure 3 (c). If forward edges (*d* = *f*) represent information flow from one node to another, we can say that more information is flowing from these early nodes to the rest of the graph. The intuition for this is that, during the writing process of articles, early sentences have a greater influence to the development of the articles' ideas; similarly, during the reading process, sentences that appear early contribute more to the understanding of the articles.

The fact that early nodes stay in the graph for a longer time leads to the second feature: early nodes may attract more edges from other nodes, as they have larger chance to be chosen and connected by other nodes. This is also intuitive for forward edges (*d* = *f*): during the writing process, later sentences refer back to early sentences more often than vice versa; and during the reading process, readers tend to re-read early sentences when they are not able to understand the current sentence.

4 Random Walk

Once a timestamped graph is built, we want to compute an importance score for each node. These scores are then used to determine which nodes (sentences) are the most important to extract summaries from. The graph *G* shows how information flows from node to node, but we have yet to let the information actually flow. One method to do this is to use the in-degree of each node as the score. However, most graph algorithms now use an iterative method that allows the weights of the nodes redistribute until stability is reached. One method for this is by applying a random walk, used in PageRank (Brin and Page, 1998). In PageRank the Web is treated as a graph of webpages connected by links. It assumes users start from a random webpage, moving from page to page by following the links. Each user follows the links at random until he gets "bored" and jumps to a random webpage. The probability of a user visiting a webpage is then proportional to its PageRank score. PageRank can be iteratively computed by:

$$PR(u) = \frac{\alpha}{N} + (1-\alpha) \sum_{v \in In(u)} \frac{1}{|Out(v)|} PR(v) \quad (1)$$

where *N* is the total number of nodes in the graph, *In*(*u*) is the set of nodes that point to *u*, and *Out*(*u*) is the set of nodes that node *u* points to. α is a damping factor that can be set between 0 and 1, which has the role of integrating into the model the probability of jumping from a given node to another random node in the graph. In the context of web surfing, a user either clicks on a link on the current page at random with probability 1 - α, or opens a completely new random page with probability α.

Equation 1 does not take into consideration the weights of edges, as the original PageRank definition assumes hyperlinks are unweighted. Thus we can use Equation 1 to rank nodes for an unweighted timestamped graph. To integrate edge weights into the graph, we modify Eq. 1, yielding:

$$PR(u) = \frac{\alpha}{N} + (1-\alpha) \sum_{v \in In(u)} \frac{w_{vu}}{\sum_{x \in Out(v)} w_{vx}} PR(v) \quad (2)$$

where W_{vu} represents the weight of the edge pointing from *v* to *u*.

As we may have a query for each document cluster, we also wish to take queries into consideration in ranking the nodes. Haveliwala (2003) introduces a topic-sensitive PageRank computation. Equations 1 and 2 assume a random walker jumps from the current node to a random node with probability α. The key to creating topic-sensitive PageRank is that we can bias the computation by restricting the user to jump only to a random node which has non-zero similarity with the query. Otterbacher et al. (2005) gives an equation for topic-sensitive and weighted PageRank as:

$$PR(u) = \alpha \frac{sim(u,Q)}{\sum_{y \in S} sim(y,Q)} + (1-\alpha) \sum_{v \in In(u)} \frac{w_{vu}}{\sum_{x \in Out(v)} w_{vx}} PR(v) \quad (3)$$

where S is the set of all nodes in the graph, and $sim(u, Q)$ is the similarity score between node u and the query Q.

5 Experiments and Results

We have generalized and formalized evolutionary timestamped graph model. We want to apply it on automatic text summarization to confirm that these evolutionary models help in extracting important sentences. However, the parameter space is too large to test all possible TSG algorithms. We conduct experiments to focus on the following research questions that relating to 3 TSG parameters - e, u and s, and the topic-sensitivity of PageRank.

Q1: Do different e values affect the summarization process?

Q2: How do topic-sensitivity and edge weighting perform in running PageRank?

Q3: How does skewing the graph affect information flow in the graph?

The datasets we use are DUC 2005 and 2006. These datasets both consist of 50 document clusters. Each cluster consists of 25 news articles which are taken from two or three different newswire sources and are relating to a common event, and a query which contains a topic for the cluster and a sequence of statements or questions. The first three experiments are run on DUC 2006, and the last experiment is run on DUC 2005.

In the first experiment, we analyze how e, the number of chosen edges for each node at each timestep, affects the performance, with other parameters fixed. Specifically the TSG algorithm we use is the tuple (*f, e, 1, 1, max-cosine-based, sentence, 1, 0, null*), where e is being tested for different values. The node selection function *max-cosine-based* takes in a sentence s and the current graph G, computes the TFIDF-based cosine similarities between s and other sentences in G, and connects s to e sentence(s) that has(have) the highest cosine score(s) and is(are) not yet chosen by s in previous iterations. We run topic-sensitive PageRank with damping factor α set to 0.5 on the graphs. Figures 5 (a)-(b) shows the ROUGE-1 and ROUGE-2 scores with e set to 1, 2, 3, 4, 5, 6, 7, 10, 15, 20 and N, where N is the total number of sentences in the cluster. We succinctly represent LexRank graphs by the tuple (*u, N, 1, 1, cosine-based, sentence, L_{max}, 0, null*) in Section 3; it can also be represented by a slightly different tuple (*f, N, 1, 1, max-cosine-based, sentence, 1, 0, null*). It differs from the first representation in that we iteratively add 1 sentence for each document in each timestep and let all nodes in the current graph connect to every other node in the graph. In this experiment, when e is set to N, the timestamped graph is equivalent to a LexRank graph. We do not use any reranker in this experiment.

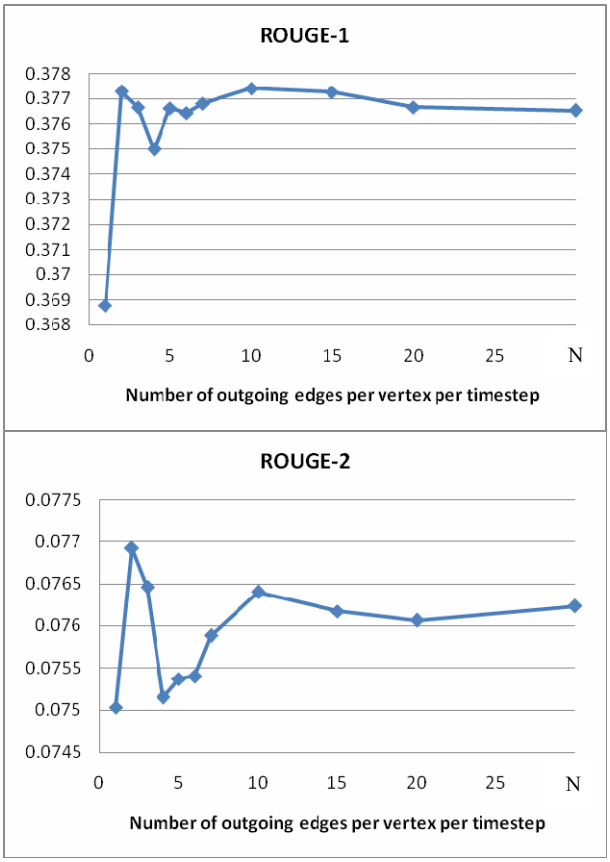


Figure 5: (a) ROUGE-1 and (b) ROUGE-2 scores for timestamped graphs with different e settings. N is the total number of sentences in the cluster.

The results allow us to make several observations. First, when e = 2, the system gives the best performance, with ROUGE-1 score 0.37728 and ROUGE-2 score 0.07692. Some values of e give better scores than LexRank graph configuration, in which e = N. Second, the system gives very bad performance when e = 1. This is because when e is set to 1, the graph is too loosely connected and is not suitable to apply random walk on it. Third, the system gives similar performance when e is set

greater than 10. The reason for this is that the higher values of *e* make the graph converge to a fully connected graph so that the performance starts to converge and display less variability.

We run a second experiment to analyze how topic-sensitivity and edge weighting affect the system performance. We use concept links (Ye et al., 2005) as the similarity function and a MMR reranker to remove redundancy. Table 1 shows the results. We observe that both topic-sensitive PageRank and weighted edges perform better than generic PageRank on unweighted timestamped graphs. When topic-sensitivity and edge weighting are both set to true, the system gives the best performance.

Topic-sensitive	Weighted edges	ROUGE-1	ROUGE-2
No	No	0.39358	0.07690
Yes	No	0.39443	0.07838
No	Yes	0.39823	0.08072
Yes	Yes	0.39845	0.08282

Table 1: ROUGE-1 and ROUGE-2 scores for different combinations of topic-sensitivity and edge weighting(u) settings.

To evaluate how skew degree *s* affects summarization performance, we use the parameter setting from the first experiment, with *e* fixed to 1. Specifically, we use the tuple (*f, 1, 1, 1, concept-link-based, sentence, 1, s, null*), with *s* set to 0, 1 and 2. Table 2 gives the evaluation results. We observe that $s = 1$ gives the best ROUGE-1 and ROUGE-2 scores. Compared to the system without skewing ($s = 0$), $s = 2$ gives slightly better ROUGE-1 score but worse ROUGE-2 score. The reason for this is that $s = 2$ introduces a delay interval that is too large. We expect that a freely skewed graph ($s = -1$) will give more reasonable delay intervals.

Skew degree	ROUGE-1	ROUGE-2
0	0.36982	0.07580
1	0.37268	0.07682
2	0.36998	0.07489

Table 2: ROUGE-1 and ROUGE-2 scores for different skew degrees.

We tune the system using different combinations of parameters, and the TSG algorithm with tuple (*f, 1, 1, 1, concept-link-based, sentence, 1, 0, null*) gives the best scores. We run this TSG algorithm with topic-sensitive PageRank and MMR reranker on DUC 2005 dataset. The results show that our system ranks third in both ROUGE-2 and ROUGE-SU4 scores.

Rank	System	ROUGE-2	System	ROUGE-SU4
1	15	0.0725	15	0.1316
2	17	0.0717	17	0.1297
3	**TSG**	**0.0712**	**TSG**	**0.1285**
4	10	0.0698	8	0.1279
5	8	0.0696	4	0.1277

Table 3: top ROUGE-2 and ROUGE-SU4 scores in DUC 2005. TSG is our system.

6 Discussion

A closer inspection of the experimental clusters reveals one problem. Clusters that consist of documents that are of similar lengths tend to perform better than those that contain extremely long documents. The reason is that a very long document introduces too many edges into the graph. Ideally we want to have documents with similar lengths in a cluster. One solution to this is that we split long documents into shorter documents with appropriate lengths. We introduce the last parameter in the formal definition of timestamped graphs, τ, which is a document segmentation function $\tau(\bullet)$. $\tau(M)$ takes in as input a set of documents M, applies segmentation on long documents to split them into shorter documents, and output a set of documents with similar lengths, M'. Slightly better results are achieved when a segmentation function is applied. One shortcoming of applying $\tau(\bullet)$ is that when a document is split into two shorter ones, the early sentences of the second half now come before the later sentences of the first half, and this may introduce inconsistencies in our representation: early sentences of the second half contribute more into later sentences of the first half than the vice versa.

7 Related Works

Dorogovtsev and Mendes (2001) suggest schemes of the growth of citation networks and the Web, which are similar to the construction process of timestamped graphs.

Erkan and Radev (2004) proposed LexRank to define sentence importance based on graph-based centrality ranking of sentences. They construct a similarity graph where the cosine similarity of each pair of sentences is computed. They introduce three different methods for computing centrality in

similarity graphs. Degree centrality is defined as the in-degree of vertices after removing edges which have cosine similarity below a pre-defined threshold. LexRank with threshold is the second method that applies random walk on an unweighted similarity graph after removing edges below a pre-defined threshold. Continuous LexRank is the last method that applies random walk on a fully connected, weighted similarity graph. LexRank has been applied on multi-document text summarization task in DUC 2004, and topic-sensitive LexRank has been applied on the same task in DUC 2006.

Mihalcea and Tarau (2004) independently proposed another similar graph-based random walk model, TextRank. TextRank is applied on keyword extraction and single-document summarization. Mihalcea, Tarau and Figa (2004) later applied PageRank to word sense disambiguation.

8 Conclusion

We have proposed a timestamped graph model which is motivated by human writing and reading processes. We believe that a suitable evolutionary text graph which changes over timesteps captures how information propagates in the text graph. Experimental results on the multi-document text summarization task of DUC 2006 showed that when *e* is set to 2 with other parameters fixed, or when *s* is set to 1 with other parameters fixed, the graph gives the best performance. It also showed that topic-sensitive PageRank and weighted edges improve summarization process. This work also unifies representations of graph-based summarization, including LexRank and TextRank, modeling these prior works as specific instances of timestamped graphs.

We are currently looking further on skewed timestamped graphs. Particularly we want to look at how a freely skewed graph propagates information. We are also analyzing in-degree distribution of timestamped graphs.

Acknowledgments

The authors would like to thank Prof. Wee Sun Lee for his very helpful comments on random walk and the construction process of timestamped graphs, and thank Xinyi Yin (Yin, 2007) for his help in spearheading the development of this work. We also would like to thank the reviewers for their helpful suggestions in directing the future of this work.

References

Jon M. Kleinberg. 1999. *Authoritative sources in a hyperlinked environment*. In Proceedings of ACM-SIAM Symposium on Discrete Algorithms, 1999.

Sergey Brin and Lawrence Page. 1998. *The anatomy of a large-scale hypertextual Web search engine*. Computer Networks and ISDN Systems, 30(1-7).

Günes Erkan and Dragomir R. Radev. 2004. *LexRank: Graph-based centrality as salience in text summarization*. Journal of Artificial Intelligence Research, (22).

Rada Mihalcea and Paul Tarau. 2004. *TextRank: Bringing order into texts*. In Proceedings of EMNLP 2004.

Rada Mihalcea, Paul Tarau, and Elizabeth Figa. 2004. *PageRank on semantic networks, with application to word sense disambiguation*. In Proceedings of COLING 2004.

S.N. Dorogovtsev and J.F.F. Mendes. 2001. *Evolution of networks*. Submitted to Advances in Physics on 6th March 2001.

Shiren Ye, Long Qiu, Tat-Seng Chua, and Min-Yen Kan. 2005. *NUS at DUC 2005: Understanding documents via concepts links*. In Proceedings of DUC 2005.

Xinyi Yin, 2007. *Random walk and web information processing for mobile devices*. PhD Thesis.

Taher H. Haveliwala. 2003. *Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search*. IEEE Transactions on Knowledge and Data Engineering

Jahna Otterbacher, Günes Erkan and Dragomir R. Radev. 2005. *Using Random Walks for Question-focused Sentence Retrieval.* In Proceedings of HLT/EMNLP 2005.

Brigitte Endres-Niggemeyer. 1998. *Summarizing information*. Springer New York.

Elizabeth D. Liddy. 1991. *The discourse-level structure of empirical abstracts: an exploratory study*. Information Processing and Management 27(1):55-81.

William C. Mann and Sandra A. Thompson. 1988. *Rhetorical structure theory: Towards a functional theory of text organization*. Text 8(3): 243-281.

Unigram Language Models using Diffusion Smoothing over Graphs

Bruno Jedynak
Dept. of Appl. Mathematics and Statistics
Center for Imaging Sciences
Johns Hopkins University
Baltimore, MD 21218-2686
bruno.jedynak@jhu.edu

Damianos Karakos
Dept. of Electrical and Computer Engineering
Center for Language and Speech Processing
Johns Hopkins University
Baltimore, MD 21218-2686
damianos@jhu.edu

Abstract

We propose to use graph-based diffusion techniques with data-dependent kernels to build unigram language models. Our approach entails building graphs, where each vertex corresponds uniquely to a word from a closed vocabulary, and the existence of an edge (with an appropriate weight) between two words indicates some form of similarity between them. In one of our constructions, we place an edge between two words if the number of times these words were seen in a training set differs by at most one count. This graph construction results in a similarity matrix with small intrinsic dimension, since words with the same counts have the same neighbors. Experimental results from a benchmark task from language modeling show that our method is competitive with the Good-Turing estimator.

1 Diffusion over Graphs

1.1 Notation

Let $G = (V, E)$ be an undirected graph, where V is a finite set of vertices, and $E \subset V \times V$ is the set of edges. Also, let $\mathcal{V}$ be a vocabulary of words, whose probabilities we want to estimate. Each vertex corresponds uniquely to a word, i.e., there is a one-to-one mapping between $\mathcal{V}$ and V. Without loss of generality, we will use V to denote both the set of words and the set of vertices. Moreover, to simplify notation, we assume that the letters x, y, z will always denote vertices of G.

The existence of an edge between x, y will be denoted by $x \sim y$. We assume that the graph is strongly connected (i.e., there is a path between any two vertices). Furthermore, we define a non-negative real valued function w over $V \times V$, which plays the role of the *similarity* between two words (the higher the value of $w(x, y)$, the more similar words x, y are). In the experimental results section, we will compare different measures of similarity between words which will result in different smoothing algorithms. The degree of a vertex is defined as

$$d(x) = \sum_{y \in V : x \sim y} w(x, y). \tag{1}$$

We assume that for any vertex x, $d(x) > 0$; that is, every word is similar to at least some other word.

1.2 Smoothing by Normalized Diffusion

The setting described here was introduced in (Szlam et al., 2006). First, we define a Markov chain $\{X_t\}$, which corresponds to a random walk over the graph G. Its initial value is equal to X_0, which has distribution π_0. (Although π_0 can be chosen arbitrarily, we assume in this paper that it is equal to the empirical, unsmoothed, distribution of words over a training set.) We then define the transition matrix as follows:

$$T(x, y) = P(X_1 = y | X_0 = x) = d^{-1}(x) w(x, y). \tag{2}$$

This transition matrix, together with π_0, induces a distribution over V, which is equal to the distribu-

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 33–36,
Rochester, April 2007

tion π_1 of X_1:

$$\pi_1(y) = \sum_{x \in V} T(x,y)\pi_0(x). \quad (3)$$

This distribution can be construed as a *smoothed* version of π_0, since the π_1 probability of an unseen word will always be non-zero, if it has a non-zero similarity to a seen word. In the same way, a whole sequence of distributions $\pi_2, \pi_3, \ldots$ can be computed; we only consider π_1 as our smoothed estimate in this paper. (One may wonder whether the *stationary* distribution of this Markov chain, i.e., the limiting distribution of X_t, as $t \to \infty$, has any significance; we do not address this question here, as this limiting distribution may have very little dependence on π_0 in the Markov chain cases under consideration.)

1.3 Smoothing by Kernel Diffusion

We assume here that for any vertex x, $w(x,x) = 0$ and that w is symmetric. Following (Kondor and Lafferty, 2002), we define the following matrix over $V \times V$

$$H(x,y) = w(x,y)\delta(x \sim y) - d(x)\delta(x = y), \quad (4)$$

where $\delta(u)$ is the delta function which takes the value 1 if property u is true, and 0 otherwise. The negative of the matrix H is called the Laplacian of the graph and plays a central role in spectral graph theory (Chung, 1997). We further define the heat equation over the graph G as

$$\frac{\partial}{\partial t} K_t = HK_t, \; t > 0, \quad (5)$$

with initial condition $K_0 = I$, where K_t is a time-dependent square matrix of same dimension as H, and I is the identity matrix. $K_t(x,y)$ can be interpreted as the amount of heat that reaches vertex x at time t, when starting with a unit amount of heat concentrated at y. Using (1) and (4), the right hand side of (5) expands to

$$HK_t(x,y) = \sum_{z:z \sim x} w(x,z)\left(K_t(z,y) - K_t(x,y)\right). \quad (6)$$

From this equation, we see that the amount of heat at x will increase (resp. decrease) if the current amount of heat at x (namely $K_t(x,y)$) is smaller (resp. larger) than the weighted average amount of heat at the neighbors of x, thus causing the system to reach a steady state.

The heat equation (5) has a unique solution which is the matrix exponential $K_t = exp(tH)$, (see (Kondor and Lafferty, 2002)) and which can be defined equivalently as

$$e^{tH} = \lim_{n \to +\infty} \left(I + \frac{tH}{n}\right)^n \quad (7)$$

or as

$$e^{tH} = I + tH + \frac{t^2}{2!}H^2 + \frac{t^3}{3!}H^3 + \cdots \quad (8)$$

Moreover, if the initial condition is replaced by $K_0(x,y) = \pi_0(x)\delta(x = y)$ then the solution of the heat equation is given by the matrix product $\pi_1 = K_t\pi_0$. In the following, π_0 will be the empirical distribution over the training set and t will be chosen by trial and error. As before, π_1 will provide a smoothed version of π_0.

2 Unigram Language Models

Let *Tr* be a training set of n tokens, and T a separate test set of m tokens. We denote by $n(x), m(x)$ the number of times the word x has been seen in the training and test set, respectively. We assume a closed vocabulary $\mathcal{V}$ containing K words. A unigram model is a probability distribution π over the vocabulary $\mathcal{V}$. We measure its performace using the average code length (Cover and Thomas, 1991) measured on the test set:

$$l(\pi) = -\frac{1}{|T|}\sum_{x \in \mathcal{V}} m(x)\log_2 \pi(x). \quad (9)$$

The empirical distribution over the training set is

$$\pi_0(x) = \frac{n(x)}{n}. \quad (10)$$

This estimate assigns a probability 0 to all unseen words, which is undesirable, as it leads to zero probability of word sequences which can actually be observed in practice. A simple way to smooth such estimates is to add a small, not necessarily integer, count to each word leading to the so-called add-β estimate π_β, defined as

$$\pi_\beta(x) = \frac{n(x) + \beta}{n + \beta K}. \quad (11)$$

One may observe that

$$\pi_\beta(x) = (1-\lambda)\pi_0(x) + \lambda\frac{1}{K}, \text{ with } \lambda = \frac{\beta K}{n+\beta K}. \quad (12)$$

Hence add-β estimators perform a linear interpolation between π_0 and the uniform distribution over the entire vocabulary.

In practice, a much more efficient smoothing method is the so-called Good-Turing (Orlitsky et al., 2003; McAllester and Schapire, 2000). The Good-Turing estimate is defined as

$$\begin{aligned}\pi_{GT}(x) &= \frac{r_{n(x)+1}(n(x)+1)}{n r_{n(x)}}, \text{ if } n(x) < M \\ &= \alpha\,\pi_0(x), \text{ otherwise,}\end{aligned}$$

where r_j is the number of distinct words seen j times in the training set, and α is such that π_{GT} sums up to 1 over the vocabulary. The threshold M is empirically chosen, and usually lies between 5 and 10. (Choosing a much larger M decreases the performance considerably.)

The Good-Turing estimator is used frequently in practice, and we will compare our results against it. The add-β will provide a baseline, as well as an idea of the variation between different smoothers.

3 Graphs over sets of words

Our objective, in this section, is to show how to design various graphs on words; different choices for the edges and for the weight function w lead to different smoothings.

3.1 Full Graph and add-β Smoothers

The simplest possible choice is the complete graph, where all vertices are pair-wise connected. In the case of normalized diffusion, choosing

$$w(x,y) = \alpha\delta(x=y) + 1, \quad (13)$$

with $\alpha \neq 0$ leads to the add-β smoother with parameter $\beta = \alpha^{-1}n$.

In the case of kernel smoothing with the complete graph and $w \equiv 1$, one can show, see (Kondor and Lafferty, 2002) that

$$\begin{aligned}K_t(x,y) &= K^{-1}\left(1+(K-1)e^{-Kt}\right) \text{ if } x = y \\ &= K^{-1}\left(1-e^{-Kt}\right) \text{ if } x \neq y.\end{aligned}$$

This leads to another add-β smoother.

3.2 Graphs based on counts

A more interesting way of designing the word graph is through a similarity function which is based on the training set. For the normalized diffusion case, we propose the following

$$w(x,y) = \delta(|n(x) - n(y)| \leq 1). \quad (14)$$

That is, 2 words are "similar" if they have been seen a number of times which differs by at most one. The obtained estimator is denoted by π_{ND}. After some algebraic manipulations, we obtain

$$\pi_{ND}(y) = \frac{1}{n}\sum_{j=n(y)-1}^{n(y)+1}\frac{j r_j}{r_{j-1}+r_j+r_{j+1}}. \quad (15)$$

This estimator has a Good-Turing "flavor". For example, the total mass associated with the unseen words is

$$\sum_{y;n(y)=0}\pi_1(y) = \frac{1}{n}\frac{r_1}{1+\frac{r_1}{r_0}+\frac{r_2}{r_0}}. \quad (16)$$

Note that the estimate of the unseen mass, in the case of the Good-Turing estimator, is equal to $n^{-1}r_1$, which is very close to the above when the vocabulary is large compared to the size of the training set (as is usually the case in practice).

Similarly, in the case of kernel diffusion, we choose $w \equiv 1$ and

$$x \sim y \iff |n(x) - n(y)| \leq 1 \quad (17)$$

The time t is chosen to be $|V|^{-1}$. The smoother cannot be computed in closed form. We used the formula (7) with $n = 3$ in the experiments. Larger values of n did not improve the results.

4 Experimental Results

In our experiments, we used Sections 00-22 (consisting of $\sim 10^6$ words) of the UPenn Treebank corpus for training, and Sections 23-24 (consisting of $\sim 10^5$ words) for testing. We split the training set into 10 subsets, leading to 10 datasets of size $\sim 10^5$ tokens each. The first of these sets was further split in subsets of size $\sim 10^4$ tokens each. Averaged results are presented in the tables below for various choices of the training set size. We show the mean code-length, as well as the standard deviation (when

	mean code length	std
$\pi_\beta, \beta = 1$	12.94	0.05
π_{GT}	11.40	0.08
π_{ND}	11.42	0.08
π_{KD}	11.51	0.08

Table 1: Results with training set of size $\sim 10^4$.

	mean code length	std
$\pi_\beta, \beta = 1$	11.10	0.03
π_{GT}	10.68	0.06
π_{ND}	10.69	0.06
π_{KD}	10.74	0.08

Table 2: Results with training set of size $\sim 10^5$.

available). In all cases, we chose $K = 10^5$ as the fixed size of our vocabulary.

The results show that π_{ND}, the estimate obtained with the Normalized Diffusion, is competitive with the Good-Turing π_{GT}. We performed a Kolmogorov-Smirnov test in order to determine if the code-lengths obtained with π_{ND} and π_{GT} in Table 1 differ significantly. The result is negative (P-value = .65), and the same holds for the larger training set in Table 2 (P-value=.95). On the other hand, π_{KD} (obtained with Kernel Diffusion) is not as efficient, but still better than add-β with $\beta = 1$.

5 Concluding Remarks

We showed that diffusions on graphs can be useful for language modeling. They yield naturally smooth estimates, and, under a particular choice of the "similarity" function between words, they are competitive with the Good-Turing estimator, which is considered to be the state-of-the-art in unigram language modeling. We plan to perform more experiments with other definitions of similarity between words. For example, we expect similarities based on co-occurence in documents, or based on notions of semantic closeness (computed, for instance, using the WordNet hierarchy) to yield significant improvements over estimators which are only based on word counts.

	mean code length
$\pi_\beta, \beta = 1$	10.34
π_{GT}	10.30
π_{ND}	10.30
π_{KD}	10.31

Table 3: Results with training set of size $\sim 10^6$.

References

F. Chung. 1997. *Spectral Graph Theory*. Number 92 in CBMS Regional Conference Series in Mathematics. American Mathematical Society.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of Information Theory*. John Wiley & Sons, Inc.

Risi Imre Kondor and John Lafferty. 2002. Diffusion kernels on graphs and other discrete input spaces. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322.

David McAllester and Robert E. Schapire. 2000. On the convergence rate of Good-Turing estimators. In *Proc. 13th Annu. Conference on Comput. Learning Theory*.

Alon Orlitsky, Narayana P. Santhanam, and Junan Zhang. 2003. Always Good Turing: Asymptotically optimal probability estimation. In *FOCS '03: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*.

Arthur D. Szlam, Mauro Maggioni, and Ronald R. Coifman. 2006. A general framework for adaptive regularization based on diffusion processes on graphs. Technical report, YALE/DCS/TR1365.

Transductive Structured Classification through Constrained Min-Cuts

Kuzman Ganchev **Fernando Pereira**
Computer and Information Science
University of Pennsylvania
Philadelphia PA
{kuzman,pereira}@cis.upenn.edu

Abstract

We extend the Blum and Chawla (2001) graph min-cut algorithm to structured problems. This extension can alternatively be viewed as a joint inference method over a set of training and test instances where parts of the instances interact through a prespecified associative network. The method has has an efficient approximation through a linear-programming relaxation. On small training data sets, the method achieves up to 34.8% relative error reduction.

1 Introduction

We describe a method for transductive classification in structured problems. Our method extends the Blum and Chawla (2001) algorithm for transductive classification. In that algorithm, each training and test instance is represented by a vertex in a graph. The algorithm finds the min-cut that separates the positively and negatively labeled instances. We give a linear program that implements an approximation of this algorithm and extend it in several ways. First, our formulation can be used in cases where there are more than two labels. Second, we can use the output of a classifier to provide a prior preference of each instance for a particular label. This lets us trade off the strengths of the min-cut algorithm against those of a standard classifier. Finally, we extend the algorithm further to deal with structured output spaces, by encoding parts of instances as well as constraints that ensure a consistent labeling of an entire instance.

The rest of this paper is organized as follows. Section 2 explains what we mean by transductive classification and by structured problems. Section 3 reviews the Blum and Chawla (2001) algorithm, how we formulate it as a linear program and our proposed extensions. Section 4 relates our proposal to previous work. Section 5 describes our experimental results on real and synthetic data and Section 6 concludes the paper.

2 Concepts and Notation

In this work we combine two separate approaches to learning: transductive methods, in which classification of test instances arises from optimizing a single objective involving both training and test instances; and structured classification, in which instances involve several interdependent classification problems. The description of structured problems also introduces useful terminology for the rest of the paper.

2.1 Transductive Classification

In supervised classification, training instances are used to induce a classifier that is then applied to individual test instances that need to be classified. In transductive classification, a single optimization problem is set up involving all training and test instances; the solution of the optimization problem yields labels for the test instances. In this way, the test instances provide evidence about the distribution of the data, which may be useful when the labeled data is limited and the distribution of unlabeled data

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 37–44,
Rochester, April 2007

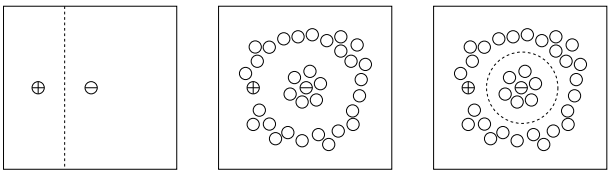


Figure 1: An example where unlabeled data helps to reveal the underlying distribution of the data points, borrowed from Sindhwani et al. (2005). The circles represent data points (unlabeled are empty, positive have a "+" and negative have a "-"). The dashed lines represent decision boundaries for a classifier. The first figure shows the labeled data and the max-margin decision boundary (we use a linear boundary to conform with Occam's razor principle). The second figure shows the unlabeled data points revealing the distribution from which the training examples were selected. This distribution suggests that a linear boundary might not be appropriate for this data. The final figure shows a more appropriate decision boundary given the distribution of the unlabeled data.

is informative about the location of the decision boundary. Figure 1 illustrates this.

2.2 Structured Classification

The usual view of structured classification is as follows. An *instance* consists of a set of classification problems in which the labels of the different problems are correlated according to a certain graphical structure. The collection of classification labels in the instance forms a single *structured label*. A typical structured problem is part of speech (POS) tagging. The parts of speech of consecutive words are strongly correlated, while the POS of words that are far away do not influence each other much. In the natural language processing tasks that motivate this work, we usually formalize this observation with a Markov assumption, implemented by breaking up the instance into *parts* consisting of pairs of consecutive words. We assign a score for each possible label of each part and then use a dynamic programming algorithm to find the highest scoring label of the entire instance.

In the rest of this paper, it will be sometimes more convenient to think of all the (labeled and unlabeled) instances of interest as forming a single joint classification problem on a large graph. In this joint problem, the atomic classification problems are linked according to the graphical structure imposed by their partition into structured classification instances. As we will see, other links between atomic problems arise in our setting that may cross between different structured instances.

2.3 Terminology

For structured problems, *instance* refers to an entire problem (for example, an entire sentence for POS tagging). A *token* refers to the smallest unit that receives a label. In POS tagging, a *token* is a word. A *part* is one or more *tokens* and is a division used by a learning algorithm. For all our experiments, a *part* is a pair of consecutive tokens, but extension to other types of parts is trivial. If two parts share a token then a consistent label for those parts has to have the same label on the shared token. For example in the sentence "I love learning ." we have parts for "I love" and "love learning". These share the token "love" and two labels for the two parts has to agree on the label for the token in order to be consistent. In all our experiments, a part is a pair of consecutive tokens so two parts are independent unless one immediately follows the other.

3 Approach

We extend the min-cut formulation of Blum and Chawla (2001) to multiple labels and structured variables by adapting a linear-programming encoding of metric labeling problems. By relaxing the linear program, we obtain an efficient approximate inference algorithm. To understand our method, it is useful to review the min-cut transductive classification algorithm (Section 3.1) as well as the metric labeling problem and its linear programming relaxation (Section 3.2). Section 3.3 describes how to encode a multi-way min-cut problem as an instance of metric labeling as well as a trivial extension that lets us introduce a bias when computing the cut.

Section 3.4 extends this formalism to structured classification.

3.1 Min-Cuts for Transductive Classification

Blum and Chawla (2001) present an efficient algorithm for semi-supervised machine learning in the unstructured binary classification setting. At a high level, the algorithm is as follows:

- Construct a graph where each instance corresponds to a vertex;
- Add weighted edges between similar vertices with weight proportional to a measure of similarity;
- Find the min-cut that separates positively and negatively labeled training instances;
- Label all instances on the positive side of the cut as positive and all others as negative.

For our purposes we need to consider two extensions to this problem: multi-way classification and constrained min-cut.

For multi-way classification, instead of computing the binary min-cut as above, we need to find the multi-way min-cut. Unfortunately, doing this in general is NP-hard, but a polynomial time approximation exists (Dahlhaus et al., 1992). In Section 3.3 we describe how we approximate this problem.

We extend this approach to structured data by constructing a graph whose vertices correspond to different parts of the instance, and add weighted edges between similar parts. We then find the multi-way min-cut that separates vertices with different labels subject to some constraints: if two parts overlap then the labels have to be consistent. Our main contribution is an algorithm that approximately computes this constrained multi-way min-cut with a linear programming relaxation.

3.2 Metric Labeling

Kleinberg and Tardos (1999) introduce the metric labeling problem as a common inference problem in a variety of fields. The inputs to the problem are a weighted graph $G = (V, E)$, a set of labels $L = \{i | i \in 1 \dots k\}$, a cost function $c(v, i)$ which represents the preference of each vertex for each possible label and a metric $d(i, j)$ between labels i and j. The goal is to assign a label to each vertex $l : V \rightarrow L$ so as to minimize the cost given by:

$$\begin{aligned} c(l) &= \sum_{v \in V} c(v, l(v)) \\ &+ \sum_{(u,v) \in E} d(l(u), l(v)) \cdot w(u, v) \ . \end{aligned} \quad (1)$$

Kleinberg and Tardos (1999) give a linear programming approximation for this problem with an approximation factor of two and explain how this can be extended to an $O(\log k)$ approximation for arbitrary metrics by creating a hierarchy of labels. Chekuri et al. (2001) present an improved linear program that incorporates arbitrary metrics directly and provides an approximation at least as good as that of Kleinberg and Tardos (1999). The idea in the new linear program is to have a variable for each edge labeling as well as one for each vertex labeling.

Following Chekuri et al. (2001), we represent the event that vertex u has label i by the variable $x(u, i)$ having the value 1; if $x(u, i) = 0$ then vertex v must have some other label. Similarly, we use the variable and value $x(u, i, v, j) = 1$ to mean that the vertices u and v (which are connected by an edge) have label i and j respectively. The edge variables allow us to encode the costs associated with violated edges in the metric labeling problem. Edge variables should agree with vertex labels, and by symmetry we should have $x(u, i, v, j) = x(v, j, u, i)$. If the linear program gives an integer solution, this is clearly the optimal solution to the original metric labeling instance. Chekuri et al. (2001) describe a rounding procedure to compute an integer solution to the LP that is guaranteed to be an approximation of the optimal integer solution. For the problems we considered, this was very rarely necessary. Their linear program relaxation is shown in Figure 2. The cost function is the sum of the vertex costs and edge costs. The first constraint requires that each vertex have a total of one labeling unit distributed over its labels, that is, we cannot assign more or less than one label per vertex. The second constraint

$$\min \quad \sum_{u \in V} \sum_{i \in L} c(u,i)x(u,i) + \sum_{(u,v) \in E} \sum_{k,j \in L} w(u,v)d(i,j)x(u,i,v,j)$$

subject to

$$\begin{aligned}
\sum_{i \in L} x(u,i) &= 1 \quad && \forall u \in V \\
x(u,i) - \sum_{j \in L} x(u,i,v,j) &= 0 \quad && \forall u \in V, v \in N(u), i \in L \\
x(u,i,v,j) - x(v,j,u,i) &= 0 \quad && \forall u,v \in V, i,j \in L \\
x(u,i,v,j), x(u,i) &\in [0,1] \quad && \forall u,v \in V, i,j \in L
\end{aligned}$$

Figure 2: The Chekuri et al. (2001) linear program used to approximate metric labeling. See text for discussion.

requires that vertex- and edge-label variables are consistent: the label that vertex variables give a vertex should agree with the labels that edge variables give that vertex. The third constraint imposes the edge-variable symmetry condition, and the final constraint requires that all the variables be in the range $[0, 1]$.

3.3 Min Cut as an Instance of Metric Labeling

Given an instance of the (multi-way) min-cut problem, we can translate it to an instance of metric labeling as follows. The underlying graph and edge weights will be the same as min-cut problem. We add vertex costs ($c(u,i)$ $\forall u \in V, i \in L$) and a label metric ($d(i,j)$ $\forall i,j \in L$). For all unlabeled vertices set the vertex cost to zero for all labels. For labeled vertices set the cost of the correct label to zero and all other labels to infinity. Finally let $d(i,j)$ be one if $i \neq j$ and zero otherwise.

The optimal solution to this instance of metric labeling will be the same as the optimal solution of the initial min cut instance: the cost of any labeling is the number of edges that link vertices with different labels, which is exactly the number of cut edges. Also by the same argument, every possible labeling will correspond to some cut and approximations of the metric labeling formulation will be approximations of the original min-cut problem.

Since the metric labeling problem allows arbitrary affinities between a vertex in the graph and possible labels for that vertex, we can trivially extend the algorithm by introducing a bias at each vertex for labels more compatible with that vertex. We use the output of a classifier to bias the cost towards agreement with the classifier. Depending on the strength of the bias, we can trade off our confidence in the performance of the min-cut algorithm against the our confidence in a fully-supervised classifier.

3.4 Extension to Structured Classification

To extend this further to structured classification we modify the Chekuri et al. (2001) linear program (Figure 2). In the structured case, we construct a vertex for every part of an instance. Since we want to find a consistent labeling for an entire instance composed of overlapping parts, we need to add some more constraints to the linear program. We want to ensure that if two vertices correspond to two overlapping parts, then they are assigned consistent labels, that is, the token shared by two parts is given the same label by both. First we add a new zero-weight edge between every pair of vertices corresponding to overlapping parts. Since its weight is zero, this edge will not affect the cost. We then add a constraint to the linear-program that the edge variables for inconsistent labelings of the new edges have a value of zero.

More formally, let $(u,i,v,j) \in \Lambda$ denote that the part u having label i is consistent with the part v having label j; if u and v do not share any tokens, then any pair of labels for those parts are consistent. Now add zero-weight edges between overlapping parts. Then the only modification to the linear program is that

$$x(u,i) - \textstyle\sum_{j \in L} x(u,i,v,j) = 0 \quad \forall u \in V, v \in N(u), i \in L$$

will become

$$x(u,i) - \textstyle\sum_{j:(u,i,v,j) \in \Lambda} x(u,i,v,j) = 0 \quad \forall u \in V, v \in N(u), i \in L\ .$$

$$\min \quad \sum_{u \in V} \sum_{i \in L} c(u,i)x(u,i) + \sum_{(u,v) \in E} \sum_{k,j \in L} w(u,v)d(i,j)x(u,i,v,j)$$

subject to

$$\begin{aligned}
\sum_{i \in L} x(u,i) &= 1 \quad && \forall u \in V \\
x(u,i) - \sum_{j:(u,i,v,j) \in \Lambda} x(u,i,v,j) &= 0 \quad && \forall u \in V, v \in N(u), i \in L \\
x(u,i,v,j) - x(v,j,u,i) &= 0 \quad && \forall\ (u,i,v,j) \in \Lambda \\
x(u,i,v,j), x(u,i) &\in [0,1] \quad && \forall u,v \in V, i,j \in L
\end{aligned}$$

Figure 3: The modified linear program used to approximate metric labeling. See text for discussion.

What this modification does is to ensure that all the mass of the edge variables between vertices u and v lies in consistent labelings for their edge. The modified linear program is shown in Figure 3. We can show that this can be encoded as a larger instance of the metric labeling problem (with roughly $|V|+|E|$ more vertices and a label set that is four times as large), but modifying the linear program directly results in a more efficient implementation. The final LP has one variable for each labeling of each edge in the graph, so we have $O(|E||L|^2)$ variables. Note that $|L|$ is the number of labelings of a pair of tokens for us – even so, computation of a single dataset took on the order of minutes using the Xpress MP package.

4 Relation to Previous work

Our work is set of extensions to the work of-Blum and Chawla (2001), which we have already described. Our extensions allow us to handle multi-class and structured data, as well as to take hints from a classifier. We can also specify a similarity metric between labels so that a cut-edge can cost different amounts depending on what partitions it spans.

Taskar et al. (2004a) describe a class of Markov networks with associative clique potentials. That is, the clique potentials always prefer that all the nodes in the clique have the same label. The inference problem in these networks is to find the assignment of labels to all nodes in the graph that maximizes the sum of the clique potentials. Their paper describes a linear programming relaxation to find (or approximate) this inference problem which is very similar to the LP formulation of Chekuri et al. (2001) when all cliques are of size 2. They generalize this to larger cliques and prove that their LP gives an integral solution when the label alphabet has size 2 (even for large cliques). For the learning problem they exploit the dual of the LP formulation and use a maximum margin objective similar to the one used by Taskar et al. (2004b). If we ignore the learning problem and focus on inference, one could view our work as inference over a Markov network created by combining a set of linear chain conditional random fields with an associative Markov network (with arbitrary structure). A direction for future work would be to train the associative Markov network either independently from the chain-structured model or jointly with it. This would be very similar to the joint inference work described in the next paragraph, and could be seen as a particular instantiation of either a non-linear conditional random field (Lafferty et al., 2001) or relational Markov network (Taskar et al., 2002).

Sutton and McCallum (2004) consider the use of linear chain CRFs augmented with extra *skip edges* which encode a probabilistic belief that the labels of two entities might be correlated. They provide experimental results on named entity recognition for e-mail messages announcing seminars, and their system achieves a 13.7% relative reduction in error on the "Speaker" field. Their work differs from ours in that they add skip edges only between identical capitalized words and only within an instance, which for them is an e-mail message. In particular, they can never have an edge between labeled and unlabeled parts. Their approach is useful for identification of personal names but less helpful for other named entity tasks where the names may not be capitalized.

Lafferty et al. (2004) show a representer theorem allowing the use of Mercer kernels with

CRFs. They use a kernel CRF with a graph kernel (Smola and Kondor, 2003) to do semi-supervised learning. For them, the graph defines an implicit representation of the data, but inference is still performed only on the (chain) structure of the CRF. By contrast, we perform inference over the whole set of examples at the same time.

Altun et al. (2006) extend the use of graph-based regularization to structured variables. Their work is in the framework of maximum margin learning for structured variables where learning is framed as an optimization problem. They modify the objective function by adding a penalty whenever two parts that are expected to have a similar label assign a different score to the same label. They show improvements of up to 5.3% on two real tasks: pitch accent prediction and optical character recognition (OCR). Unfortunately, to solve their optimization problem they have to invert an $n \times n$ matrix, where n is the number of parts in the training and testing data times the number of possible labels for each part. Because of this they are forced to train on an unrealistically small amount of data (4-40 utterances for pitch accent prediction and 10 words for OCR).

5 Experiments

We performed experiments using our approach on three different datasets using a conditional random field as the base classifier. Unless otherwise noted this was regularized using a zero-mean Gaussian prior with a variance of 1.

The first dataset is the pitch-accent prediction dataset used in semi-supervised learning by Altun et al. (2006). There are 31 real and binary features (all are encoded as real values) and only two labels. Instances correspond to an utterance and each token corresponds to a word. Altun et al. (2006) perform experiments on 4 and 40 training instances using at most 200 unlabeled instances.

The second dataset is the reference part of the Cora information extraction dataset.[1] This consists of 500 computer science research paper citations. Each token in a citation is labeled as being part of the name of an author, part of the title, part of the date or one of several other labels that we combined into a single category ("other").

The third dataset is the chunking dataset from the CoNLL 2000 (Sang and Buchholz, 2000) shared task restricted to noun phrases. The task for this dataset is, given the words in a sentence as well as automatically assigned parts of speech for these words, label each word with `B-NP` if it is the first word in a base noun phrase, `I-NP` if it is part of a base noun phrase but not the first word and `O` if it is not part of a noun phrase.

For all experiments, we let each word be a token and consider parts consisting of two consecutive tokens.

5.1 Pitch Accent Prediction

For the pitch accent prediction dataset, we used the 5-nearest neighbors of each instance according to the Euclidean distance in the original feature space to construct the graph for min-cut. Table 1 shows the results of our experiments on this data, as well as the results reported by Altun et al. (2006). The numbers in the table are per-token accuracy and each entry is the mean of 10 random train-test data selections.

For this problem, our method improves performance over the base CRF classifier (except when the training data consists of only 4 utterances), but we do not see improvements as dramatic as those observed by Altun et al. (2006). Note that even the larger dataset here is quite small – 40 utterances where each token has been annotated with a binary value.

5.2 Cora-IE

For the Cora information extraction dataset, we used the first 100 principal components of the feature space to find 5 nearest neighbors of each part. This approximation is due to the cost of comuting nearest neighbors in high dimensions. In these experiments we trained on 40 instances

[1] The Cora IE dataset has been used in Seymore et al. (1999), Peng and McCallum (2004), McCallum et al. (2000) and Han et al. (2003), among others. We obtained the dataset from `http://www.cs.umass.edu/~mccallum/data/cora-ie.tar.gz`.

Method	4:80	40:80	40:200
CRF	71.2	72.5	73.1
MinCut	69.4	74.4	74.3
STR	70.7	75.7	77.5
SVM	69.9	72.0	73.1

Table 1: Results on the pitch accent prediction task. The methods we compare are as follows. CRF is supervised CRF training. MinCut is our method with a CRF as base classifier. STR and SVM are the semi-supervised results reported in Altun et al. (2006). The experiments are 4 labeled and 80 unlabeled, 40 labeled and 80 unlabeled and 40 labeled and 200 unlabeled respectively.

Variance	10	100	1000
CRF	84.5%	84.3%	83.9%
MinCut	88.8%	89.6%	89.9%

Table 2: Accuracy on the Cora-IE dataset as a percentage of tokens correctly classified at different settings for the CRF variance. Results for training on 40 instances and testing on 80. In all cases the scores are the mean of 10 random selections of 120 instances from the set of 500 available.

and used 80 as testing data. In all cases we randomly selected training and testing instances 10 times from the total set of 500. Table 2 shows the average accuracies for the 10 repetitions, with different values for the variance of the Gaussian prior used to regularize the CRF. If we choose the optimal value for each method, our approach gives a 34.8% relative reduction in error over the CRF, and improves over it in each of the 10 random data selections, and all settings of the Guassian prior variance.

5.3 CoNLL NP-Chunking

Our results are worst for the CoNLL NP-Chunking dataset. As above, we used 10 random selections of training and test sets, and used the 100 principal components of the feature space to find 5 nearest neighbors of each part. Table 3 shows the results of our experiments. The numbers in the table are per-token

Method	20:40	40:80
CRF	87.6	90.6
MinCut(CRF)	88.2	89.6

Table 3: Results on the NP-chunking task. The table compares a CRF with our method using a CRF as a base classifier. The experiments use 20 labeled and 40 unlabeled and 40 labeled and 80 unlabeled instances.

accuracy as before. When the amount of training data is very small (20 instances) we improve slightly over the base CRF classifier, but with an increased amount of training data, the small improvement is replaced with a small loss.

6 Discussion

We have presented a new transductive algorithm for structured classification, which achieves error reductions on some real-world problems. Unfortunately, those gains are not always realized, and sometimes our approach leads to an increase in error. The main reason that our approach does not always work seems to be that our measure of similarity between different parts is very coarse. In general, finding all the pairs of parts have the same label is as difficult as finding the correct labeling of all instances, but it might be possible to use unlabeled data to learn the similarity measure.

References

Yasemin Altun, David McAllester, and Mikhail Belkin. 2006. Maximum margin semi-supervised learning for structured variables. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 33–40. MIT Press, Cambridge, MA.

Avrim Blum and Shuchi Chawla. 2001. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conf. on Machine Learning*, pages 19–26. Morgan Kaufmann, San Francisco, CA.

Chandra Chekuri, Sanjeev Khanna, Joseph Naor, and Leonid Zosin. 2001. Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Symposium on Discrete Algorithms*, pages 109–118.

E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. 1992. The complexity of multiway cuts (extended abstract). In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 241–251, New York, NY, USA. ACM Press.

H. Han, C. Giles, E. Manavoglu, H. Zha, Z. Zhang, and E. Fox. 2003. Automatic document metadata extraction using support vector machines. In *Joint Conference on Digital Libraries*.

Jon Kleinberg and Eva Tardos. 1999. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 14, Washington, DC, USA. IEEE Computer Society.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 10th International Conference on Machine Learning*, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

John Lafferty, Xiaojin Zhu, and Yan Liu. 2004. Kernel conditional random fields: representation and clique selection. In *Proceedings of the twenty-first international conference on Machine learning*, page 64, New York, NY, USA. ACM Press.

A. McCallum, K. Nigam, J. Rennie, and K. Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163.

Fuchun Peng and Andrew McCallum. 2004. Accurate information extraction from research papers using conditional random fields. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *Main Proceedings of HLT-NAACL*, pages 329–336, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Erik Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop*. Association for Computational Linguistics.

K. Seymore, A. McCallum, and R. Rosenfeld. 1999. Learning hidden markov model structure for information extraction. In *AAAI'99 Workshop on Machine Learning for Information Extraction*.

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 824–831.

Alexander Smola and Risi Kondor. 2003. Kernels and regularization on graphs. In M. Warmuth and B. Scholkopf, editors, *Proceedings of the Sixteenth Annual Conference on Learning Theory and Kernels Workshop*.

Charles Sutton and Andrew McCallum. 2004. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, July. Presented at ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields.

Ben Taskar, Abbeel Pieter, and Daphne Koller. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Annual Conference on Uncertainty in Artificial Intelligence (UAI-02)*, pages 485–492, San Francisco, CA. Morgan Kaufmann Publishers.

B. Taskar, V. Chatalbashev, and D. Koller. 2004a. Learning associative markov networks. In *Proceedings of the Twenty-First International Conference on Machine Learning (ICML)*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2004b. Max-margin markov networks. In Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484.

Latent Semantic Grammar Induction: Context, Projectivity, and Prior Distributions

Andrew M. Olney
Institute for Intelligent Systems
University of Memphis
Memphis, TN 38152
aolney@memphis.edu

Abstract

This paper presents latent semantic grammars for the unsupervised induction of English grammar. Latent semantic grammars were induced by applying singular value decomposition to n-gram by context-feature matrices. Parsing was used to evaluate performance. Experiments with context, projectivity, and prior distributions show the relative performance effects of these kinds of prior knowledge. Results show that prior distributions, projectivity, and part of speech information are not necessary to beat the right branching baseline.

1 Introduction

Unsupervised grammar induction (UGI) generates a grammar from raw text. It is an interesting problem both theoretically and practically. Theoretically, it connects to the linguistics debate on innate knowledge (Chomsky, 1957). Practically, it has the potential to supersede techniques requiring structured text, like treebanks. Finding structure in text with little or no prior knowledge is therefore a fundamental issue in the study of language.

However, UGI is still a largely unsolved problem. Recent work (Klein and Manning, 2002; Klein and Manning, 2004) has renewed interest by using a UGI model to parse sentences from the Wall Street Journal section of the Penn Treebank (WSJ). These parsing results are exciting because they demonstrate real-world applicability to English UGI. While other contemporary research in this area is promising, the case for real-world English UGI has not been as convincingly made (van Zaanen, 2000; Solan et al., 2005).

This paper weaves together two threads of inquiry. The first thread is latent semantics, which have not been previously used in UGI. The second thread is dependency-based UGI, used by Klein and Manning (2004), which nicely dovetails with our semantic approach. The combination of these threads allows some exploration of what characteristics are sufficient for UGI and what characteristics are necessary.

2 Latent semantics

Previous work has focused on syntax to the exclusion of semantics (Brill and Marcus, 1992; van Zaanen, 2000; Klein and Manning, 2002; Paskin, 2001; Klein and Manning, 2004; Solan et al., 2005). However, results from the speech recognition community show that the inclusion of latent semantic information can enhance the performance of their models (Coccaro and Jurafsky, 1998; Bellegarda, 2000; Deng and Khudanpur, 2003). Using latent semantic information to improve UGI is therefore both novel and relevant.

The latent semantic information used by the speech recognition community above is produced by latent semantic analysis (LSA), also known as latent semantic indexing (Deerwester et al., 1990; Landauer et al., 1998). LSA creates a semantic representation of both words and collections of words in a vector space, using a two part process. First,

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 45–52,
Rochester, April 2007

a term by document matrix is created in which the frequency of word w_i in document d_j is the value of cell c_{ij}. Filters may be applied during this process which eliminate undesired terms, e.g. common words. Weighting may also be applied to decrease the contributions of frequent words (Dumais, 1991). Secondly, singular value decomposition (SVD) is applied to the term by document matrix. The resulting matrix decomposition has the property that the removal of higher-order dimensions creates an optimal reduced representation of the original matrix in the least squares sense (Berry et al., 1995). Therefore, SVD performs a kind of dimensionality reduction such that words appearing in different documents can acquire similar row vector representations (Landauer and Dumais, 1997). Words can be compared by taking the cosine of their corresponding row vectors. Collections of words can likewise be compared by first adding the corresponding row vectors in each collection, then taking the cosine between the two collection vectors.

A stumbling block to incorporating LSA into UGI is that grammars are inherently ordered but LSA is not. LSA is unordered because the sum of vectors is the same regardless of the order in which they were added. The incorporation of word order into LSA has never been successfully carried out before, although there have been attempts to apply word order post-hoc to LSA (Wiemer-Hastings and Zipitria, 2001). A straightforward notion of incorporating word order into LSA is to use n-grams instead of individual words. In this way a unigram, bigram, and trigram would each have an atomic vector representation and be directly comparable.

It may seem counterintuitive that such an n-gram scheme has never been used in conjunction with LSA. Simple as this scheme may be, it quickly falls prey to memory limitations of modern day computers for computing the SVD. The standard for computing the SVD in the NLP sphere is Berry (1992)'s SVDPACK, whose single vector Lanczos recursion method with re-orthogonalization was incorporated into the BellCore LSI tools. Subsequently, either SVDPACK or the LSI tools were used by the majority of researchers in this area (Schütze, 1995; Landauer and Dumais, 1997; Landauer et al., 1998; Coccaro and Jurafsky, 1998; Foltz et al., 1998; Bellegarda, 2000; Deng and Khudanpur, 2003). Using

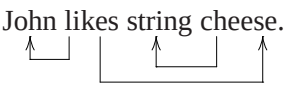


Figure 1: A Dependency Graph

the equation reported in Larsen (1998), a standard orthogonal SVD of a unigram/bigram by sentence matrix of the LSA Touchstone Applied Science Associates Corpus (Landauer et al., 1998) requires over **60 gigabytes** of random access memory. This estimate is prohibitive for all but current supercomputers.

However, it is possible to use a non-orthogonal SVD approach with significant memory savings (Cullum and Willoughby, 2002). A non-orthogonal approach creates the same matrix decomposition as traditional approaches, but the resulting memory savings allow dramatically larger matrix decompositions. Thus a non-orthongonal SVD approach is key to the inclusion of ordered latent semantics into our UGI model.

3 Dependency grammars

Dependency structures are an ideal grammar representation for evaluating UGI. Because dependency structures have no higher order nodes, e.g. *NP*, their evaluation is simple: one may compare with a reference parse and count the proportion of correct dependencies. For example, Figure 1 has three dependencies {(John, likes), (cheese, likes), (string, cheese) }, so the trial parse {(John, likes), (string, likes), (cheese, string)} has $1/3$ directed dependencies correct and $2/3$ undirected dependencies correct. This metric avoids the biases created by bracketing, where over-generation or undergeneration of brackets may cloud actual performance (Carroll et al., 2003). Dependencies are equivalent with lexicalized trees (see Figures 1 and 2) so long as the dependencies are projective. Dependencies are projective when all heads and their dependents are a contiguous sequence.

Dependencies have been used for UGI before with mixed success (Paskin, 2001; Klein and Manning, 2004). Paskin (2001) created a projective model using words, and he evaluated on WSJ. Although he reported beating the random baseline for that task, both Klein and Manning (2004) and we have repli-

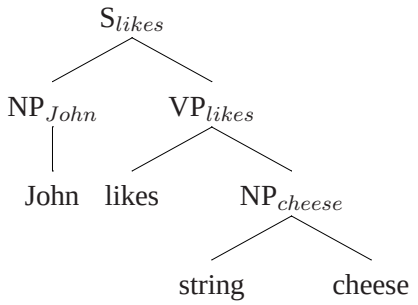


Figure 2: A Lexicalized Tree

cated the random baseline above Paskin's results. Klein and Manning (2004), on the other hand, have handily beaten a random baseline using a projective model over part of speech tags and evaluating on a subset of WSJ, WSJ10.

4 Unanswered questions

There are several unanswered questions in dependency-based English UGI. Some of these may be motivated from the Klein and Manning (2004) model, while others may be motivated from research efforts outside the UGI community. Altogether, these questions address what kinds of prior knowledge are, or are not necessary for successful UGI.

4.1 Parts of speech

Klein and Manning (2004) used part of speech tags as basic elements instead of words. Although this move can be motivated on data sparsity grounds, it is somewhat at odds with the lexicalized nature of dependency grammars. Since Paskin (2001)'s previous attempt using words as basic elements was unsuccessful, it is not clear whether parts of speech are necessary prior knowledge in this context.

4.2 Projectivity

Projectivity is an additional constraint that may not be necessary for successful UGI. English is a projective language, but other languages, such as Bulgarian, are not (Pericliev and Ilarionov, 1986). Nonprojective UGI has not previously been studied, and it is not clear how important projectivity assumptions are to English UGI. Figure 3 gives an example of a nonprojective construction: not all heads and their dependents are a contiguous sequence.

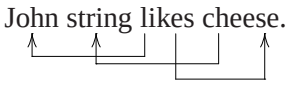


Figure 3: A Nonprojective Dependency Graph

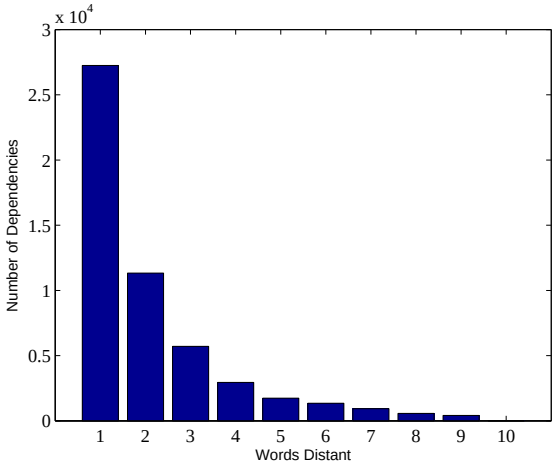


Figure 4: Distance Between Dependents in WSJ10

4.3 Context

The core of several UGI approaches is distributional analysis (Brill and Marcus, 1992; van Zaanen, 2000; Klein and Manning, 2002; Paskin, 2001; Klein and Manning, 2004; Solan et al., 2005). The key idea in such distributional analysis is that the function of a word may be known if it can be substituted for another word (Harris, 1954). If so, both words have the same function. Substitutability must be defined over a context. In UGI, this context has typically been the preceding and following words of the target word. However, this notion of context has an implicit assumption of word order. This assumption is true for English, but is not true for other languages such as Latin. Therefore, it is not clear how dependent English UGI is on local linear context, e.g. preceding and following words, or whether an unordered notion of context would also be effective.

4.4 Prior distributions

Klein and Manning (2004) point their model in the right direction by initializing the probability of dependencies inversely proportional to the distance between the head and the dependent. This is a very good initialization: Figure 4 shows the actual distances for the dataset used, WSJ10.

Klein (2005) states that, "It should be emphasized that this initialization was important in getting reasonable patterns out of this model." (p. 89). However, it is not clear that this is necessarily true for all UGI models.

4.5 Semantics

Semantics have not been included in previous UGI models, despite successful application in the speech recognition community (see Section 2). However, there have been some related efforts in unsupervised part of speech induction (Schütze, 1995). These efforts have used SVD as a dimensionality reduction step between distributional analysis and clustering. Although not labelled as "semantic" this work has produced the best unsupervised part of speech induction results. Thus our last question is whether SVD can be applied to a UGI model to improve results.

5 Method

5.1 Materials

The WSJ10 dataset was used for evaluation to be comparable to previous results (Klein and Manning, 2004). WSJ10 is a subset of the Wall Street Journal section of the Penn Treebank, containing only those sentences of 10 words or less after punctuation has been removed. WSJ10 contains 7422 sentences. To counteract the data sparsity encountered by using ngrams instead of parts of speech, we used the entire WSJ and year 1994 of the North American News Text Corpus. These corpora were formatted according to the same rules as the WSJ10, split into sentences (as documents) and concatenated. The combined corpus contained roughly 10 million words and 460,000 sentences.

Dependencies, rather than the original bracketing, were used as the gold standard for parsing performance. Since the Penn Treebank does not label dependencies, it was necessary to apply rules to extract dependencies from WSJ10 (Collins, 1999).

5.2 Procedure

The first step is unsupervised latent semantic grammar induction. This was accomplished by first creating n-gram by context feature matrices, where the feature varies as per Section 4.3. The $Context_{global}$ approach uses a bigram by document matrix such that word order is eliminated. Therefore the value of $cell_{ij}$ is the number of times $ngram_i$ occurred in $document_j$. The matrix had approximate dimensions 2.2 million by 460,000.

The $Context_{local}$ approach uses a bigram by local window matrix. If there are n distinct unigrams in the corpus, the first n columns contain the counts of the words preceding a target word, and the last n columns contain the counts of the words following a target word. For example, the value of at $cell_{ij}$ is the number of times $unigram_j$ occurred before the target $ngram_i$. The value of $cell_{i(j+n)}$ is the number of times $unigram_j$ occurred after the target $ngram_i$. The matrix had approximate dimensions 2.2 million by 280,000.

After the matrices were constructed, each was transformed using SVD. Because the non-orthogonal SVD procedure requires a number of Lanczos steps approximately proportional to the square of the number of dimensions desired, the number of dimensions was limited to 100. This kept running time and storage requirements within reasonable limits, approximately 4 days and 120 gigabytes of disk storage to create each.

Next, a parsing table was constructed. For each bigram, the closest unigram neighbor, in terms of cosine, was found, cf. Brill and Marcus (1992). The neighbor, cosine to that neighbor, and cosines of the bigram's constituents to that neighbor were stored. The constituent with the highest cosine to the neighbor was considered the likely head, based on classic head test arguments (Hudson, 1987). This data was stored in a lookup table so that for each bigram the associated information may be found in constant time.

Next, the WSJ10 was parsed using the parsing table described above and a minimum spanning tree algorithm for dependency parsing (McDonald et al., 2005). Each input sentence was tokenized on whitespace and lowercased. Moving from left to right, each word was paired with all remaining words on its right. If a pair existed in the parsing table, the associated information was retrieved. This information was used to populate the fully connected graph that served as input to the minimum spanning tree algorithm. Specifically, when a pair was retrieved from the parsing table, the arc from

the stored head to the dependent was given a weight equal to the cosine between the head and the nearest unigram neighbor for that bigram pair. Likewise the arc from the dependent to the head was given a weight equal to the cosine between the dependent and the nearest unigram neighbor for that bigram pair. Thus the weight on each arc was based on the degree of substitutability between that word and the nearest unigram neighbor for the bigram pair.

If a bigram was not in the parsing table, it was given maximum weight, making that dependency maximally unlikely. After all the words in the sentence had been processed, the average of all current weights was found, and this average was used as the weight from a dummy root node to all other nodes (the dummy ROOT is further motivated in Section 5.3). Therefore all words were given equal likelihood of being the root of the sentence. The end result of this graph construction process is an n by $n+1$ matrix, where n is the number of words and there is one dummy root node. Then this graph was input to the minimum spanning tree algorithm. The output of this algorithm is a non-projective dependency tree, which was directly compared to the gold standard dependency tree, as well as the respective baselines discussed in Section 5.3.

To gauge the differential effects of projectivity and prior knowledge, the above procedure was modified in additional evaluation trials. Projectivity was incorporated by using a bottom-up algorithm (Covington, 2001). The algorithm was applied in two stages. First, it was applied using the nonprojective parse as input. By comparing the output parse to the original nonprojective parse, it is possible to identify independent words that could not be incorporated into the projective parse. In the second stage, the projective algorithm was run again on the nonprojective input, except this time the independent words were allowed to link to any other words defined by the parsing table. In other words, the first stage identifies unattached words, and the second stage "repairs" the words by finding a projective attachment for them. This method of enforcing projectivity was chosen because it makes use of the same information as the nonprojective method, but it goes a step further to enforce projectivity.

Prior distributions of dependencies, as depicted in Figure 4, were incorporated by inversely weighting graph edges by the distance between words. This modification transparently applies to both the nonprojective case and the projective case.

Figure 5: Right Branching Baseline

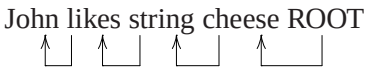


Figure 6: Left Branching Baseline

5.3 Scoring

Two performance baselines for dependency parsing were used in this experiment, the so-called right and left branching baselines. A right branching baseline predicts that the head of each word is the word to the left, forming a chain from left to right. An example is given in Figure 5. Conversely, a left branching baseline predicts that the head of each word is the word to the right, forming a chain from right to left. An example is given in Figure 6. Although perhaps not intuitively very powerful baselines, the right and left branching baselines can be very effective for the WSJ10. For WSJ10, most heads are close to their dependents, as shown in Figure 4. For example, the percentage of dependencies with a head either immediately to the right or left is 53%. Of these neighboring heads, 17% are right branching, and 36% are left branching.

By using the sign test, the statistical significance of parsing results can be determined. The sign test is perhaps the most basic non-parametric tests and so is useful for this task because it makes no assumptions regarding the underlying distribution of data.

Consider each sentence. Every word must have exactly one head. That means that for n words, there is a $1/n$ chance of selecting the correct head (excluding self-heads and including a dummy root head). If all dependencies in a sentence are independent, then a sentence's dependencies follow a binomial distribution, with n equal to the number of words, p equal to $1/n$, and k equal to the number of correct dependencies. From this it follows that the expected number of correct dependencies per sentence is np, or 1. Thus the random baseline for nonprojective depen-

dency parsing performance is one dependency per sentence.

Using the gold standard of the WSJ10, the number of correct dependencies found by the latent semantic model can be established. The null hypothesis is that one randomly generated dependency should be correct per sentence. Suppose that r^+ sentences have more correct dependencies and r^- sentences have fewer correct dependencies (i.e. 0). Under the null hypothesis, half of the values should be above 1 and half below, so $p = 1/2$. Since signed difference is being considered, sentences with dependencies equal to 1 are excluded. The corresponding binomial distribution of the signs to calculate whether the model is better than chance is $b(n, p) = b(r^+ + r^-, 1/2)$. The corresponding p-value may be calculated using Equation 1.

$$1 - \sum_{k=0}^{r^+-1} \frac{n!}{k!(n-k)!} 1/2(1/2)^{n-k} \qquad (1)$$

This same method can be used for determining statistically significant improvement over right and left branching baselines. For each sentence, the difference between the number of correct dependencies in the candidate parse and the number of correct dependencies in the baseline may be calculated. The number of positive and negative signed differences are counted as r^+ and r^-, respectively, and the procedure for calculating statistically significant improvement is the same.

6 Results

Each model in Table 6 has significantly better performance than item above using statistical procedure described in Section 5.2. A number of observations can be drawn from this table. First, all the models outperform random and right branching baselines. This is the first time we are aware of that this has been shown with lexical items in dependency UGI. Secondly, local context outperforms global context. This is to be expected given the relatively fixed word order in English, but it is somewhat surprising that the differences between local and global are not greater. Thirdly, it is clear that the addition of prior knowledge, whether projectivity or prior distributions, improves performance. Fourthly,

Method Context/Projectivity/Prior	Dependencies Correct
Random/no/no	14.2%
Right branching	17.6%
Global/no/no	17.9%
Global/no/yes	21.0%
Global/yes/no	21.4%
Global/yes/yes	21.7%
Local/no/no	22.5%
Local/no/yes	25.7%
Local/yes/yes	26.3%
Local/yes/no	26.7%
Left branching	35.8%

Table 1: Parsing results on WSJ10

projectivity and prior distributions have little additive effect. Thus it appears that they bring to bear similar kinds of constraints.

7 Discussion

The results in Section 6 address the unanswered questions identified in Section 4, i.e. parts of speech, semantics, context, projectivity, and prior distributions.

The most salient result in Section 6 is successful UGI without part of speech tags. As far as we know, this is the first time dependency UGI has been successful without the hidden syntactic structure provided by part of speech tags. It is interesting to note that latent semantic grammars improve upon Paskin (2001), even though that model is projective. It appears that lexical semantics are the reason. Thus these results address two of the unanswered questions from Section 6 regarding parts of speech and semantics. Semantics improve dependency UGI. In fact, they improve dependency UGI so much so that parts of speech are not necessary to beat a right branching baseline.

Context has traditionally been defined locally, e.g. the preceding and following word(s). The results above indicate that a global definition of context is also effective, though not quite as highly performing as a local definition on the WSJ10. This suggests that English UGI is not dependent on local linear context, and it motivates future exploration of word-order free languages using global context. It is

also interesting to note that the differences between global and local contexts begin to disappear as projectivity and prior distributions are added. This suggests that there is a certain level of equivalence between a global context model that favors local attachments and a local context model that has no attachment bias.

Projectivity has been assumed in previous cases of English UGI (Klein and Manning, 2004; Paskin, 2001). As far as we know, this is the first time a nonprojective model has outperformed a random or right branching baseline. It is interesting that a nonprojective model can do so well when it assumes so little about the structure of a language. Even more interesting is that the addition of projectivity to the models above increases performance only slightly. It is tempting to speculate that projectivity may be something of a red herring for English dependency parsing, cf. McDonald et al. (2005).

Prior distributions have been previously assumed as well (Klein and Manning, 2004). The differential effect of prior distributions in previous work has not been clear. Our results indicate that a prior distribution will increase performance. However, as with projectivity, it is interesting how well the models perform without this prior knowledge and how slight an increase this prior knowledge gives. Overall, the prior distribution used in the evaluation is not necessary to beat the right branching baseline.

Projectivity and prior distributions have significant overlap when the prior distribution favors closer attachments. Projectivity, by forcing a head to govern a contiguous subsequence, also favors closer attachments. The results reported in Section 6 suggest that there is a great deal of overlap in the benefit provided by projectivity and the prior distribution used in the evaluation. Either one or the other produces significant benefits, but the combination is much less impressive.

It is worthwhile to reiterate the sparseness of prior knowledge contained in the basic model used in these evaluations. There are essentially four components of prior knowledge. First, the ability to create an ngram by context feature matrix. Secondly, the application of SVD to that matrix. Thirdly, the creation of a fully connected dependency graph from the post-SVD matrix. And finally, the extraction of a minimum spanning tree from this graph. Although we have not presented evaluation on word-order free languages, the basic model just described has no obvious bias against them. We expect that latent semantic grammars capture some of the universals of grammar induction. A fuller exploration and demonstration is the subject of future research.

8 Conclusion

This paper presented latent semantic grammars for the unsupervised induction of English grammar. The creation of latent semantic grammars and their application to parsing were described. Experiments with context, projectivity, and prior distributions showed the relative performance effects of these kinds of prior knowledge. Results show that assumptions of prior distributions, projectivity, and part of speech information are not necessary for this task.

References

Jerome R. Bellegarda. 2000. Large vocabulary speech recognition with multispan statistical language models. *IEEE Transactions on Speech and Audio Processing*, 8(1):76–84.

Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. 1995. Using linear algebra for intelligent information retrieval. *Society for Industrial and Applied Mathematics Review*, 37(4):573–595.

Michael W. Berry. 1992. Large scale singular value computations. *International Journal of Supercomputer Applications*, 6(1):13–49.

Eric Brill and Mitchell Marcus. 1992. Automatically acquiring phrase structure using distributional analysis. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York*, pages 155–160, Philadelphia, February 23-26. Association for Computational Linguistics.

John Carroll, Guido Minnen, and Ted Briscoe. 2003. Parser evaluation using a grammatical relation annotation scheme. In A. Abeill, editor, *Treebanks: Building and Using Syntactically Annotated Corpora*, chapter 17, pages 299–316. Kluwer, Dordrecht.

Noam Chomsky. 1957. *Syntactic Structures*. Mouton, The Hague.

Noah Coccaro and Daniel Jurafsky. 1998. Towards better integration of semantic predictors in statistical language modeling. In *Proceedings of the International Conference on Spoken Language Processing*, pages 2403–2406, Piscataway, NJ, 30th November-4th December. IEEE.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Michael A. Covington. 2001. A fundamental algorithm for dependency parsing. In John A. Miller and Jeffery W. Smith, editors, *Proceedings of the 39th Annual Association for Computing Machinery Southeast Conference*, pages 95–102, Athens, Georgia.

Jane K. Cullum and Ralph A. Willoughby. 2002. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Volume 1: Theory*. Society for Industrial and Applied Mathematics, Philadelphia.

Scott C. Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Yonggang Deng and Sanjeev Khudanpur. 2003. Latent semantic information in maximum entropy language models for conversational speech recognition. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–63, Philadelphia, May 27-June 1. Association for Computational Linguistics.

Susan Dumais. 1991. Improving the retrieval of information from external sources. *Behavior Research Methods, Instruments and Computers*, 23(2):229–236.

Peter W. Foltz, Walter Kintsch, and Thomas K. Landauer. 1998. The measurement of textual coherence with latent semantic analysis. *Discourse Processes*, 25(2&3):285–308.

Zellig Harris. 1954. Distributional structure. *Word*, 10:140–162.

Richard A. Hudson. 1987. Zwicky on heads. *Journal of Linguistics*, 23:109–132.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, July 7-12. Association for Computational Linguistics.

Dan Klein and Christopher D. Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 478–485, Philadelphia, July 21-26. Association for Computational Linguistics.

Dan Klein. 2005. *The Unsupervised Learning of Natural Language Structure*. Ph.D. thesis, Stanford University.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240.

Thomas. K. Landauer, Peter W. Foltz, and Darrell Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, 25(2&3):259–284.

Rasmus M. Larsen. 1998. Lanczos bidiagonalization with partial reorthogonalization. Technical Report DAIMI PB-357, Department of Computer Science, Aarhus University.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Philadelphia, October 6-8. Association for Computational Linguistics.

Mark A. Paskin. 2001. Grammatical bigrams. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 91–97. MIT Press, Cambridge, MA.

Vladimir Pericliev and Ilarion Ilarionov. 1986. Testing the projectivity hypothesis. In *Proceedings of the 11th International Conference on Computational Linguistics*, pages 56–58, Morristown, NJ, USA. Association for Computational Linguistics.

Hinrich Schütze. 1995. Distributional part-of-speech tagging. In *Proceedings of the 7th European Association for Computational Linguistics Conference (EACL-95)*, pages 141–149, Philadelphia, March 27-31. Association for Computational Linguistics.

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

Menno M. van Zaanen. 2000. ABL: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics*, pages 961–967, Philadelphia, July 31-August 4. Association for Computational Linguistics.

Peter Wiemer-Hastings and Iraide Zipitria. 2001. Rules for syntax, vectors for semantics. In *Proceedings of the 23rd Annual Conference of the Cognitive Science Society*, pages 1112–1117, Mahwah, NJ, August 1-4. Erlbaum.

Learning to Transform Linguistic Graphs

Valentin Jijkoun and Maarten de Rijke
ISLA, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
jijkoun,mdr@science.uva.nl

Abstract

We argue in favor of the the use of labeled directed graph to represent various types of linguistic structures, and illustrate how this allows one to view NLP tasks as graph transformations. We present a general method for learning such transformations from an annotated corpus and describe experiments with two applications of the method: identification of non-local depenencies (using Penn Treebank data) and semantic role labeling (using Proposition Bank data).

1 Introduction

Availability of linguistically annotated corpora such as the Penn Treebank (Bies et al., 1995), Proposition Bank (Palmer et al., 2005), and FrameNet (Johnson et al., 2003) has stimulated much research on methods for automatic syntactic and semantic analysis of text. Rich annotations of corpora has allowed for the development of techniques for recovering deep linguistic structures: syntactic non-local dependencies (Johnson, 2002; Hockenmaier, 2003; Dienes, 2004; Jijkoun and de Rijke, 2004) and semantic arguments (Gildea, 2001; Pradhan et al., 2005; Toutanova et al., 2005; Giuglea and Moschitti, 2006). Most state-of-the-art methods for the latter two tasks use a cascaded architecture: they employ syntactic parsers and re-cast the corresponding tasks as pattern matching (Johnson, 2002) or classification (Pradhan et al., 2005) problems. Other methods (Jijkoun and de Rijke, 2004) use combinations of pattern matching and classification.

The method presented in this paper belongs to the latter category. Specifically, we propose (1) to use a flexible and expressive graph-based representation of linguistic structures at different levels; and (2) to view NLP tasks as graph transformation problems: namely, problems of transforming graphs of one type into graphs of another type. An example of such a transformation is adding a level of the predicate argument structure or semantic arguments to syntactically annotated sentences. Furthermore, we describe a general method to automatically learn such transformations from annotated corpora. Our method combines pattern matching on graphs and machine learning (classification) and can be viewed as an extension of the Transformation-Based Learning paradigm (Brill, 1995). After describing the method for learning graph transformations we demonstrate its applicability on two tasks: identification of non-local dependencies (using Penn Treebank data) and semantic roles labeling (using Proposition Bank data).

The paper is organized as follows. In Section 2 we give our motivations for using graphs to encode linguistic data. In Section 3 we describe our method for learning graph transformations and in Section 4 we report on experiments with applications of our method. We conclude in Section 5.

2 Graphs for linguistic structures and language processing tasks

Trees and graphs are natural and common ways of encoding linguistic information, in particular, syn-

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 53–60,
Rochester, April 2007

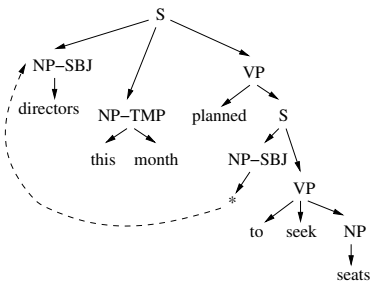


Figure 1: Local and non-local syntantic relations.

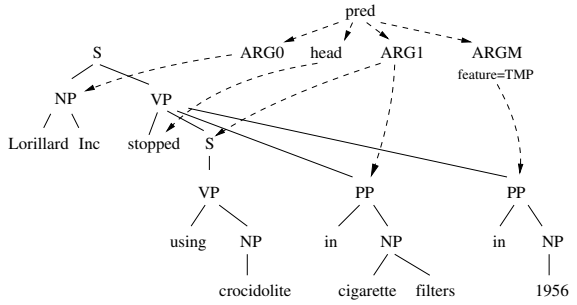


Figure 2: Syntactic structure and semantic roles.

tactic structures (phrase trees, dependency structures). In this paper we use node- and edge-labeled directed graphs as our representational formalism. Figures 1 and 2 give informal examples of such representations.

Figure 1 shows a graph encoding of the Penn Treebank annotation of the local (solid edges) and non-local (dashed edges) syntantic structure of the sentence *directors this month planned to seek more seats*. In this example, the co-indexing-based implicit annotation of the non-local dependency (subject control) in the Penn Treebank (Bies et al., 1995) is made explicit in the graph-based encoding.

Figure 2 shows a graph encoding of linguistic structures for the sentence *Lorillard Inc stopped using crocodolite in sigarette filters in 1956*. Here, solid lines correspond to surface syntactic structure, produced by Charniak's parser (Charniak, 2000), and dashed lines are an encoding of the Proposition Bank annotation of the semantic roles with respect to the verb *stopped*.

Graph-based representations allow for a uniform view on the linguistic structures on different layers. An advantage of such a uniform view is that apparently different NLP tasks can be considered as

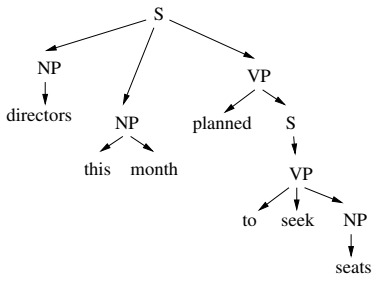


Figure 3: Output of a syntactic parser.

manipulations with graphs, in other words, as graph transformation problems.

Consider the task of recovering non-local dependencies (such as control, WH-extraction, topicalization) in the surface syntactic phrase trees produced by the state-of-the-art parser of (Charniak, 2000). Figure 3 shows a graph-based encoding of the output of the parser, and the task in question would consist in transforming the graph in Figure 3 into the graph in Figure 1. We notice that this transformation can be realised as a sequence of independent and relatively simple graph transformations: adding nodes and edges to the graph or changing their labels (e.g., from NP to NP-SBJ).

Similarly, for the example in Figure 2, adding a semantic layer (dashed edges) to the syntactic structure can also be seen as transforming a graph.

In general, we can view NLP tasks as adding additional linguistic information to text, based on the information already present: e.g., syntactic parsing taking part-of-speech tagged sentences as input (Collins, 1999), or anaphora resolution taking sequences of syntactically analysed and named-entity-tagged sentences. If both input and output linguistic structures are encoded as graphs, such NLP tasks become graph transformation problems.

In the next section we describe our general method for learning graph transformations from an annotated corpus.

3 Learning graph transformations

We start with a few basic definitions. Similar to (Schürr, 1997), we define ıemphgraph as a relational structure, i.e., a set of objects and relations between them; we represent such structures as sets of first-order logic atomic predicates defining nodes,

directed edges and their attributes (labels). Constants used in the predicates represent *objects* (nodes and edges) of graphs, as well as attribute names and values. Atomic predicates $\mathsf{node}(\cdot)$, $\mathsf{edge}(\cdot,\cdot,\cdot)$ and $\mathsf{attr}(\cdot,\cdot,\cdot)$ define nodes, edges and their attributes. We refer to (Schürr, 1997; Jijkoun, 2006) for formal definitions and only illustrate these concepts with an example. The following set of predicates:

$$\mathsf{node}(n_1), \mathsf{node}(n_2), \mathsf{edge}(e, n_1, n_2),$$
$$\mathsf{attr}(n_1, \mathsf{label}, \mathsf{Src}), \mathsf{attr}(n_2, \mathsf{label}, \mathsf{Dst})$$

defines a graph with two nodes, n_1 and n_2, having labels Src and Dst (encoded as attributes named label), and an (unlabelled) edge e going from n_1 to n_2.

A *pattern* is an arbitrary graph and an *occurence* of a pattern P in graph G is a total injective homomorphism Ω from P to G, i.e., a mapping that associates each object of P with one object G and preserves the graph structure (relations between nodes, edges, attribute names and values). We will also use the term *occurence* to refer to the graph $\Omega(P)$, a subgraph of G, the image of the mapping Ω on P.

A *graph rewrite rule* is a triple $r = \langle lhs_r, C_r, rhs_r \rangle$: the left-hand side, the constraint and the right-hand side of r, respectively, where lhs_r and rhs_r are graphs and C_r is a function that returns 0 or 1 given a graph G, pattern lhs_r and its occurence in G (i.e., C_r specifies a constraint on occurences of a pattern in a graph).

To *apply* a rewrite rule $r = \langle lhs_r, C_r, rhs_r \rangle$ to a graph G means finding all occurences of lhs_r in G for which C_r evaluates to 1, and replacing such occurences of lhs_r with occurences of rhs_r. Effectively, objects and relations present in lhs_r but not in rhs_r will be removed from G, objects and relations in rhs_r but not in lhs_r will be added to G, and common objects and relations will remain intact. Again, we refer to (Jijkoun, 2006) for formal definitions.

As will be discussed below, our method for learning graph transformations is based on the ability to compare pairs of graphs, identifying where the two graphs are similar and where they differ. An *alignment* of two graphs is a partial one-to-one homomorphism between their nodes and edges, such that if two edges of the two graphs are aligned, their respective endpoints are aligned as well. A *maximal alignment* of two graphs is an alignment that maximizes the sum of (1) the number of aligned objects (nodes and edges), and (2) the number of matching attribute values of all aligned objects. In other words, a maximal alignment identifies as many similarities between two graphs as possible. Given an alignment of two graphs, it is possible to extract a list of rewrite rules that can transform one graph into another. For a maximal alignment such a list will consist of rules with the smallest possible left- and right-hand sides. See (Jijkoun, 2006) for details.

As stated above, we view NLP applications as graph transformation modules. Our supervised method for learning graph transformation requires two corpora: input graphs $In = \{In_k\}$ and corresponding output graphs $Out = \{Out_k\}$, such that Out_k is the desired output of the NLP module on the input In_k.

The result of the method is an ordered list of graph rewrite rules $R = \langle r_1, \ldots r_n \rangle$, that can be applied in sequence to input graphs to produce the output of the NLP module.

Our method for learning graph transformations follows the structure of Transformation-Based Learning (Brill, 1995) and proceeds iteratively, as shown in Figure 4. At each iteration, we compare and align pairs of input and output graphs, identify possible rewrite rules and select rules with the most frequent left-hand sides. For each selected rewrite rule r, we extract all occurences of its left-hand side and use them to train a two-class classifier implementing the constraint C_r: the classifier, given an encoding of an occurence of the left-hand side predicts whether this particular occurence should be replaced with the corresponding right-hand side. When encoding an occurence as a feature vector, we add as features all paths and all attributes of nodes and edges in the one-edge neighborhood from the nodes of the occurence. For the experiments described in this paper we used the SVM Light classifier (Joachims, 1999) with a standard linear kernel. See (Jijkoun, 2006) for details.

4 Applications

Having presented a general method for learning graph transformations, we now illustrate the method at work and describe two applications to concrete

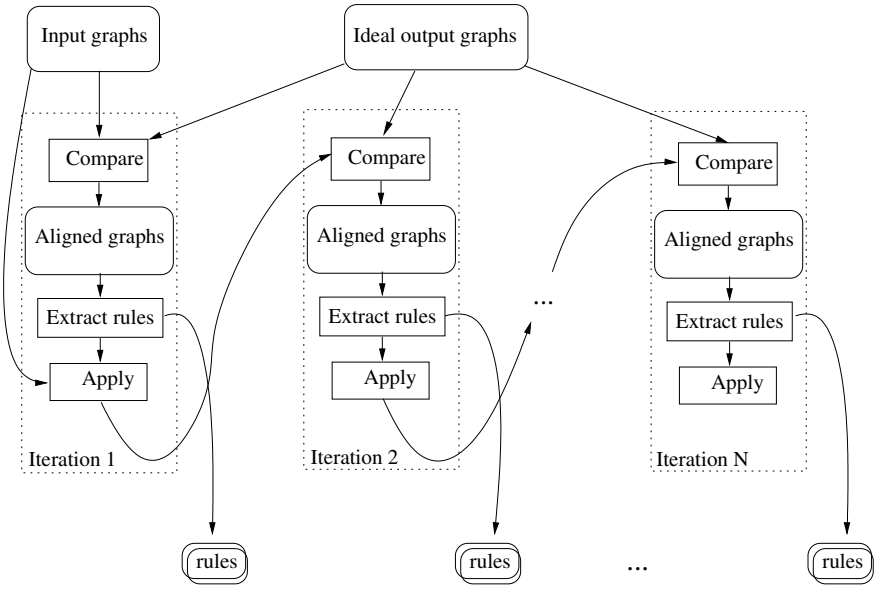


Figure 4: Structure of our method for learning graph transformations.

NLP problems: identification of non-local dependencies (with the Penn Treebank data) and semantic role labeling (with the Proposition Bank data).

4.1 Non-local dependencies

State-of-the-art statistical phrase structure parsers, e.g., Charniak's and Collins' parsers trained on the Penn Treebank, produce syntactic parse trees with bare phrase labels, (NP, PP, S, see Figure 3), i.e., providing surface grammatical analysis of sentences, even though the training corpus, the Penn Treebank, is richer and contains additional grammatical and semantic information: it distinguishes various types of modifiers, complements, subjects, objects and annotates non-local dependencies, i.e., relations between phrases not adjacent in the parse tree (see Figure 1). The task of recovering this information in the parser's output has received a good deal of attention. (Campbell, 2004) presents a rule-based algorithm for empty node identification in syntactic trees, competitive with the machine learning methods we mention next. In (Johnson, 2002) a simple pattern-matching algorithm was proposed for inserting empty nodes into syntactic trees, with patterns extracted from the Penn Treebank. (Dienes, 2004) used a preprocessor that identified surface location of empty nodes and a syntactic parser incorporating non-local dependencies into its probabilistic model. (Jijkoun and de Rijke, 2004) described an extension of the pattern-matching method with a classifier trained on the dependency graphs derived from the Penn Treebank data.

In order to apply our graph transformation method to the task of identifying non-local dependencies, we need to encode the information provided in the Penn Treebank annotations and in the output of a syntactic parser using directed labeled graphs. We used a straightforward encoding of syntactic trees, with nodes representing terminals and non-terminals and edges defining the parent-child relationship. For each node, we used the attribute type to specify whether it is a terminal or a non-terminal. Terminals corresponding to Penn empty nodes were marked with the attribute empty $= 1$. For each terminal (i.e., each word), the values of attributes pos, word and lemma provided the part-of-speech tag, the actual form and the lemma of the word. For non-terminals, the attribute label contained the label of the corresponding syntactic phrase. The co-indexing of empty nodes and non-terminals used in the Penn Treebank to annotate non-local dependencies was encoded using explicit edges with a distinct type attribute, connecting empty nodes with their antecedents (e.g., the dashed edge in Figure 1). For each non-terminal node, its head child was marked by attaching attribute head with value 1 to the corre-

sponding parent-child edge, and the lexical head of each non-terminal was explicitly indicated using additional edges with the attribute type = lexhead. We used a heuristic method of (Collins, 1999) for head identification.

When Penn Treebank sentences and the output of the parser are encoded as directed labeled graphs as described above, the task of identifying non-local dependencies can be formulated as transforming phrase structure graphs produced by a parser into graphs of the type used in Penn Treebank annotations.

We parsed the strings of the Penn Treebank with Charniak's parser and then used the data from sections 02–21 of the Penn Treebank for training: encoding of the parser's output was used as the corpus of input graphs for our learning method, and the encoding of the original Penn annotations was used as the corpus of output graphs. Similarly, we used the data of sections 00–01 for development and section 23 for testing. Using the input and output corpora, we ran the learning method as described above, at each iteration considering 20 most frequent left-hand sides of rewrite rules. At each iteration, the learned rewrite rules were applied to the current training and development corpora to create a corpus of input graphs for the next iteration (see Figure 4) and to estimate the performance of the system at the current iteration. The system was evaluated on the development corpus with respect to non-local dependencies using the "strict" evaluation measure of (Johnson, 2002): the F_1 score of precision and recall of correctly identified empty nodes and antecedents. If the absolute improvement of the F_1 score for the evaluation measure was smaller than 0.1, the learning cycle was terminated, otherwise a new iteration was started.

The learning cycle terminated after 12 iterations. The resulting sequence of $12 \times 20 = 240$ graph rewrite rules was applied to the test corpus of input graphs: Charniak's parser output on the strings of section 23 of the Penn Treebank. The result was evaluated against the original annotations of the Penn Treebank.

The results of the evaluation of the system on empty nodes and non-local dependencies and the PARSEVAL F_1 score on local syntactic phrase structure against the test corpus at each iteration are shown in Table 1.

Stage	P	R	F_1	PARSEVAL F_1
Initial	0.0	0.0	0.0	88.7
1	88.2	38.6	53.7	88.4
2	87.2	48.6	62.5	88.4
3	87.5	51.9	65.2	88.4
4	86.7	52.1	65.1	88.4
5	86.1	56.3	68.1	88.3
6	86.0	57.2	68.7	88.4
7	86.3	61.3	71.7	88.4
8	86.6	63.4	73.2	88.4
9	86.7	64.6	74.0	88.4
10	86.7	64.9	74.2	88.4
11	86.6	65.1	74.3	88.4
12	86.7	65.2	74.4	88.4

Table 1: Evaluation of our method for identification of empty nodes and their antecedents (12 first iterations).

As one can expect, at each iteration the method extracts graph rewrite rules that introduce empty nodes and non-local relations into syntactic structures, increasing the recall. The performance of the final system (P/R/F_1 = 86.7/65.2/74.4) for the task of identifying non-local dependencies is comparable to the performance of the best model of (Dienes, 2004): P/R/F_1=82.5/70.1/75.8. The PARSEVAL score for the present system (88.4) is, however, higher than the 87.3 for the system of Dienes.

Another effect of the learned transformations is changing node labels of non-terminals, specifically, modifying labels to include Penn functional tags (e.g., changing NP in the input graph in Figure 3 to NP-SBJ in the output graph in Figure 1). In fact, 17% of all learned rewrite rules involved only changing labels of non-terminal nodes. Analysis of the results showed that the system is capable of assigning Penn function tags to constituents produced by Charniak's parser with $F_1 = 91.4$ (we use here the evaluation measure of (Blaheta, 2004): the F_1 score of the precision and recall for assigning function tags to constituents with surface spans correctly identified by Charniak's parser). Comparison to the evaluation results of the function tagging method presented in (Blaheta, 2004) is shown in Table 2.

The present system outperforms the system of Blaheta on semantic tags such as -TMP or -MNR marking temporal and manner adjuncts, respectively, but performs worse on syntactic tags such as -SBJ or -PRD marking subjects and predicatives,

Type	Count	(Blaheta, 2004) P / R / F_1	Here P / R / F_1
All tags	8480	-	93.3 / 89.6 / 91.4
Syntactic	4917	**96.5** / 95.3 / **95.9**	95.4 / **95.5** / 95.5
Semantic	3225	86.7 / 80.3 / 83.4	**89.7** / **82.5** / **86.0**

Table 2: Evaluation of adding Penn Treebank function tags.

respectively. Note that the present method was not specifically designed to add functional tags to constituent labels. The method is not even "aware" that functional tags exists: it simply treats NP and NP-SBJ as different labels and tries to correct labels comparing input and output graphs in the training corpora.

In general, of the 240 graph rewrite rules extracted during the 12 iterations of the method, 25% involved only one graph node in the left-hand side, 16% two nodes, 12% three nodes, etc. The two most complicated extracted rewrite rules involved left-hand sides with ten nodes.

We now switch to the second application of our graph transformation method.

4.2 Semantic role labeling

Put very broadly, the task of semantic role labeling consists in detecting and labeling simple predicates: *Who did what to whom, where, when, how, why*, etc. There is no single definition of a universal set of semantic roles and moreover, different NLP applications may require different specificity of role labels. In this section we apply the graph transformation method to the task of identification of semantic roles as annotated in the Proposition Bank (Palmer et al., 2005), PropBank for short. In PropBank, for all verbs (except copular) of the syntactically annotated sentences of the Wall Street Journal section of the Penn Treebank, semantic arguments are marked using references to the syntactic constituents of the Penn Treebank. For the 49,208 syntactically annotated sentences of the Penn Treebank, the PropBank annotated 112,917 verb predicates (2.3 predicates per sentence on average), with a total of 292,815 semantic arguments (2.6 arguments per predicate on average).

PropBank does not aim at cross-verb semantically consistent labeling of arguments, but rather at annotating the different ways arguments of a verb can be realized syntactically in the corpus, which resulted in the choice of theory-neutral numbered labels (e.g., *Arg0, Arg1,* etc.) for semantic arguments. Figure 2 shows an example of a PropBank annotation (dashed edges).

In this section we address a specific NLP task: identifying and labeling semantic arguments in the output of a syntactic parser. For the example in Figure 2 this task corresponds to adding "semantic" nodes and edges to the syntactic tree.

As before, in order to apply our graph transformation method, we need to encode the available information using graphs. Our encoding of syntactic phrase structure is the same as in Section 4.1 and the encoding of the semantic annotations of PropBank is straightforward. For each PropBank predicate, a new node with attributes type = propbank and label = pred is added. Another node with label = head and nodes for all semantic arguments of the predicate (with labels indicating PropBank argument names) are added and connected to the predicate node. Argument nodes with label ARGM (adjunct) additionally have a feature attribute with values TMP, LOC, etc., as specified in PropBank. The head node and all argument nodes are linked to their respective syntactic constituents, as specified in the PropBank annotation. All introduced semantic edges are marked with the attribute type = propbank.

As before, we used section 02–21 of the PropBank (which annotates the same text as the Penn Treebank) to train our graph transformation system, section 00-01 for development and section 23 for testing. We ran three experiments, taking three different corpora of input graphs:

1. the original syntactic structures of the Penn Treebank containing function tags, empty nodes, non-local dependencies, etc.;

2. the output of Charniak's parser (i.e., bare syntactic trees) on the strings of sections 02–21; and

3. the output of Charniak's parser processed with the graph transformation system described in 4.1.

For all three experiments we used the gold standard syntactic and semantic annotations from the

Iter.	Penn Treebank P	Penn Treebank R	Charniak P	Charniak R	Charniak + P	Charniak + R
1	90.0	70.7	79.5	58.6	79.9	59.1
2	90.7	76.5	81.2	63.9	81.0	64.2
3	90.7	78.1	81.3	65.6	81.1	65.8
4	90.6	78.9	81.4	66.5	81.2	66.7
5	90.5	80.4	81.4	67.0	81.2	68.3
6	90.4	81.2	81.4	68.3	81.1	68.8
7	90.3	81.9	81.3	68.9	81.0	69.3
8	90.3	82.2	81.3	69.3	81.0	69.8
9	90.3	82.5	81.3	69.6	81.0	70.1
10	90.3	82.8	81.4	69.8	81.0	70.3
11	90.3	83.0	81.3	69.9	81.0	70.4
12	90.3	83.2				

Table 3: Evaluation of our method for semantic role identification with Propbank: with Charniak parses and with parses processed by the system of Section 4.1.

Penn Treebank and PropBank as the corpora of output graphs (for the experiment with bare Charniak parses, we dropped function tags, empty nodes and non-local dependencies from the syntactic annotation of the output graphs: we did not want our system to start recovering these annotations, but were interested in the identification of PropBank information alone).

For each of the experiments, we used the corpora of input and output graphs as before, at each iteration extracting 20 rewrite rules with most frequent left-hand sides, applying the rules to the development data to measure the current performance of the system. We stopped the learning in case the performance improvement was less than a threshold and, otherwise, continued the learning loop. As our performance measure we used the F_1 score of precision and recall of the correctly identified and labeled non-empty constituents—semantic arguments.

In all experiments, the learning stopped after 11 or 12 iterations. The results of the evaluation of the system at each iteration on the test section of PropBank are shown in Table 3.

As one may expect, the performance of our semantic role labeler is substantially higher on the gold Penn Treebank syntactic structures than on the parser's output. Surprisingly, however, adding extra information to the parser's output (i.e., processing it with the system of Section 4.1) does not significantly improve the performance of the resulting system.

In Table 4 we compare our system for semantic roles labeling with the output of Charniak's parser to the state-of-the-art system of (Pradhan et al., 2005).

System	P	R	F_1
(Pradhan et al., 2005)	80.9	76.8	78.8
Here	81.0	70.4	75.3

Table 4: Evaluation of our methods for semantic role identification with Propbank (12 first iterations).

While showing good precision, our system performs worse than state-of-the-art with respect to recall. Taking into account the iterative nature of the method and imperfect rule selection criteria (we simply take the most frequent left-hand sides), we believe that it is the rule selection and learning termination condition that account for the relatively low recall values. Indeed, in all three experiments described above the learning loop stops while the recall is still on the rise, albeit very slowly. It seems that a more careful rule selection mechanism and loop termination criteria are needed to address the recall problem.

5 Conclusions

In this paper we argued that encoding diverse and complex linguistic structures as directed labeled graphs allows one to view many NLP tasks as graph transformation problems. We proposed a general method for learning graph transformation from annotated corpora and described experiments with two NLP applications.

For the task of identifying non-local dependencies and for function tagging our general method demonstrates performance similar to the state-of-the-art systems, designed specifically for these tasks. For the PropBank semantic role labeling the method shows a relatively low recall, which can be explained by our sub-optimal "rule of thumb" heuristics (such as selecting 20 most frequent rewrite rules at each iteration of the learning method). We see two ways of avoiding such heuristics. First, one can define and fine-tune the heuristics for each specific application. Second, one can use more informed rewrite rule selection methods, based on graph-based relational learning and frequent subgraph detection algorithms (Cook and Holder, 2000; Yan and Han, 2002). Furthermore, more experiments are required

to see how the details of encoding linguistic information in graphs affect the performance of the method.

Acknowledgements

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 017.001.190, 220-80-001, 264-70-050, 354-20-005, 600.065.120, 612-13-001, 612.000.106, 612.066.302, 612.069.006, 640.001.501, 640.002.501, and by the E.U. IST programme of the 6th FP for RTD under project MultiMATCH contract IST-033104.

References

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II style Penn Treebank project. Technical report, University of Pennsylvania.

Don Blaheta. 2004. *Function Tagging*. Ph.D. thesis, Brown University.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4):543–565.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 645–653.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of NAACL*, pages 132–139.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Diane J. Cook and Lawrence B. Holder. 2000. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41.

Péter Dienes. 2004. *Statistical Parsing with Non-local Dependencies*. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany.

Daniel Gildea. 2001. *Statistical Language Understanding Using Frame Semantics*. Ph.D. thesis, University of California, Berkeley.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Semantic role labeling via framenet, verbnet and propbank. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 929–936.

Julia Hockenmaier. 2003. Parsing with generative models of predicate-argument structure. In *Proceedings of the 41st Meeting of ACL*, pages 359–366.

Valentin Jijkoun and Maarten de Rijke. 2004. Enriching the output of a parser using memory-based learning. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 311–318, Barcelona, Spain, July.

Valentin Jijkoun. 2006. *Graph Transformations for Natural Language Processing*. Ph.D. thesis, University of Amsterdam.

Thorsten Joachims. 1999. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press.

Christopher R. Johnson, Miriam R. L. Petruck, Collin F. Baker, Michael Ellsworth, Josef Ruppenhofer, and Charles J. Fillmore. 2003. FrameNet: Theory and Practice. `http://www.icsi.berkeley.edu/~framenet`.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of the 40th meeting of ACL*, pages 136–143.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Sameer Pradhan, Wayne Ward, Kadri Hacioglu, Jim Martin, and Dan Jurafsky. 2005. Semantic role labeling using different syntactic views. In *Proceedings of ACL-2005*.

A. Schürr. 1997. Programmed graph replacement systems. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, chapter 7, pages 479–546.

Kristina Toutanova, Aria Haghighi, and Chris Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics (ACL)*.

Xifeng Yan and Jiawei Han. 2002. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*.

Semi-supervised Algorithm for Human-Computer Dialogue Mining

Calkin S. Montero and **Kenji Araki**
Graduate School of Information Science and Technology
Hokkaido University, Kita 14-jo Nishi 9-chome
Kita-ku, Sapporo 060-0814 Japan
{calkin,araki}@media.eng.hokudai.ac.jp

Abstract

This paper describes the analysis of weak local coherence utterances during human-computer conversation through the application of an emergent data mining technique, *data crystallization*. Results reveal that by adding utterances with weak local relevance the performance of a baseline conversational partner, in terms of user satisfaction, showed betterment.

1 Introduction

Data mining can be defined as the process of finding new and potentially useful knowledge from data. An enhanced trend of data mining is *chance discovery*, which in spite of being an emergent field of research has been applied to different branches of science. Recently, data crystallization (Ohsawa, 2005) has been proposed as a chance discovery extension devoted to find desired unobservable events within a given data. This method of data mining has been intended for revealing events that are significant but are *not* included in the analyzed data (sparseness).

The knowledge database of a conversational partner computer program is a good example of data sparseness, making very difficult to realize a human-like conversation between a user and a computer, since the database is missing relevant or coherent answers for a given user input, e.g., ELIZA system (Weizenbaum, 1966), corpus-based dialogue systems (Inui et al., 2003), and so forth. Herring (1999) noticed the weak sequential coherence during human-computer interaction that is caused by weak local relevance, pointing out the violation of the Gricean maxim of local relevance during the dialogue. However, we argue that conversational responses - often of very weak local relevance - generated by a conversational partner computer program (or "chatbot") could successfully simulate human-like coherent dialogue (see example given by (Herring, 1999), where topical coherence is stated in spite of violation of local relevance Gricean maxim). Moreover, we argue that weak local relevance utterances could be regarded as *topic-shifting utterances*. These utterances are vital in order to keep the flow of the dialogue and its overall coherence. Determining the moment when the topic-shifting utterances are needed is hence essential to the success of a chatbot.

In this paper KeyGraph and data crystallization techniques are applied to analyze human-computer trivial dialogue, *chat*, considering as chances to discover the *missing* topic-shifting utterances, used to improve the overall sequential coherence and the flow of the computer dialogue.

2 Human-Computer Conversation Analysis Approach

During the process of dialogue mining, in order to analyze the behavior of human-computer conversational interaction, the interrelationship of utterances of a dialogue section was obtained. The following is a fragment of an analyzed dialogue section between a user and ALICE chatbot (Wallace, 2005):

[1] User: hi

[2] Chatbot: hello there! what is your name?

[3] User: Joe

[4] Chatbot: Ok, I will call you Joe

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 61–64,
Rochester, April 2007

[5] User: what should I call you?

[6] Chatbot: my name is Alice

[7] User: cool [....]

(Conversation continues)

[17] User: what do you mean by robot? there are lots of different robots

[18] Chatbot: It is a good question. Have you tried a web search? Would you like to know more?

[19] User: sure

[20] Chatbot: Ok let's do it. What would you like to know?

[21] User: what Alice thinks a robot means

[22] Chatbot: is that a rhetorical question? I would search for it

[23] User: when I hear robot, I think of something that moves

[24] Chatbot: you never know what to expect from life. Why? oh yeah, we are talking about something that moves [....]

For the analysis, the application of data crystallization is introduced in order to identify the missing topic-shifting utterances. The interrelationship of utterances (user-chatbot) is represented in a co-occurrence document created as described below.

2.1 The *KeyGraph*

The KeyGraph has been used as a data-mining tool for extracting patterns of the appearance of chance events (Ohsawa et al.(2003)). The KeyGraph identifies relationships between terms in a document particularly focusing on co-occurrence relationships of both high-probability and low-probability events. Montero et al. (2005) have applied this tool for analyzing the dynamic behavior of human-human chat, identifying criticality.

In this paper the KeyGraph is applied in combination with data crystallization in order to visualized utterances that do not appear during human-computer chat. The interrelationship of utterances (user-chatbot) is represented in a co-occurrence document created by the following algorithm: a) Each utterance (from both, the user and the chatbot) was considered as one sentence. b) Each sentence was segmented into words. c) High frequency words were eliminated, i.e., I, you, is, follow-ups and the like, as to avoid false co-occurrence. d) A vectorial representation of each sentence (at word level) was obtained and sentences co-occurrence relationship was determined as[1]:

D= w_1:: S_1, S_2, S_4 .../ w_2:: S_9, S_{25} .../
w_3:: S_1, S_3, S_{10} .../ ... / w_n:: S_{24}, S_{25}, ... S_m

where: w_k ($k = 1, 2, 3, ..., n$), represents a word in a sentence. S_l ($l = 1, 2, 3, ..., m$), represents a sentence.

Then it could be said that the obtained D document contains the co-occurrence relationship of the utterances during the analyzed dialogue section. In the graph, the most frequent items in *D* are shown as black nodes and the most strongly co-occurring item-pairs are linked by black lines according to the Jaccard coefficient:

$$J(S_x, S_y) = p(S_x \cap S_y)/p(S_x \cup S_y)$$

where $p(S_x \cap S_y)$ is the probability that both elements S_x and S_y co-occur in a line in *D*, and $p(S_x \cup S_y)$ is the probability that either S_x or S_y appears in a line. In the graph, nodes are interpreted as sentences (from *D*) and clusters of nodes as particular topics (Figure 1).

2.2 Data Crystallization

Data crystallization (Ohsawa, 2005), is dedicated to experts working in real domains where discoveries of events that are important but are not included in the analyzed data are desired. The process of data crystallization involves to insert *dummy items* in the given data in order to represent unobservable events. In this paper, each dummy item inserted in the *D* document (one in each vector of *D*) is named X_Y, where X represents the level of the insertion and Y represents the line where the dummy item was inserted. The KeyGraph is applied to the new *D* document and all of the dummy nodes that did not appear linking clusters in the graph are eliminated from the data, and then the cycle is iterated to higher levels. In the case of the *D* document of Sec.2.1, after the first level of insertion it becomes:

D'= w_1:: S_1, S_2, S_4 ...1_1 / w_2:: S_9, S_{25} ...1_2 /
w_3:: S_1, S_3, S_{10} ...1_3 / ... / w_n:: S_{24}, S_{25}, ... S_m 1_n

where 1_o ($o = 1, 2, 3, ..., n$), represents each dummy item inserted in each vector of D.

After feeding the KeyGraph with D', all the dummy items that did not appear linking clusters as bridges in the outputted graph are deleted. At this point new dummy items with higher hierarchy (2_x) are inserted in D', and the cycle iterates. Unobservable events and their relations with other events are to be visualized by the application of KeyGraph iteratively to the data that is been crystallized (Figure 2).

[1] Since follow-ups were eliminated, the number of sentences in D might be smaller than the actual number of sentences in the dialogue.

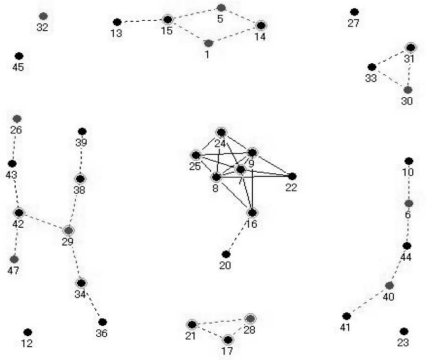


Figure 1: User-Computer Chat Graph

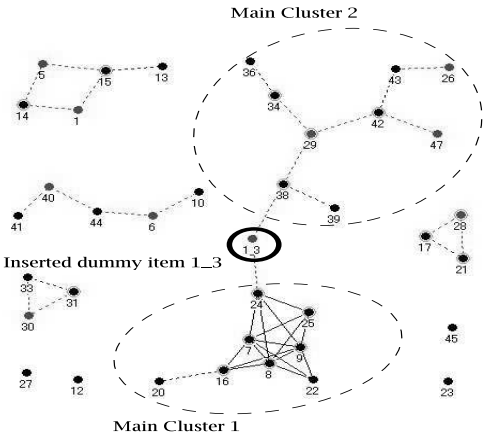


Figure 2: Crystallized Data Graph

3 Experiment and Visual Results

The performed experiment was carried out in three stages. In the first stage of the experiment, three different dialogue sections (including the one shown in Sec.2) between three native English speakers and a chatbot (Wallace, 2005) were analyzed in order to find co-occurrence between the users' utterance and the chatbot replies, i.e., D document. This D document was then examined by the KeyGraph (unsupervised process). Figure 1 shows the graphical view of the dialogue in Sec.2 (48 turns, user - chatbot, in total). A characteristic of the KeyGraph is the visualization of co-occurring events by means of clusters. In Figure 1, the nodes represent sentences from the *D* document, the clusters represent the relationship among those sentences, i.e., a specific topic, and the nodes that link the clusters represent the transition from one topic to the next. It can be observed that the main clusters are not interconnected, leading to the conclusion that the chatbot in many cases could not keep a smooth and natural flow of the dialogue.

In the second stage of the experiment, a *crystallized document* of utterance co-occurrence, i.e., D' document, was obtained for the same dialogue sections, following the process described in Sec.2.2. The graphical output of the dialogue in Sec.2, after crystallization, can be observed in Figure 2. It can be seen in this figure how the two main clusters appear to be interconnected by the dummy item *1_3*. Although this dummy item was inserted in the third line of the *D* document, it appears in the graph connecting the two main clusters. The dummy item *1_3* branches from utterance [24]. This interconnecting point can be regarded as the system considering it appropriate to insert a topic-shifting utterance at this point of the conversation. In doing so, a well interconnected graph is obtained (Figure 2). This information is valuable for making the chatbot to ask "intelligent questions" as a mean of *conversational* responses to keep the interest from the user.

In the third stage of the experiment, the information yielded by the previous analysis, i.e., regarding the timing where a topic-shifting utterance might be needed, was used to feed the chatbot database. *Topic-shifting responses* were inserted by hand (supervised process) as general patterns (around one hundred patterns) for smoothly change the topic when there is not a pattern that matches a given utterance. In this way a bridge, represented in Figure 2 by the dummy item, is created giving to the dialogue the desired smoothness. Seven users (four native English speakers, three non native speakers) were requested to perform a chat with the plain chatbot and with the enhanced chatbot (the users did not know which chatbot was plain or which was enhanced). The time set up was maximum 30 minutes-chat with each program, the user was free to stop at any time before the time limit. The evaluation of the chatbots performances was made through a ques-

	Turns	% V.R.*	Accuracy/relevance
Baseline	74	21.11%	fair
Enhanced	128	7.37%	good

V.R.* = Vague Reply (This table contains the average of Turns and VR)

Table 1: Chatbots Overall Performance

tionnaire given to the users. In the questionnaire the degree of satisfaction of the user was surveyed based on the accuracy/relevance of the chatbot responses and its ability to keep the dialogue flowing. The accuracy/relevance had a ranking scale: 1. excellent, 2. good, 3. average, 4. fair, and 5. poor, while the user satisfaction ranked : 1. very satisfied, 2. satisfied, 3. not sure, 4. dissatisfied, and 5. very dissatisfied.

The four native English speaker users globally agreed ranking enhanced chatbot as having "good" accurate/relevant responses during the overall dialogue, giving remarks like "interesting to talk with", "better at keeping the conversation going", "easier to talk to", and "more impression of consciousness". In the case of the plain chatbot, the native speakers showed "dissatisfied" at its performance, giving remarks like "loop responses", "slow in keeping the conversation going", "no so nice responses", "no constant memory". Table 1 shows a resume of the average performance of the chatbot for all of the users. An example of a vague reply is given in the following fragment:

Chatbot: What is like living in Sapporo?

User: sweet as

Chatbot: In the context of living in Sapporo I don't understand "what is like living in Sapporo?" **[vague reply]**

Two non native English speaker users ranked the enhanced chatbot as having "fair" and "average" accurate/relevant responses while the plain chatbot was ranked as having "poor" and "fair" accurate/relevant responses. The third non native English speaker user ranked both chatbots as "poor" due to "the chatbots lack of understanding deixis, and anaphor".

As a mean of discussion, in Figure 2 it could be expected that the dummy item *1_3* would branch from utterance [25] {*User: no, you asked me who is the best robot*}, which is in the same cluster with utterance [24]. However, under closer examination it becomes clear that utterance [24] has stronger co-occurrence with utterance [38] {*Chatbot: I know you are but what am I*} than utterance [25]. Hence, the algorithm suggests to link the clusters via utterance [24].

In other aspect, based on the feedback given by the seven users of the experiment, the overall performance of the enhanced chatbot can be considered better than the plain chatbot. It is worth noticing that the evaluation of the non native English speaker users tended to emphasize the grammatical aspect of the chatbots responses. On the other hand, the evaluation of the native English speaker users tended to emphasize the smoothness of the dialogue. Although there is still plenty of room for improvement and research a betterment in the chatbot performance could be seen through this approach.

4 Conclusion

In this paper the application of a novel data mining method, data crystallization, for visualizing missing topic-shifting utterances during human-computer chat has been described. Based on this information, during the experiment, the use of weak local relevance utterances, i.e., topic-shifting responses, despite of violation of Grecian maxim of local relevance, showed to meliorate the overall dialogue flow. Future research will be oriented to the extended implementation of the obtained results for enhancing the chat flow modeling of a conversational partner program.

References

Susan Herring. 1999. Interactional coherence in cmc. *Journal of Computer Mediated Communication*, 4(1).

Nobuo Inui, Takuya Koiso, Junpei Nakamura, and Yoshiyuki Kotani. 2003. Fully corpus-based natural language dialogue system. In *Natural Language Generation in Spoken and Written Dialogue, AAAI Spring Symposium*.

Yukio Ohsawa and Peter McBurney, editors. 2003. *Chance Discovery*. Springer, Berlin Heidelberg New York.

Yukio Ohsawa. 2005. Data crystallization: Chance discovery extended for dealing with unobservable events. *New Mathematics and Natural Computation*, 1(3):373–392.

Calkin S Montero and Kenji Araki. 2005. Human chat and self-organized criticality: A chance discovery application. *New Mathematics and Natural Computation*, 1(3):407–420.

Richard Wallace. 2005. A.l.i.c.e. artificial intelligence foundation. http://www.alicebot.org.

Joseph Weizenbaum. 1966. Eliza a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.

Correlations in the organization of large-scale syntactic dependency networks

Ramon Ferrer i Cancho
Departament de Física Fonamental
Universitat de Barcelona
Martí i Franquès 1, 08028 Barcelona, Spain.
ramon.ferrericancho@ub.edu

Alexander Mehler
Department of Computational Linguistics and Text Technology
Bielefeld University
D-33615 Bielefeld, Germany
Alexander.Mehler@uni-bielefeld.de

Olga Pustylnikov
Department of Computational Linguistics and Text Technology
Bielefeld University
D-33615 Bielefeld, Germany
Olga.Pustylnikov.@uni-bielefeld.de

Albert Díaz-Guilera
Departament de Física Fonamental
Universitat de Barcelona
Martí i Franquès 1, 08028 Barcelona, Spain.
albert.diaz@ub.edu

Abstract

We study the correlations in the connectivity patterns of large scale syntactic dependency networks. These networks are induced from treebanks: their vertices denote word forms which occur as nuclei of dependency trees. Their edges connect pairs of vertices if at least two instance nuclei of these vertices are linked in the dependency structure of a sentence. We examine the syntactic dependency networks of seven languages. In all these cases, we consistently obtain three findings. Firstly, clustering, i.e., the probability that two vertices which are linked to a common vertex are linked on their part, is much higher than expected by chance. Secondly, the mean clustering of vertices decreases with their degree — this finding suggests the presence of a hierarchical network organization. Thirdly, the mean degree of the nearest neighbors of a vertex x tends to decrease as the degree of x grows — this finding indicates disassortative mixing in the sense that links tend to connect vertices of dissimilar degrees. Our results indicate the existence of common patterns in the large scale organization of syntactic dependency networks.

1 Introduction

During the last decade, the study of the statistical properties of networks as different as technical, biological and social networks has grown tremendously. See (Barabási and Albert, 2002; Dorogovtsev and Mendes, 2002; Newman, 2003) for a review. Among them many kinds of linguistic networks have been studied: e.g., free word association networks (Steyvers and Tenenbaum, 2005), syllable networks (Soares et al., 2005), thesaurus networks (Sigman

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 65–72,
Rochester, April 2007

and Cecchi, 2002), and document networks (Mehler, 2006). See (Mehler, 2007a) for a review of linguistic network studies. Here we focus on the so called *global syntactic dependency networks* (GSDN) (Ferrer i Cancho et al., 2004; Ferrer i Cancho, 2005). A GSDN is induced from a dependency treebank in two steps:

1. The vertices of the network are obtained from the word forms appearing as nuclei in the input treebank and from punctuation marks as far as they have been annotated and mapped onto dependency trees. The notion of a nucleus is adapted from Lucien Tesnière: a nucleus is a node of a dependency tree. Note that multipart nuclei may also occur. We use the term *type* in order to denote word forms and punctuation marks. The reason that we induce vertices from types, but not from lexemes, is that not all corpora are lemmatized. Thus, the type level is the *least common denominator* which allows comparing the different networks. Note also that a systematization of the corpora with respect to the inclusion of punctuation marks is needed.

2. Two vertices (i.e. types) of a GSDN are connected if there is at least one dependency tree in which their corresponding instance nuclei are linked. When it comes to applying the apparatus of complex network theory, the arc direction is generally disregarded (Newman, 2003). Thus, GSDNs are *simple undirected* graphs without loops or multiple edges.

The attribute 'global' distinguishes *macroscopic* syntactic dependency networks from their *microscopic* counterparts in the form of syntactic dependency structures of single sentences. The latter are the usual object of dependency grammars and related formalisms. The goal of this article is to shed light on the large-scale organization of syntactic dependency structures. In terms of theoretical linguistics, we aim to determine the statistical properties that are common to all languages (if they exist), the ones that are not and to explain our findings. To achieve this goal, we must overcome the limits of many studies of linguistic networks. Firstly, by using GSDNs we intend to solve the problems of co-occurrence networks in which words are linked if they (a) are adjacent, (b) co-occur within a short window (Ferrer i Cancho and Solé, 2001; Milo et al., 2004; Antiqueira et al., 2006; Masucci and Rodgers, 2006) or (c) appear in the same sentence (Caldeira et al., 2006). This approach is problematic: with a couple of exceptions (Bordag et al., 2003; Ferrer i Cancho and Solé, 2001), no attempt is made to filter out statistically insignificant co-occurrences. Unfortunately the filter used in (Ferrer i Cancho and Solé, 2001) is not well-defined because it does not consider fluctuations of the frequencies of word co-occurrences. (Bordag et al., 2003) implement a collocation measure based on the Poisson distribution and, thus, induce *collocation* instead of *co-occurrence* networks. However, the notion of a sentence window and related notions are problematic as the probability that two words depend syntactically decays exponentially with the number of intermediate words (Ferrer i Cancho, 2004). Further, (Ferrer i Cancho et al., 2004) shows that the proportion of syntactically wrong links captured from a sentence by linking adjacent words is about 0.3 while this proportion is about 0.5 when linking a word to its 1st and 2nd neighbors. Thus, dependency treebanks offer connections between words that are *linguistically* precise according to a dependency grammar formalism. Secondly, the majority of linguistic network studies is performed on English only — with some exceptions (Soares et al., 2005; Ferrer i Cancho et al., 2004; Mehler, 2006). Concerning GSDNs, (Ferrer i Cancho et al., 2004) considers three languages but the syntactic dependency information of sentences is systematically incomplete in two of them. Here we aim to use complete treebanks and analyze more (i.e. seven) languages so that we can obtain stronger conclusions about the common statistical patterns of GSDNs than in (Ferrer i Cancho et al., 2004).

Therefore, this article is about statistical regularities of the organization of GSDNs. These networks are analyzed with the help of complex network theory and, thus by means of quantitative graph theory. We hypothesize that GSDNs are homogeneous in terms of their network characteristics while they differ from *non-syntactic* networks. The long-term objective to analyze such distinctive features is to explore *quality criteria* of dependency treebanks which allow separating high quality annota-

tions from erroneous ones.

The remainder of this article is organized as follows: Section 2 introduces the statistical measures that will be used for studying GSDNs of seven languages. Section 3 presents the treebanks and their unified representations from which we induce these networks. Section 4 shows the results and Section 5 discusses them.

2 The statistical measures

Two essential properties of a network are N, the number of vertices (i.e. the number of types), and $\bar{k}$ the mean vertex degree (Barabási and Albert, 2002). The literature about distinctive indices and distributions of complex networks is huge. Here we focus on correlations in the network structure (Serrano et al., 2006). The reason is that correlation analysis provides a deeper understanding of network organization compared to classical aggregative "small-world" indices. For instance, two networks may have the same degree distribution (whose similarity is measured by the exponent of power laws fitted to them) while they differ in the degree correlation of the vertices forming a link. Correlation analysis is performed as follows: We define $p(k)$ as the proportion of vertices with degree k. Here we study three measures of correlation (Serrano et al., 2006):

- $\bar{k}_{nn}(k)$ is the average degree of the nearest neighbors of the vertices with degree k (Pastor-Satorras et al., 2001). If $\bar{k}_{nn}(k)$ tends to grow as k grows the network is said to exhibit assortative mixing. In this case, edges tend to connect vertices of similar degree. If in contrast to this $\bar{k}_{nn}(k)$ tends to shrink as k grows, the network is said to exhibit disassortative mixing. In this case, edges tend to connect vertices of dissimilar degree. If there are no correlations, then $\bar{k}_{nn}(k) = \kappa$ with $\kappa = \langle k^2 \rangle / \langle k \rangle$; $\langle k \rangle = \bar{k}$ is the 1st and $\langle k^2 \rangle$ the 2nd moment of the degree distribution, namely

$$\langle k \rangle = \sum_{k=1}^{N-1} k p(k) \quad (1)$$

$$\left\langle k^2 \right\rangle = \sum_{k=1}^{N-1} k^2 p(k). \quad (2)$$

  In order to enable comparisons of different networks, $\bar{k}_{nn}(k)$ is normalized using κ and replaced by $\bar{k}_{nn}(k)/\kappa$.

- $\bar{c}(k)$ is the mean clustering coefficient of vertices of degree k. The clustering coefficient of a vertex is defined as the proportion of pairs of adjacent vertices (u, v) such that u and v are linked.

- $\bar{c}$ is the mean clustering coefficient defined as

$$\bar{c} = \sum_{k=1}^{N-1} p(k)\bar{c}(k). \quad (3)$$

  In order to test the significance of $\bar{c}$, we calculate $\bar{c}_{binom} = \bar{k}/(N-1)$, the expected clustering coefficient in a control binomial graph. In a binomial graph, two vertices are linked with probability p. $p = \bar{k}/(N-1)$ is chosen so that the expected number of links of the binomial graphs is $n\bar{k}/2$ as in the original network.

Assortative mixing is known to be characteristic for *social-semiotic*, but not for *technical* networks (Newman, 2003). Recently, (Mehler, 2006) has shown that this characteristic varies a lot for different document networks and thus allows distinguishing linguistic networks which are homogeneously called 'small-worlds'. We have excluded on purpose the Pearson correlation coefficient of the degrees at the endpoints of edges that has been used in previous studies (Ferrer i Cancho et al., 2004) due to the statistical problems that this measure has in large networks with power degree distributions (Serrano et al., 2006).

3 The treebanks

We analyze seven treebanks each from a different language. Their features are summarized in Table 1. A comprehensive description of these and related banks is given by (Kakkonen, 2005). As explained by Kakkonen, one generally faces the problem of the heterogeneity not only of the annotation schemes, but also of the serialization formats used by them. Thus, we unified the various formats in order to get a single interface to the analysis of syntactic dependency networks derived thereof. Although

there exists a representation format for syntactic annotations (i.e. TIGER-XML — cf. (Mengel and Lezius, 2000)) we decided to use the Graph eXchange Language (GXL) in order to solve the heterogeneity problem. The GXL has been proposed as a uniform format for data interchange (Holt et al., 2006). It allows representing attributed, directed, undirected, mixed, ordered, hierarchical graphs as well as hypergraphs. Its application-dependent attribution model concerns vertices, edges and graphs. Because of its expressiveness it was utilized in modeling constituency structures (Pustylnikov, 2006) as well as nonlinear document structures (Mehler, 2007b). We utilize it to map syntactic dependency structures.

Our GXL binding is schematically explained as follows: corpora are mapped onto graphs which serialize graph models of sentence-related dependency structures. Each of these structures is mapped as a forest whose directed edges are mapped by means of the GXL's edge model. This model preserves the orientation of the input dependency relations. Figure 1 visualizes a sample dependency tree of the Slovene dependency treebank (Džeroski et al., 2006).

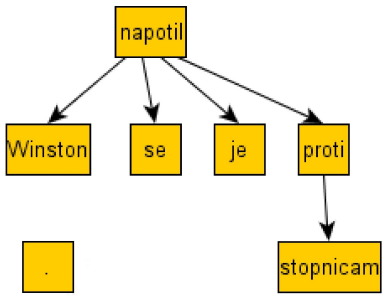


Figure 1: Visualization of a sample sentence of the Slovene dependency treebank (Džeroski et al., 2006) based on its reconstruction in terms of the GXL.

4 Results

A summary of the network measures obtained on the seven corpora is shown in Table 2. We find that $\bar{c} \gg \bar{c}_{binom}$ indicating a clear tendency of vertices connected to be connected if they are linked to the same vertex.

Since the Italian and the Romanian corpus are Romanic languages and the size of their networks is similar, they are paired in the figures. Figure 2 shows that the clustering $\bar{c}(k)$ decreases as k increases. Figure 3 shows that $\bar{k}_{nn}(k)$ decreases as k increases, indicating the presence of disassortative mixing when forming links, i.e. links tend to combine vertices of dissimilar degrees. For sufficiently large k the curves suggest a power-law behavior, i.e. $\bar{k}_{nn}(k) \sim k^{-\eta}$.

5 Discussion

We have found that the behavior of $\bar{k}_{nn}(k)$ suggests $\bar{k}_{nn}(k) \sim k^{-\eta}$ for sufficiently large k. A power-law behavior has been found in technical systems (Serrano et al., 2006). In a linguistic context, a power-law like behavior with two regimes has been found in the word adjacency network examined in (Masucci and Rodgers, 2006). A decreasing $\bar{k}_{nn}(k)$ for growing k (an indicator of dissortative mixing) has been found in biological and social systems (Serrano et al., 2006). A decreasing $\bar{c}(k)$ for growing k has been found in many non-linguistic systems (e.g. the Internet map at the autonomous system level), and also in a preliminary study of Czech and German syntactic dependency networks (Ferrer i Cancho et al., 2004). (Ravasz and Barabási, 2003) suggest that this behavior indicates the existence of a hierarchical network organization (Ravasz and Barabási, 2003). In our case this may indicate the existence of a core vocabulary surrounded by more and more special vocabularies. This observation is in accordance with a multipart organization of the rank frequency distribution of the lexical units involved. But this stratification is not simply due to the words' collocation patterns, but to their behavior in syntactic dependency structures. We have also found that $\bar{c} \gg \bar{c}_{binom}$, which is a common feature of non-linguistic (Newman, 2003) and linguistic networks (Mehler, 2007a) and, thus, is not very informative.

In sum, we have seen that GSDNs follow a common pattern of statistical correlations regardless of the heterogeneity of the languages and annotation criteria used. This suggests that the structure of GSDNs may originate from language independent principles. Since the correlational properties of GSDNs are not unique to these networks, our findings suggest that these principles may also be common to certain non-linguistic systems. Thus, in order to make GSDNs distinguishable in terms of their characteristics, finding more expressive network coeffi-

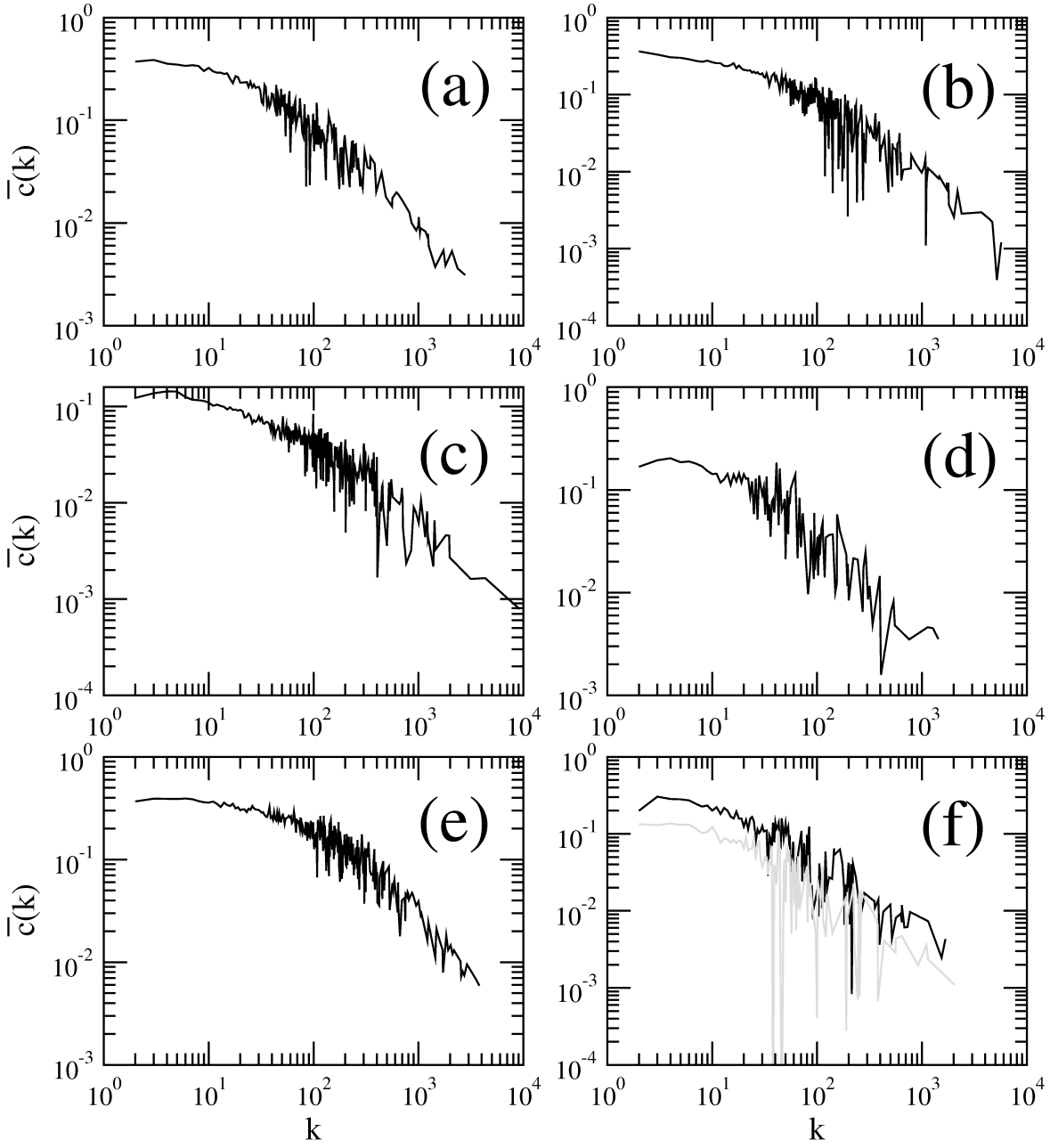


Figure 2: $\bar{c}(k)$, the mean clustering coefficient of vertices of degree k. (a) Danish, (b) Dutch, (c) Russian, (d) Slovene, (e) Swedish and (f) Italian (black) and Romanian (gray).

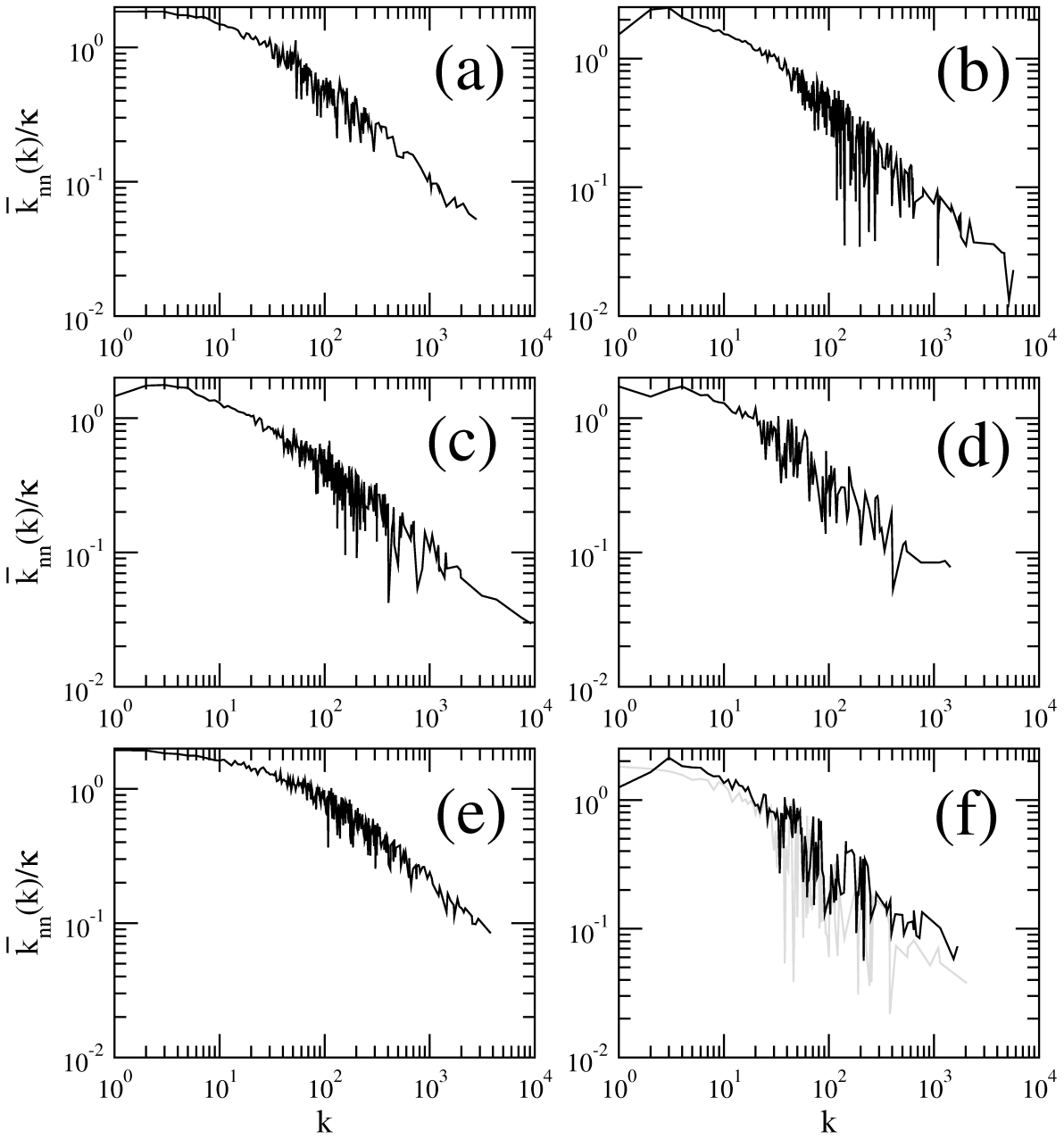


Figure 3: $\bar{k}_{nn}(k)/\kappa$, the normalized mean degree of the nearest neighbors of vertices of degree k. (a) Danish, (b) Dutch, (c) Russian, (d) Slovene, (e) Swedish and (f) Italian (black) and Romanian (gray).

Treebank	Language	Size (#nuclei)	Marks included	Reference
Alpino Treebank v. 1.2	Dutch	195.069	yes	(van der Beek et al., 2002)
Danish Dependency Treebank v. 1.0	Danish	100.008	yes	(Kromann, 2003)
Sample of sentences of the Dependency Grammar Annotator	Romanian	36.150	no	http://www.phobos.ro/roric/DGA/dga.html
Russian National Corpus	Russian	253.734	no	(Boguslavsky et al., 2002)
A sample of the Slovene Dependency Treebank v. 0.4	Slovene	36.554	yes	(Džeroski et al., 2006)
Talkbanken05 v. 1.1	Swedish	342.170	yes	(Nivre et al., 2006)
Turin University Treebank v. 0.1	Italian	44.721	no	(Bosco et al., 2000)

Table 1: Summary of the features of the treebanks used in this study. Besides the name, language and version of the corpus we indicate its size in terms of the number of nuclei tokens in the treebank. We also indicate if punctuation marks are treated as vertices of the syntactic structure of sentencess or not.

Language	N	$\bar{k}$	$\bar{c}$	$\bar{c}_{\mathrm{binom}}$
Alpino Treebank v. 1.2	28491	8.1	0.24	0.00028
Danish Dependency Treebank v. 1.0	19136	5.7	0.20	0.00030
Dependency Grammar Annotator	8867	5.3	0.093	0.00060
Russian National Corpus	58285	6.1	0.088	0.00010
Slovene Dependency Treebank v. 0.4	8354	5.3	0.12	0.00064
Talkbanken05 v. 1.1	25037	10.5	0.27	0.00042
Turin University Treebank v. 0.1	8001	6.9	0.18	0.00086

Table 2: Summary of the properties of the GSDNs analyzed. N is the number of vertices, $\bar{k}$ is the mean degree, $\bar{c}$ is the mean clustering coefficient, $\bar{c}_{binom}$ is the clustering coefficient of the control binomial graph.

cients is needed. A possible track could be considering the weight of a link, which is known to provide a more accurate description of the architecture of complex networks (Barrat et al., 2004).

Acknowledgement

We are grateful to Tomaž Erjavec for the opportunity to analyze a sample of the Slovene Dependency Treebank (http://nl.ijs.si/sdt/). ADG and RFC's work was funded by the projects FIS2006-13321-C02 and BFM2003-08258-C02-02 of the Spanish Ministry of Education and Science. AM and OP's work is supported by the SFB 673 *Alignment in Communication* (http://ariadne.coli.uni-bielefeld.de/sfb/) funded by the German Research Foundation (DFG).

References

Lucas Antiqueira, Maria das Gracas V. Nunes, Osvaldo N. Oliveira, and Luciano da F. Costa. 2006. Strong correlations between text quality and complex networks features. *Physica A*, 373:811–820.

Albert-László Barabási and Réka Albert. 2002. Statistical mechanics of complex networks. *Rev. Mod. Phys.*, 74:47–97.

A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. 2004. The architecture of complex weighted networks. In *Proc. Nat. Acad. Sci. USA*, volume 101, pages 3747–3752.

Igor Boguslavsky, Ivan Chardin, Svetlana Grigorieva, Nikolai Grigoriev, Leonid Iomdin, Leonid Kreidlin, and Nadezhda Frid. 2002. Development of a dependency treebank for russian and its possible applications in NLP. In *Proc. of LREC 2002*.

Stefan Bordag, Gerhard Heyer, and Uwe Quasthoff. 2003. Small worlds of concepts and other principles of semantic search. In *Proc. of the Second International Workshop on Innovative Internet Computing Systems (IICS '03)*.

Cristina Bosco, Vincenzo Lombardo, Daniela Vassallo, and Lesmo Lesmo. 2000. Building a treebank for Italian: a data-driven annotation schema. In *Proc. of LREC 2000*.

Silvia Maria Gomes Caldeira, Thierry Petit Lobão, Roberto Fernandes Silva Andrade, Alexis Neme, and J. G. Vivas Miranda. 2006. The network of concepts in written texts. *European Physical Journal B*, 49:523–529.

Serguei N. Dorogovtsev and Jose Fernando Ferreira Mendes. 2002. Evolution of random networks. *Adv. Phys.*, 51:1079–1187.

Sašo Džeroski, Tomaž Erjavec, Nina Ledinek, Petr Pajas, Zdenek Žabokrtský, and Andreja Žele. 2006. Towards a Slovene dependency treebank. In *Proc. of LREC 2006*.

Ramon Ferrer i Cancho and Ricard V. Solé. 2001. The small-world of human language. *Proc. R. Soc. Lond. B*, 268:2261–2266.

Ramon Ferrer i Cancho, Ricard V. Solé, and Reinhard Köhler. 2004. Patterns in syntactic dependency networks. *Physical Review E*, 69:051915.

Ramon Ferrer i Cancho. 2004. Euclidean distance between syntactically linked words. *Physical Review E*, 70:056135.

Ramon Ferrer i Cancho. 2005. The structure of syntactic dependency networks from recent advances in the study of linguistic networks. In V. Levickij and G. Altmann, editors, *The problems in quantitative linguistics*, pages 60–75. Ruta, Chernivtsi.

Richard C. Holt, Andy Schürr, Susan Elliott Sim, and Andreas Winter. 2006. GXL: A graph-based standard exchange format for reengineering. *Science of Computer Programming*, 60(2):149–170.

Tuomo Kakkonen. 2005. Dependency treebanks: methods, annotation schemes and tools. In *Proc. of NODALIDA 2005*, pages 94–104, Joensuu, Finland.

Matthias T. Kromann. 2003. The Danish dependency treebank and the underlying linguistic theory. In Joakim Nivre and Erhard Hinrichs, editors, *Proc. of TLT 2003*. Växjö University Press.

Adolfo Paolo Masucci and Geoff J. Rodgers. 2006. Network properties of written human language. *Physical Review E*, 74:026102.

Alexander Mehler. 2006. Text linkage in the wiki medium – a comparative study. In *Proc. of the EACL Workshop on New Text – Wikis and blogs and other dynamic text sources*, pages 1–8.

Alexander Mehler. 2007a. Large text networks as an object of corpus linguistic studies. In A. Lüdeling and M. Kytö, editors, *Corpus linguistics. An international handbook of the science of language and society*. de Gruyter, Berlin/New York.

Alexander Mehler. 2007b. Structure formation in the web. A graph-theoretical model of hypertext types. In A. Witt and D. Metzing, editors, *Linguistic Modeling of Information and Markup Languages*. Springer, Dordrecht.

Andreas Mengel and Wolfgang Lezius. 2000. An XML-based representation format for syntactically annotated corpora. In *Proc. of LREC 2000*.

Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. 2004. Superfamilies of evolved and designed networks. *Science*, 303:1538–1542.

Mark E. J. Newman. 2003. The structure and function of complex networks. *SIAM Review*, 45:167–256.

Joakim Nivre, Jens Nilsson, and Johan Hall. 2006. Talbanken05: A swedish treebank with phrase structure and dependency annotation. In *Proc. of LREC 2006*.

Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vesipignani. 2001. Dynamical and correlation properties of the internet. *Physical Review Letters*, 87(25):268701.

Olga Pustylnikov. 2006. How much information is provided by text structure? Automatic text classification using structural features (in German). Master thesis, University of Bielefeld, Germany.

Erzsébet Ravasz and Albert-László Barabási. 2003. Hierarchical organization in complex networks. *Phys. Rev. E*, 67:026112.

M. Ángeles Serrano, Marian Boguñá, Romualdo Pastor-Satorras, and Alessandro Vespignani. 2006. Correlations in complex networks. In G. Caldarelli and A. Vespignani, editors, *Structure and Dynamics of Complex Networks, From Information Technology to Finance and Natural Science*, chapter 1. World Scientific.

Mariano Sigman and Guillermo A. Cecchi. 2002. Global organization of the WordNet lexicon. In *Proc. Natl. Acad. Sci. USA*, volume 99, pages 1742–1747.

Márcio Medeiros Soares, Gilberto Corso, and Liacir dos Santos Lucena. 2005. The network of syllables in Portuguese. *Physica A*, 355(2-4):678–684.

Mark Steyvers and Josh Tenenbaum. 2005. The large-scale structure of semantic networks: statistical analyses and a model of semantic growth. *Cognitive Science*, 29(1):41–78.

Leonoor van der Beek, Gosse Bouma, Robert Malouf, and Gertjan van Noord. 2002. The Alpino dependency treebank. In *Proc. of the Conf. on Computational Linguistics in the Netherlands (CLIN '02)*.

DLSITE-2: Semantic Similarity Based on Syntactic Dependency Trees Applied to Textual Entailment

Daniel Micol, Óscar Ferrández, Rafael Muñoz, and Manuel Palomar
Natural Language Processing and Information Systems Group
Department of Computing Languages and Systems
University of Alicante
San Vicente del Raspeig, Alicante 03690, Spain
{dmicol, ofe, rafael, mpalomar}@dlsi.ua.es

Abstract

In this paper we attempt to deduce textual entailment based on syntactic dependency trees of a given text-hypothesis pair. The goals of this project are to provide an accurate and fast system, which we have called *DLSITE-2*, that can be applied in software systems that require a near-real-time interaction with the user. To accomplish this we use *MINIPAR* to parse the phrases and construct their corresponding trees. Later on we apply syntactic-based techniques to calculate the semantic similarity between text and hypothesis. To measure our method's precision we used the test text corpus set from *Second PASCAL Recognising Textual Entailment Challenge* (RTE-2), obtaining an accuracy rate of 60.75%.

1 Introduction

There are several methods used to determine textual entailment for a given text-hypothesis pair. The one described in this paper uses the information contained in the syntactic dependency trees of such phrases to deduce whether there is entailment or not. In addition, semantic knowledge extracted from WordNet (Miller et al., 1990) has been added to achieve higher accuracy rates.

It has been proven in several competitions and other workshops that textual entailment is a complex task. One of these competitions is *PASCAL Recognising Textual Entailment Challenge* (Bar-Haim et al., 2006), where each participating group develops a textual entailment recognizing system attempting to accomplish the best accuracy rate of all competitors. Such complexity is the reason why we use a combination of various techniques to deduce whether entailment is produced.

Currently there are few research projects related to the topic discussed in this paper. Some systems use syntactic tree matching as the textual entailment decision core module, such as (Katrenko and Adriaans, 2006). It is based on maximal embedded syntactic subtrees to analyze the semantic relation between text and hypothesis. Other systems use syntactic trees as a collaborative module, not being the core, such as (Herrera et al., 2006). The application discussed in this paper belongs to the first set of systems, since syntactic matching is its main module.

The remainder of this paper is structured as follows. In the second section we will describe the methods implemented in our system. The third one contains the experimental results, and the fourth and last discusses such results and proposes future work based on our actual research.

2 Methods

The system we have built aims to provide a good accuracy rate in a short lapse of time, making it feasible to be included in applications that require near-real-time responses due to their interaction with the user. Such a system is composed of few modules that behave collaboratively. These include tree construction, filtering, embedded subtree search and graph node matching. A schematic representation of the system architecture is shown in Figure 1.

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 73–80,
Rochester, April 2007

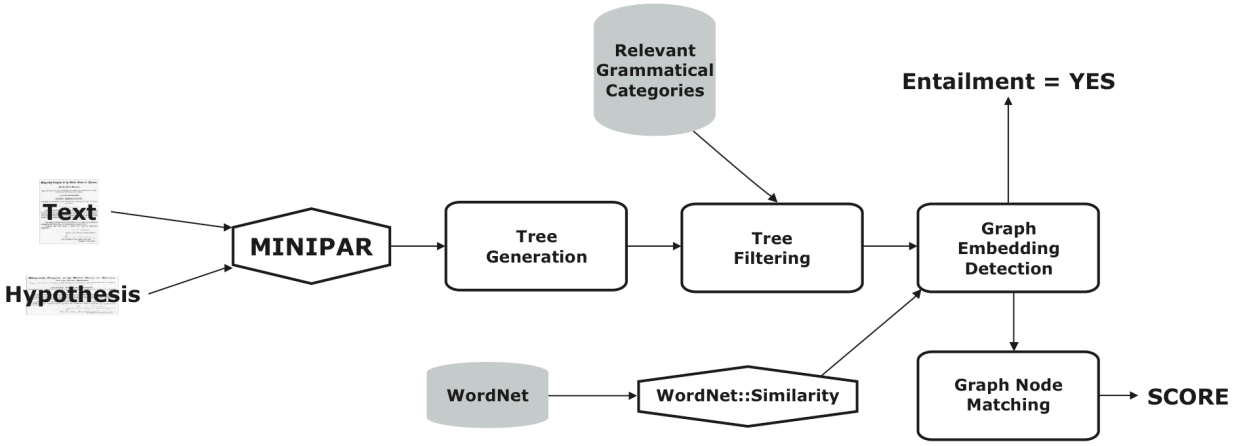


Figure 1: *DLSITE-2* system architecture.

Each of the steps or modules of *DLSITE-2* is described in the following subsections, that are numbered sequentially according to their execution order.

2.1 Tree generation

The first module constructs the corresponding syntactic dependency trees. For this purpose, *MINIPAR* (Lin, 1998) output is generated and afterwards parsed for each text and hypothesis of our corpus. Phrase tokens, along with their grammatical information, are stored in an on-memory data structure that represents a tree, which is equivalent to the mentioned syntactic dependency tree.

2.2 Tree filtering

Once the tree has been constructed, we may want to discard irrelevant data in order to reduce our system's response time and noise. For this purpose we have generated a database of relevant grammatical categories, represented in Table 1, that will allow us to remove from the tree all those tokens whose category does not belong to such list. The resulting tree will have the same structure as the original, but will not contain any stop words nor irrelevant tokens, such as determinants or auxiliary verbs. The whole list of ignored grammatical categories is represented in Table 2.

We have performed tests taking into account and discarding each grammatical category, which has allowed us to generate both lists of relevant and ignored grammatical categories.

Verbs, verbs with one argument, verbs with two arguments, verbs taking clause as complement, verb *Have*, verb *Be*
Nouns
Numbers
Adjectives
Adverbs
Noun-noun modifiers

Table 1: Relevant grammatical categories.

2.3 Graph embedding detection

The next step of our system consists in determining whether the hypothesis' tree is embedded into the text's. Let us first define the concept of embedded tree (Katrenko and Adriaans, 2006).

> **Definition 1: Embedded tree** A tree $T_1 = (V_1, E_1)$ is embedded into another one $T_2 = (V_2, E_2)$ iff
>
> 1. $V_1 \subseteq V_2$, and
> 2. $E_1 \subseteq E_2$
>
> where V_1 and V_2 represent the vertices, and E_1 and E_2 the edges.

In other words, a tree, T_1, is embedded into another one, T_2, if all nodes and branches of T_1 are present in T_2.

We believe that it makes sense to reduce the strictness of such a definition to allow the appearance of intermediate nodes in the text's branches that are

Determiners
Pre-determiners
Post-determiners
Clauses
Inflectional phrases
Preposition and preposition phrases
Specifiers of preposition phrases
Auxiliary verbs
Complementizers

Table 2: Ignored grammatical categories.

not present in the corresponding hypothesis' branch, which means that we allow partial matching. Therefore, a match between two branches will be produced if all nodes of the first one, namely $\theta_1 \in E_1$, are present in the second, namely $\theta_2 \in E_2$, and their respective order is the same, allowing the possibility of appearance of intermediate nodes that are not present in both branches. This is also described in (Katrenko and Adriaans, 2006).

To determine whether the hypothesis' tree is embedded into the text's, we perform a top-down matching process. For this purpose we first compare the roots of both trees. If they coincide, we then proceed to compare their respective child nodes, which are the tokens that have some sort of dependency with their respective root token.

In order to add more flexibility to our system, we do not require the pair of tokens to be exactly the same, but rather set a threshold that represents the minimum similarity value between them. This is a difference between our approach and the one described in (Katrenko and Adriaans, 2006). Such a similarity is calculated by using the *WordNet::Similarity* tool (Pedersen et al., 2004), and, concretely, the Wu-Palmer measure, as defined in Equation 1 (Wu and Palmer, 1994).

$$Sim(C_1, C_2) = \frac{2N_3}{N_1 + N_2 + 2N_3} \qquad (1)$$

where C_1 and C_2 are the synsets whose similarity we want to calculate, C_3 is their least common superconcept, N_1 is the number of nodes on the path from C_1 to C_3, N_2 is the number of nodes on the path from C_2 to C_3, and N_3 is the number of nodes on the path from C_3 to the root. All these synsets and distances can be observed in Figure 2.

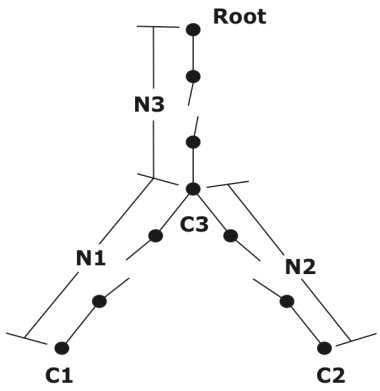


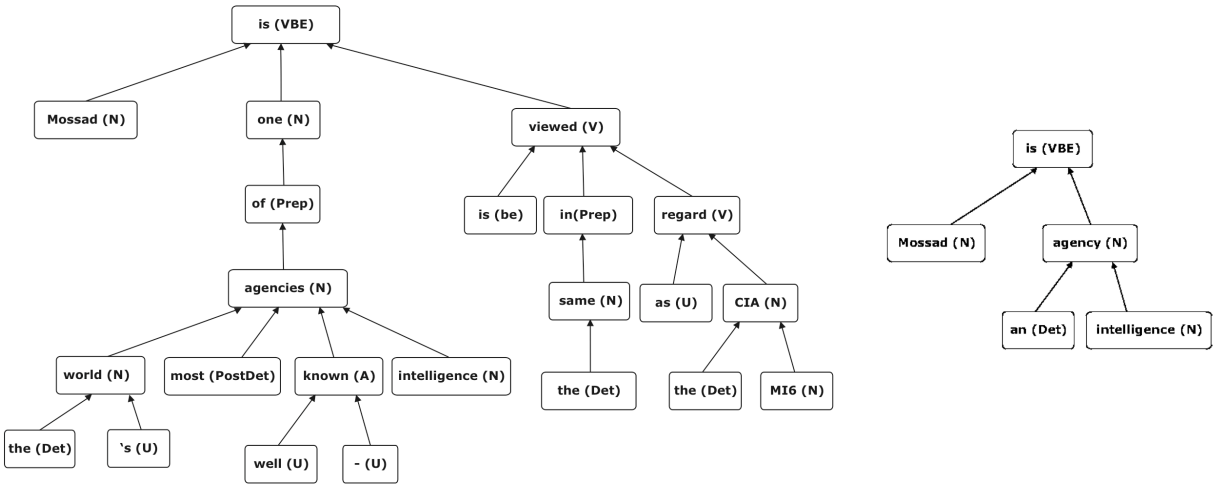
Figure 2: Distance between two synsets.

If the similarity rate is greater or equal than the established threshold, which we have set empirically to 80%, we will consider the corresponding hypothesis' token as suitable to have the same meaning as the text's token, and will proceed to compare its child nodes in the hypothesis' tree. On the other hand, if such similarity value is less than the corresponding threshold, we will proceed to compare the children of such text's tree node with the actual hypothesis' node that was being analyzed.

The comparison between the syntactic dependency trees of both text and hypothesis will be completed when all nodes of either tree have been processed. If we have been able to find a match for all the tokens within the hypothesis, the corresponding tree will be embedded into the text's and we will believe that there is entailment. If not, we will not be able to assure that such an implication is produced and will proceed to execute the next module of our system.

Next, we will present a text-hypothesis pair sample where the syntactic dependency tree of the hypothesis (Figure 3(b)) is embedded into the text's (Figure 3(a)). The mentioned text-hypothesis pair is the following:

> **Text:** *Mossad is one of the world's most well-known intelligence agencies, and is often viewed in the same regard as the CIA and MI6.*
> **Hypothesis:** *Mossad is an intelligence agency.*



(a) *Mossad is one of the world's most well-known intelligence agencies, and is often viewed in the same regard as the CIA and MI6.*

(b) *Mossad is an intelligence agency.*

Figure 3: *Representation of a hypothesis' syntactic dependency tree that is embedded into the text's.*

As one can see in Figure 3, the hypothesis' syntactic dependency tree represented is embedded into the text's because all of its nodes are present in the text in the same order. There is one exception though, that is the word *an*. However, since it is a determinant, the filtering module will have deleted it before the graph embedding test is performed. Therefore, in this example the entailment would be recognized.

2.4 Graph node matching

Once the embedded subtree comparison has finished, and if its result is negative, we proceed to perform a graph node matching process, termed alignment, between both the text and the hypothesis. This operation consists in finding pairs of tokens in both trees whose lemmas are identical, no matter whether they are in the same position within the tree. We would like to point out that in this step we do not use the *WordNet::Similarity* tool.

Some authors have already designed similar matching techniques, such as the ones described in (MacCartney et al., 2006) and (Snow et al., 2006). However, these include semantic constraints that we have decided not to consider. The reason of this decision is that we desired to overcome the textual entailment recognition from an exclusively syntactic perspective. Therefore, we did not want this module to include any kind of semantic knowledge.

The weight given to a token that has been found in both trees will depend on the depth in the hypothesis' tree and the token's grammatical relevance. The first of these factors depends on an empirically-calculated weight that assigns less importance to a node the deeper it is located in the tree. This weight is defined in Equation 2. The second factor gives different relevance depending on the grammatical category and relationship. For instance, a verb will have the highest weight, while an adverb or an adjective will have less relevance. The values assigned to each grammatical category and relationship are also empirically-calculated and are shown in Tables 3 and 4, respectively.

Grammatical category	**Weight**
Verbs, verbs with one argument, verbs with two arguments, verbs taking clause as complement	1.0
Nouns, numbers	0.75
Be used as a linking verb	0.7
Adjectives, adverbs, noun-noun modifiers	0.5
Verbs *Have* and *Be*	0.3

Table 3: Weights assigned to the grammatical categories.

Grammatical relationship	Weight
Subject of verbs, surface subject, object of verbs, second object of ditransitive verbs	1.0
The rest	0.5

Table 4: Weights assigned to the grammatical relationships.

Let τ and λ represent the text's and hypothesis' syntactic dependency trees, respectively. We assume we have found members of a synset, namely β, present in both τ and λ. Now let γ be the weight assigned to β's grammatical category (defined in Table 3), σ the weight of β's grammatical relationship (defined in Table 4), μ an empirically-calculated value that represents the weight difference between tree levels, and δ_β the depth of the node that contains the synset β in λ. We define the function $\phi(\beta)$ as represented in Equation 2.

$$\phi(\beta) = \gamma \cdot \sigma \cdot \mu^{-\delta_\beta} \tag{2}$$

The value obtained by calculating the expression of Equation 2 would represent the relevance of a synset in our system. The experiments performed reveal that the optimal value for μ is 1.1.

For a given pair (τ, λ), we define the set ξ as the one that contains the synsets present in both trees:

$$\xi = \tau \cap \lambda \;\; \forall \alpha \in \tau, \beta \in \lambda \tag{3}$$

Therefore, the similarity rate between τ and λ, denoted by the symbol ψ, would be defined as:

$$\psi(\tau, \lambda) = \sum_{\nu \in \xi} \phi(\nu) \tag{4}$$

One should note that a requirement of our system's similarity measure would be to be independent of the hypothesis length. Thus, we must define the normalized similarity rate, as shown in Equation 5.

$$\overline{\psi(\tau, \lambda)} = \frac{\psi(\tau, \lambda)}{\sum_{\beta \in \lambda} \phi(\beta)} = \frac{\sum_{\nu \in \xi} \phi(\nu)}{\sum_{\beta \in \lambda} \phi(\beta)} \tag{5}$$

Once the similarity value, $\overline{\psi(\tau, \lambda)}$, has been calculated, it will be provided to the user together with the corresponding text-hypothesis pair identifier. It will be his responsibility to choose an appropriate threshold that will represent the minimum similarity rate to be considered as entailment between text and hypothesis. All values that are under such a threshold will be marked as not entailed. For this purpose, we suggest using a development corpus in order to obtain the optimal threshold value, as it is done in the RTE challenges.

3 Experimental results

The experimental results shown in this paper were obtained processing a set of text-hypothesis pairs from RTE-2. The organizers of this challenge provide development and test corpora to the participants, both of them containing 800 pairs manually annotated for logical entailment. It is composed of four subsets, each of them corresponding to typical true and false entailments in different tasks, such as Information Extraction (IE), Information Retrieval (IR), Question Answering (QA), and Multi-document Summarization (SUM). For each task, the annotators selected the same amount of true entailments as negative ones (50%-50% split).

The organizers have also defined two measures to evaluate the participating systems. All judgments returned by the systems will be compared to those manually assigned by the human annotators. The percentage of matching judgments will provide the *accuracy* of the system, i.e. the percentage of correct responses. As a second measure, the *average precision* will be computed. This measure evaluates the ability of the systems to rank all the pairs in the corpus according to their entailment confidence, in decreasing order from the most certain entailment to the least. *Average precision* is a common evaluation measure for system rankings that is defined as shown in Equation 6.

$$AP = \frac{1}{R} \sum_{i=1}^{n} E(i) \frac{\#correct\_up\_to\_pair\_i}{i} \tag{6}$$

where n is the amount of the pairs in the test corpus, R is the total number of positive pairs in it, i ranges over the pairs, ordered by their ranking, and $E(i)$ is defined as follows:

$$E(i) = \begin{cases} 1 & \text{if the } i-th \text{ pair is positive,} \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

As we previously mentioned, we tested our system against RTE-2 development corpus, and used the test one to evaluate it.

First, Table 5 shows the accuracy (ACC) and average precision (AP), both as a percentage, obtained processing the development corpus from RTE-2 for a threshold value of 68.9%, which corresponds to the highest accuracy that can be obtained using our system for the mentioned corpus. It also provides the rate of correctly predicted true and false entailments.

Task	**ACC**	**AP**	**TRUE**	**FALSE**
IE	52.00	51.49	54.00	50.00
IR	55.50	58.99	32.00	79.00
QA	57.50	54.72	53.00	62.00
SUM	65.00	81.35	39.00	91.00
Overall	**57.50**	**58.96**	**44.50**	**70.50**

Table 5: Results obtained for the development corpus.

Next, let us show in Table 6 the results obtained processing the test corpus, which is the one used to compare the different systems that participated in RTE-2, with the same threshold as before.

Task	**ACC**	**AP**	**TRUE**	**FALSE**
IE	50.50	47.33	75.00	26.00
IR	64.50	67.67	59.00	70.00
QA	59.50	58.16	80.00	39.00
SUM	68.50	75.86	49.00	88.00
Overall	**60.75**	**57.91**	**65.75**	**55.75**

Table 6: Results obtained for the test corpus.

As one can observe in the previous table, our system provides a high accuracy rate by using mainly syntactical measures. The number of text-hypothesis pairs that succeeded the graph embedding evaluation was three for the development corpus and one for the test set, which reflects the strictness of such module. However, we would like to point out that the amount of pairs affected by the mentioned module will depend on the corpus nature, so it can vary significantly between different corpora.

Let us now compare our results with the ones that were achieved by the systems that participated in RTE-2. One should note that the criteria for such ranking is based exclusively on the accuracy, ignoring the average precision value. In addition, each participating group was allowed to submit two different systems to RTE-2. We will consider here the best result of both systems for each group. The mentioned comparison is shown in Table 7, and contains only the systems that had higher accuracy rates than our approach.

Participant	**Accuracy**
(Hickl et al., 2006)	75.38
(Tatu et al., 2006)	73.75
(Zanzotto et al., 2006)	63.88
(Adams, 2006)	62.62
(Bos and Markert, 2006)	61.62
DLSITE-2	60.75

Table 7: Comparison of some of the teams that participated in RTE-2.

As it is reflected in Table 7, our system would have obtained the sixth position out of twenty-four participants, which is an accomplishment considering the limited number of resources that it has built-in.

Since one of our system's modules is based on (Katrenko and Adriaans, 2006), we will compare their results with ours to analyze whether the modifications we introduced perform correctly. In RTE-2, they obtained an accuracy rate of 59.00% for the test corpus. The reason why we believe we have achieved better results than their system is due to the fact that we added semantic knowledge to our graph embedding module. In addition, the syntactic dependency trees to which we have applied such a module have been previously filtered to ensure that they do not contain irrelevant words. This reduces the system's noise and allows us to achieve higher accuracy rates.

In the introduction of this paper we mentioned that one of the goals of our system was to provide

a high accuracy rate in a short lapse of time. This is one of the reasons why we chose to construct a light system where one of the aspects to minimize was its response time. Table 8 shows the execution times[1] of our system for both development and test text corpora from RTE-2. These include total and average[2] response times.

	Development	Test
Total	1045	1023
Average	1.30625	1.27875

Table 8: *DLSITE-2* response times (in seconds).

As we can see, accurate results can be obtained using syntactic dependency trees in a short lapse of time. However, there are some limitations that our system does not avoid. For instance, the tree embedding test is not applicable when there is no verb entailment. This is reflected in the following pair:

> **Text:** *Tony Blair, the British Prime Minister, met Jacques Chirac in London.*
> **Hypothesis:** *Tony Blair is the British Prime Minister.*

The root node of the hypothesis' tree would be the one corresponding to the verb *is*. Since the entailment here is implicit, there is no need for such a verb to appear in the text. However, this is not compatible with our system, since *is* would not match any node of the text's tree, and thus the hypothesis' tree would not be found embedded into the text's.

The graph matching process would not behave correctly either. This is due to the fact that the main verb, which has the maximum weight because it is the root of the hypothesis' tree and its grammatical category has the maximum relevance, is not present in the text, so the overall similarity score would have a considerable handicap.

The example of limitation of our system that we have presented is an apposition. To avoid this specific kind of situations that produce an undesired behavior in our system, we could add a preprocessing module that transforms the phrases that have the structure X, Y, Z into X *is* Y, *and* Z. For the shown example, the resulting text and hypothesis would be as follows:

> **Text:** *Tony Blair is the British Prime Minister, and met Jacques Chirac in London.*
> **Hypothesis:** *Tony Blair is the British Prime Minister.*

The transformed text would still be syntactically correct, and the entailment would be detected since the hypothesis' syntactic dependency tree is embedded into the text's.

4 Conclusions and future work

The experimental results obtained from this research demonstrate that it is possible to apply a syntactic-based approach to deduce textual entailment from a text-hypothesis pair. We can obtain good accuracy rates using the discussed techniques with very short response times, which is very useful for assisting different kinds of tasks that demand near-real-time responses to user interaction.

The baseline we set for our system was to achieve better results than the ones we obtained with our last participation in RTE-2. As it is stated in (Ferrández et al., 2006), the maximum accuracy value obtained by then was 55.63% for the test corpus. Therefore, our system is 9.20% more accurate compared to the one that participated in RTE-2, which represents a considerable improvement.

The authors of this paper believe that if higher accuracy rates are desired, a step-based system must be constructed. This would have several preprocessing units, such as negation detectors, multi-word associators and so on. The addition of these units would definitely increase the response time preventing the system from being used in real-time tasks.

Future work can be related to the cases where no verb entailment is produced. For this purpose we propose to extract a higher amount of semantic information that would allow us to construct a characterized representation based on the input text, so that we can deduce entailment even if there is no apparent structure similarity between text and hypothesis. This would mean to create an abstract conceptualization of the information contained in the analyzed phrases, allowing us to deduce ideas that are not

[1] The machine we used to measure the response times had an Intel Core 2 Duo processor at 2GHz.

[2] Average response times are calculated diving the totals by the number of pairs in the corpus.

explicitly mentioned in the parsed text-hypothesis pairs.

In addition, the weights and thresholds defined in our system have been established empirically. It would be interesting to calculate those values by means of a machine learning algorithm and compare them to the ones we have obtained empirically. Some authors have already performed this comparison, being one example the work described in (MacCartney et al., 2006).

Acknowledgments

The authors of this paper would like to thank professors Borja Navarro and Rafael M. Terol for their help and critical comments.

This research has been supported by the undergraduate research fellowships financed by the Spanish Ministry of Education and Science, the project TIN2006-15265-C06-01 financed by such ministry, and the project ACOM06/90 financed by the Spanish Generalitat Valenciana.

References

Rod Adams. 2006. *Textual Entailment Through Extended Lexical Overlap*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. *The Second PASCAL Recognising Textual Entailment Challenge*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

Johan Bos, and Katja Markert. 2006. *When logical inference helps determining textual entailment (and when it doesnt)*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

Óscar Ferrández, Rafael M. Terol, Rafael Muñoz, Patricio Martínez-Barco, and Manuel Palomar. 2006. *An approach based on Logic Forms and WordNet relationships to Textual Entailment performance*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

Jesús Herrera, Anselmo Peñas, Álvaro Rodrigo, and Felisa Verdejo. 2006. *UNED at PASCAL RTE-2 Challenge*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. *Recognizing Textual Entailment with LCC's GROUNDHOG System*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

Sophia Katrenko, and Pieter Adriaans. 2006. *Using Maximal Embedded Syntactic Subtrees for Textual Entailment Recognition*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

Dekang Lin. 1998. *Dependency-based Evaluation of MINIPAR*. In Workshop on the Evaluation of Parsing Systems, Granada, Spain.

Bill MacCartney, Trond Grenager, Marie-Catherine de Marneffe, Daniel Cer, and Christopher D. Manning. 2006. *Learning to recognize features of valid textual entailments*. In Proceedings of the North American Association of Computational Linguistics (NAACL-06), New York City, New York, United States of America.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. *Introduction to WordNet: An On-line Lexical Database*. International Journal of Lexicography 1990 3(4):235-244.

Ted Pedersen, Siddhart Patwardhan, and Jason Michelizzi. 2004. *WordNet::Similarity - Measuring the Relatedness of Concepts*. In Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL-04), Boston, Massachussets, United States of America.

Rion Snow, Lucy Vanderwende, and Arul Menezes. 2006. *Effectively using syntax for recognizing false entailment*. In Proceedings of the North American Association of Computational Linguistics (NAACL-06), New York City, New York, United States of America.

Marta Tatu, Brandon Iles, John Slavick, Adrian Novischi, and Dan Moldovan. 2006. *COGEX at the Second Recognizing Textual Entailment Challenge*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

Zhibiao Wu, and Martha Palmer. 1994. *Verb Semantics and Lexical Selection*. In Proceedings of the 32nd Annual Meeting of the Associations for Computational Linguistics, pages 133-138, Las Cruces, New Mexico, United States of America.

Fabio M. Zanzotto, Alessandro Moschitti, Marco Pennacchiotti, and Maria T. Pazienza. 2006. *Learning textual entailment from examples*. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.

How Difficult is it to Develop a Perfect Spell-checker? A Cross-linguistic Analysis through Complex Network Approach

Monojit Choudhury[1], **Markose Thomas**[2], **Animesh Mukherjee**[1], **Anupam Basu**[1], **and Niloy Ganguly**[1]
[1]Department of Computer Science and Engineering, IIT Kharagpur, India
{monojit,animeshm,anupam,niloy}@cse.iitkgp.ernet.in
[2]Google Inc. Bangalore, India
markysays@gmail.com

Abstract

The difficulties involved in spelling error detection and correction in a language have been investigated in this work through the conceptualization of SpellNet – the weighted network of words, where edges indicate orthographic proximity between two words. We construct SpellNets for three languages - Bengali, English and Hindi. Through appropriate mathematical analysis and/or intuitive justification, we interpret the different topological metrics of SpellNet from the perspective of the issues related to spell-checking. We make many interesting observations, the most significant among them being that the probability of making a real word error in a language is propotionate to the average weighted degree of SpellNet, which is found to be highest for Hindi, followed by Bengali and English.

1 Introduction

Spell-checking is a well researched area in NLP, which deals with detection and automatic correction of spelling errors in an electronic text document. Several approaches to spell-checking have been described in the literature that use statistical, rule-based, dictionary-based or hybrid techniques (see (Kukich, 1992) for a dated but substantial survey). Spelling errors are broadly classified as *non-word errors* (NWE) and *real word errors* (RWE). If the misspelt string is a valid word in the language, then it is called an RWE, else it is an NWE. For example, in English, the word "fun" might be misspelt as "gun" or "vun"; while the former is an RWE, the latter is a case of NWE. It is easy to detect an NWE, but correction process is non-trivial. RWE, on the other hand are extremely difficult to detect as it requires syntactic and semantic analysis of the text, though the difficulty of correction is comparable to that of NWE (see (Hirst and Budanitsky, 2005) and references therein).

Given a lexicon of a particular language, how hard is it to develop a perfect spell-checker for that language? Since context-insensitive spell-checkers cannot detect RWE and neither they can effectively correct NWE, the difficulty in building a perfect spell-checker, therefore, is reflected by quantities such as the probability of a misspelling being RWE, probability of more than one word being orthographically closer to an NWE, and so on. In this work, we make an attempt to understand and formalize some of these issues related to the challenges of spell-checking through a complex network approach (see (Albert and Barabási, 2002; Newman, 2003) for a review of the field). This in turn allows us to provide language-specific quantitative bounds on the performance level of spell-checkers.

In order to formally represent the orthographic structure (spelling conventions) of a language, we conceptualize the lexicon as a weighted network, where the nodes represent the words and the weights of the edges indicate the orthoraphic similarity between the pair of nodes (read words) they connect. We shall call this network the **S**pelling **N**etwork or **SpellNet** for short. We build the SpellNets for three languages – Bengali, English and Hindi, and carry out standard topological analysis of the networks following complex network theory. Through appropriate mathematical analysis and/or intuitive justi-

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 81–88,
Rochester, April 2007

fication, we interpret the different topological metrics of SpellNet from the perspective of difficulties related to spell-checking. Finally, we make several cross-linguistic observations, both invariances and variances, revealing quite a few interesting facts. For example, we see that among the three languages studied, the probability of RWE is highest in Hindi followed by Bengali and English. A similar observation has been previously reported in (Bhatt et al., 2005) for RWEs in Bengali and English.

Apart from providing insight into spell-checking, the complex structure of SpellNet also reveals the self-organization and evolutionary dynamics underlying the orthographic properties of natural languages. In recent times, complex networks have been successfully employed to model and explain the structure and organization of several natural and social phenomena, such as the foodweb, protien interaction, formation of language inventories (Choudhury et al., 2006), syntactic structure of languages (i Cancho and Solé, 2004), WWW, social collaboration, scientific citations and many more (see (Albert and Barabási, 2002; Newman, 2003) and references therein). This work is inspired by the aforementioned models, and more specifically a couple of similar works on *phonological neighbors'* network of words (Kapatsinski, 2006; Vitevitch, 2005), which try to explain the human perceptual and cognitive processes in terms of the organization of the mental lexicon.

The rest of the paper is organized as follows. Section 2 defines the structure and construction procedure of SpellNet. Section 3 and 4 describes the degree and clustering related properties of SpellNet and their significance in the context of spell-checking, respectively. Section 5 summarizes the findings and discusses possible directions for future work. The derivation of the probability of RWE in a language is presented in Appendix A.

2 SpellNet: Definition and Construction

In order to study and formalize the orthographic characteristics of a language, we model the lexicon Λ of the language as an undirected and fully connected weighted graph $G(V, E)$. Each word $w \in \Lambda$ is represented by a vertex $v_w \in V$, and for every pair of vertices v_w and $v_{w'}$ in V, there is an edge

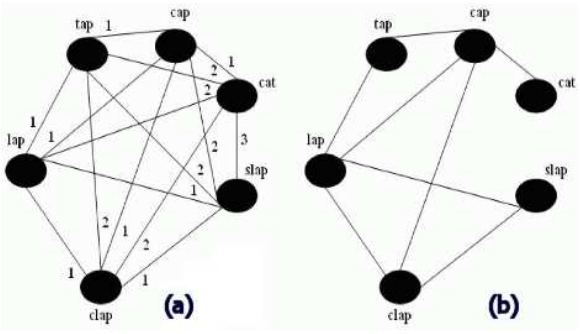


Figure 1: The structure of SpellNet: (a) the weighted SpellNet for 6 English words, (b) Thresholded counterpart of (a), for $\theta = 1$

$(v_w, v_{w'}) \in E$. The *weight* of the edge $(v_w, v_{w'})$, is equal to $ed(w, w')$ – the orthographic edit distance between w and w' (considering substitution, deletion and insertion to have a cost of 1). Each node $v_w \in V$ is also assigned a node weight $W_V(v_w)$ equal to the unigram occurrence frequency of the word w. We shall refer to the graph $G(V, E)$ as the SpellNet. Figure 1(a) shows a hypothetical SpellNet for 6 common English words.

We define unweighted versions of the graph $G(V, E)$ through the concept of thresholding as described below. For a threshold θ, the graph $G_\theta(V, E_\theta)$ is an unweighted sub-graph of $G(V, E)$, where an edge $(v_w, v_{w'}) \in E$ is assigned a weight 1 in E_θ if and only if the weight of the edge is less than or equal to θ, else it is assigned a weight 0. In other words, E_θ consists of only those edges in E whose edge weight is less than or equal to θ. Note that all the edges in E_θ are unweighted. Figure 1(b) shows the thresholded SpellNet shown in 1(a) for $\theta = 1$.

2.1 Construction of SpellNets

We construct the SpellNets for three languages – Bengali, English and Hindi. While the two Indian languages – Bengali and Hindi – use Brahmi derived scripts – Bengali and Devanagari respectively, English uses the Roman script. Moreover, the orthography of the two Indian languages are highly phonemic in nature, in contrast to the morpheme-based orthography of English. Another point of disparity lies in the fact that while the English alphabet consists of 26 characters, the alphabet size of both Hindi and Bengali is around 50.

The lexica for the three languages have been taken from public sources. For English it has been obtained from the website *www.audiencedialogue.org/susteng.html*; for Hindi and Bengali, the word lists as well as the unigram frequencies have been estimated from the monolingual corpora published by Central Institute of Indian Languages. We chose to work with the most frequent 10000 words, as the medium size of the two Indian language corpora (around 3M words each) does not provide sufficient data for estimation of the unigram frequencies of a large number of words (say 50000). Therefore, all the results described in this work pertain to the SpellNets corresponding to the most frequent 10000 words. However, we believe that the trends observed do not reverse as we increase the size of the networks.

In this paper, we focus on the networks at three different thresholds, that is for $\theta = 1, 3, 5$, and study the properties of G_θ for the three languages. We do not go for higher thresholds as the networks become completely connected at $\theta = 5$. Table 1 reports the values of different topological metrics of the SpellNets for the three languages at three thresholds. In the following two sections, we describe in detail some of the topological properties of SpellNet, their implications to spell-checking, and observations in the three languages.

3 Degree Distribution

The *degree* of a vertex in a network is the number of edges incident on that vertex. Let P_k be the probability that a randomly chosen vertex has degree k or more than k. A plot of P_k for any given network can be formed by making a histogram of the degrees of the vertices, and this plot is known as the *cumulative degree distribution* of the network (Newman, 2003). The (cumulative) degree distribution of a network provides important insights into the topological properties of the network.

Figure 2 shows the plots for the cumulative degree distribution for $\theta = 1, 3, 5$, plotted on a log-linear scale. The linear nature of the curves in the semi-logarithmic scale indicates that the distribution is exponential in nature. The exponential behaviour is clearly visible for $\theta = 1$, however at higher thresholds, there are very few nodes in the network with low degrees, and therefore only the tail of the curve shows a pure exponential behavior. We also observe that the steepness (i.e. slope) of the $\log(P_k)$ with respect to k increases with θ. It is interesting to note that although most of the naturally and socially occurring networks exhibit a power-law degree distribution (see (Albert and Barabási, 2002; Newman, 2003; i Cancho and Solé, 2004; Choudhury et al., 2006) and references therein), SpellNets feature exponential degree distribution. Nevertheless, similar results have also been reported for the phonological neighbors' network (Kapatsinski, 2006).

3.1 Average Degree

Let the degree of the node v be denoted by $k(v)$. We define the quantities – the average degree $\langle k \rangle$ and the weighted average degree $\langle k_{wt} \rangle$ for a given network as follows (we drop the subscript w for clarity of notation).

$$\langle k \rangle = \frac{1}{N} \sum_{v \in V} k(v) \tag{1}$$

$$\langle k_{wt} \rangle = \frac{\sum_{v \in V} k(v) W_V(v)}{\sum_{v \in V} W_V(v)} \tag{2}$$

where N is the number of nodes in the network.

Implication: The average weighted degree of SpellNet can be interpreted as the probability of RWE in a language. This correlation can be derived as follows. Given a lexicon Λ of a language, it can be shown that the probability of RWE in a language, denoted by $p_{rwe}(\Lambda)$ is given by the following equation (see Appendix A for the derivation)

$$p_{rwe}(\Lambda) = \sum_{w \in \Lambda} \sum_{\substack{w' \in \Lambda \\ w \neq w'}} \rho^{ed(w,w')} p(w) \tag{3}$$

Let $neighbor(w, d)$ be the number of words in Λ whose edit distance from w is d. Eqn 3 can be rewritten in terms of $neighbor(w, d)$ as follows.

$$p_{rwe}(\Lambda) = \sum_{w \in \Lambda} \sum_{d=1}^{\infty} \rho^d \; neighbor(w, d) p(w) \tag{4}$$

Practically, we can always assume that d is bounded by a small positive integer. In other words, the number of errors simultaneously made on a word is always small (usually assumed to be 1 or a

	English			Hindi			Bengali		
	$\theta = 1$	$\theta = 3$	$\theta = 5$	$\theta = 1$	$\theta = 3$	$\theta = 5$	$\theta = 1$	$\theta = 3$	$\theta = 5$
M	8.97k	0.70M	8.46M	17.6k	1.73M	17.1M	11.9k	1.11M	13.2M
$\langle k \rangle$	2.79	140.25	1692.65	4.52	347.93	3440.06	3.38	223.72	2640.11
$\langle k_{wt} \rangle$	6.81	408.03	1812.56	13.45	751.24	4629.36	7.73	447.16	3645.37
r_{dd}	0.696	0.480	0.289	0.696	0.364	0.129	0.702	0.389	0.155
$\langle CC \rangle$	0.101	0.340	0.563	0.172	0.400	0.697	0.131	0.381	0.645
$\langle CC_{wt} \rangle$	0.221	0.412	0.680	0.341	0.436	0.760	0.229	0.418	0.681
$\langle l \rangle$	7.07	3.50	N.E	7.47	2.74	N.E	8.19	2.95	N.E
D	24	14	N.E	26	12	N.E	29	12	N.E

Table 1: Various topological metrics and their associated values for the SpellNets of the three languages at thresholds 1, 3 and 5. Metrics: M – number of edges; $\langle k \rangle$ – average degree; $\langle k_{wt} \rangle$ – average weighted degree; $\langle CC \rangle$ – average clustering coefficient; $\langle CC_{wt} \rangle$ - average weighted clustering coefficient; r_{dd} – Pearson correlation coefficient between degrees of neighbors; $\langle l \rangle$ – average shortest path; D – diameter. N.E – Not Estimated. See the text for further details on definition, computation and significance of the metrics.

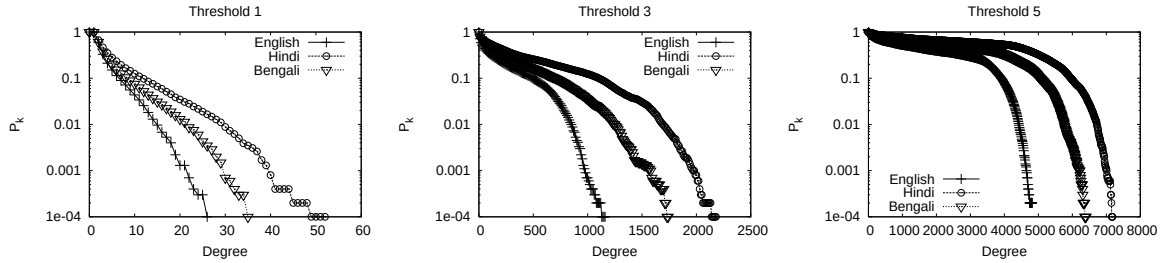


Figure 2: Cumulative degree distribution of SpellNets at different thresholds presented in semi-logarithmic scale.

slowly growing function of the word length (Kukich, 1992)). Let us denote this bound by θ. Therefore,

$$p_{rwe}(\Lambda) \approx \sum_{w \in \Lambda} \sum_{d=1}^{\theta} \rho^d \; neighbor(w, d) p(w) \quad (5)$$

Since $\rho < 1$, we can substitute ρ^d by ρ to get an upper bound on $p_{rwe}(\Lambda)$, which gives

$$p_{rwe}(\Lambda) < \rho \sum_{w \in \Lambda} \sum_{d=1}^{\theta} neighbor(w, d) p(w) \quad (6)$$

The term $\sum_{d=1}^{\theta} neighbor(w, d)$ computes the number of words in the lexicon, whose edit distance from w is atmost θ. This is nothing but $k(v_w)$, i.e. the degree of the node v_w, in G_θ. Moreover, the term $p(w)$ is proportionate to the node weight $W_V(v_w)$. Thus, rewriting Eqn 6 in terms of the network parameters for G_θ, we get (subscript w is dropped for clarity)

$$p_{rwe}(\Lambda) < \rho \frac{\sum_{v \in V} k(v) W_V(v)}{\sum_{v \in V} W_V(v)} \quad (7)$$

Comparing Eqn 2 with the above equation, we can directly obtain the relation

$$p_{rwe}(\Lambda) < C_1 \langle k_{wt} \rangle \quad (8)$$

where C_1 is some constant of proportionality. Note that for $\theta = 1$, $p_{rwe}(\Lambda) \propto \langle k_{wt} \rangle$. If we ignore the distribution of the words, that is if we assume $p(w) = 1/N$, then $p_{rwe}(\Lambda) \propto \langle k \rangle$.

Thus, the quantity $\langle k_{wt} \rangle$ provides a good estimate of the probability of RWE in a language.

Observations and Inference: At $\theta = 1$, the average weighted degrees for Hindi, Bengali and English are 13.81, 7.73 and 6.61 respectively. Thus, the probability of RWE in Hindi is significantly higher

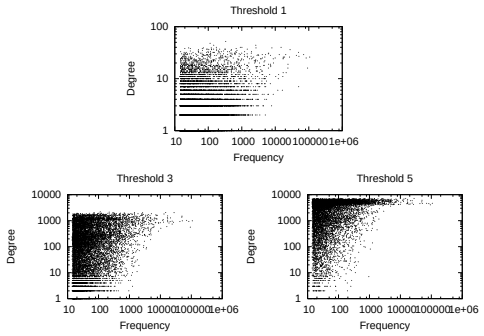


Figure 3: Scatter-plots for degree versus unigram frequency at different θ for Hindi

than that of Bengali, which in turn is higher than that of English (Bhatt et al., 2005). Similar trends are observed at all the thresholds for both $\langle k_{wt} \rangle$ and $\langle k \rangle$. This is also evident from Figures 2, which show the distribution of Hindi to lie above that of Bengali, which lies above English (for all thresholds).

The average degree $\langle k \rangle$ is substantially smaller (0.5 to 0.33 times) than the average weighted degree $\langle k_{wt} \rangle$ for all the 9 SpellNets. This suggests that the higher degree nodes in SpellNet have higher node weight (i.e. occurrence frequency). Indeed, as shown in Figure 3 for Hindi, the high unigram frequency of a node implies higher degree, though the reverse is not true. The scatter-plots for the other languages are similar in nature.

3.2 Correlation between Degrees of Neighbors

The relation between the degrees of adjacent words is described by the *degree assortativity coefficient*. One way to define the assortativity of a network is through the Pearson correlation coefficient between the degrees of the two vertices connected by an edge. Each edge (u, v) in the network adds a data item corresponding to the degrees of u and v to two data sets x and y respectively. The Pearson correlation coefficient for the data sets x and y of n items each is then defined as

$$r = \frac{n \sum xy - \sum x \sum y}{\sqrt{[n \sum x^2 - (\sum x)^2][n \sum y^2 - (\sum y)^2]}}$$

Observation: r is positive for the networks in which words tend to associate with other words of similar degree (i.e. high degree with high degree and vice versa), and it is negative for networks in which words associate with words having degrees in the opposite spectrum. Refering to table 1, we see that the correlation coefficient r_{dd} is roughly the same and equal to around 0.7 for all languages at $\theta = 1$. As θ increases, the correlation decreases as expected, due to the addition of edges between dissimilar words.

Implication: The high positive correlation coefficients suggest that SpellNets feature assortative mixing of nodes in terms of degrees. If there is an RWE corresponding to a high degree node v_w, then due to the assortative mixing of nodes, the misspelling w' obtained from w, is also expected to have a high degree. Since w' has a high degree, even after detection of the fact that w' is a misspelling, choosing the right suggestion (i.e. w) is extremely difficult unless the linguistic context of the word is taken into account. Thus, more often than not it is difficult to correct an RWE, even after successful detection.

4 Clustering and Small World Properties

In the previous section, we looked at some of the degree based features of SpellNets. These features provide us insights regarding the probability of RWE in a language and the level of difficulty in correcting the same. In this section, we discuss some of the other characteristics of SpellNets that are useful in predicting the difficulty of non-word error correction.

4.1 Clustering Coefficient

Recall that in the presence of a complete list of valid words in a language, detection of NWE is a trivial task. However, correction of NWE is far from trivial. Spell-checkers usually generate a suggestion list of possible candidate words that are within a small edit distance of the misspelling. Thus, correction becomes hard as the number of words within a given edit distance from the misspelling increases. Suppose that a word $w \in \Lambda$ is transformed into w' due to some typing error, such that $w' \notin \Lambda$. Also assume that $ed(w, w') \leq \theta$. We want to estimate the number of words in Λ that are within an edit distance θ of w'. In other words we are interested in finding out the degree of the node $v_{w'}$ in G_θ, but since there is no such node in SpellNet, we cannot compute this quantity directly. Nevertheless, we can provide an

approximate estimate of the same as follows.

Let us conceive of a hypothetical node $v_{w'}$. By definition of SpellNet, there should be an edge connecting $v_{w'}$ and v_w in G_θ. A crude estimate of $k(v_{w'})$ can be $\langle k_{wt} \rangle$ of G_θ. Due to the assortative nature of the network, we expect to see a high correlation between the values of $k(v_w)$ and $k(v_{w'})$, and therefore, a slightly better estimate of $k(v_{w'})$ could be $k(v_w)$. However, as $v_{w'}$ is not a part of the network, it's behavior in SpellNet may not resemble that of a real node, and such estimates can be grossly erroneous.

One way to circumvent this problem is to look at the local neighborhood of the node v_w. Let us ask the question – what is the probability that two randomly chosen neighbors of v_w in G_θ are connected to each other? If this probability is high, then we can expect the local neighborhood of v_w to be dense in the sense that almost all the neighbors of v_w are connected to each other forming a *clique-like* local structure. Since $v_{w'}$ is a neighbor of v_w, it is a part of this dense cluster, and therefore, its degree $k(v_{w'})$ is of the order of $k(v_w)$. On the other hand, if this probability is low, then even if $k(v_w)$ is high, the space around v_w is sparse, and the local neighborhood is *star-like*. In such a situation, we expect $k(v_{w'})$ to be low.

The topological property that measures the probability of the neighbors of a node being connected is called the *clustering coefficient* (CC). One of the ways to define the clustering coefficient $C(v)$ for a vertex v in a network is

$$C(v) = \frac{\text{number of triangles connected to vertex } v}{\text{number of triplets centered on } v}$$

For vertices with degree 0 or 1, we put $C(v) = 0$. Then the clustering coefficient for the whole network $\langle CC \rangle$ is the mean CC of the nodes in the network. A corresponding weighted version of the CC $\langle CC_{wt} \rangle$ can be defined by taking the node weights into account.

Implication: The higher the value of $k(v_w)C(v_w)$ for a node, the higher is the probability that an NWE made while typing w is hard to correct due to the presence of a large number of orthographic neighbors of the misspelling. Therefore, in a way $\langle CC_{wt} \rangle$ reflects the level of difficulty in correcting NWE for the language in general.

Observation and Inference: At threshold 1, the values of $\langle CC \rangle$ as well as $\langle CC_{wt} \rangle$ is higher for Hindi (0.172 and 0.341 respectively) and Bengali (0.131 and 0.229 respectively) than that of English (0.101 and 0.221 respectively), though for higher thresholds, the difference between the CC for the languages reduces. This observation further strengthens our claim that the level of difficulty in spelling error detection and correction are language dependent, and for the three languages studied, it is hardest for Hindi, followed by Bengali and English.

4.2 Small World Property

As an aside, it is interesting to see whether the SpellNets exhibit the so called *small world effect* that is prevalent in many social and natural systems (see (Albert and Barabási, 2002; Newman, 2003) for definition and examles). A network is said to be a *small world* if it has a high clustering coefficient and if the average shortest path between any two nodes of the network is small.

Observation: We observe that SpellNets indeed feature a high CC that grows with the threshold. The average shortest path, denoted by $\langle l \rangle$ in Table 1, for $\theta = 1$ is around 7 for all the languages, and reduces to around 3 for $\theta = 3$; at $\theta = 5$ the networks are near-cliques. Thus, SpellNet is a small world network.

Implication: By the application of triangle inequality of edit distance, it can be easily shown that $\langle l \rangle \times \theta$ provides an upper bound on the average edit distance between all pairs of the words in the lexicon. Thus, a small world network, which implies a small $\langle l \rangle$, in turn implies that as we increase the error bound (i.e. θ), the number of edges increases sharply in the network and soon the network becomes fully connected. Therefore, it becomes increasingly more difficult to correct or detect the errors, as any word can be a possible suggestion for any misspelling. In fact this is independently observed through the exponential rise in M – the number of edges, and fall in $\langle l \rangle$ as we increase θ.

Inference: It is impossible to correct very noisy texts, where the nature of the noise is random and words are distorted by a large edit distance (say 3 or more).

5 Conclusion

In this work, we have proposed the network of orthographic neighbors of words or the SpellNet and studied the structure of the same across three languages. We have also made an attempt to relate some of the topological properties of SpellNet to spelling error distribution and hardness of spell-checking in a language. The important observations of this study are summarized below.

- The probability of RWE in a language can be equated to the average weighted degree of SpellNet. This probablity is highest in Hindi followed by Bengali and English.
- In all the languages, the words that are more prone to undergo an RWE are more likely to be misspelt. Effectively, this makes RWE correction very hard.
- The hardness of NWE correction correlates with the weighted clustering coefficient of the network. This is highest for Hindi, followed by Bengali and English.
- The basic topology of SpellNet seems to be an invariant across languages. For example, all the networks feature exponential degree distribution, high clustering, assortative mixing with respect to degree and node weight, small world effect and positive correlation between degree and node weight, and CC and degree. However, the networks vary to a large extent in terms of the actual values of some of these metrics.

Arguably, the language-invariant properties of SpellNet can be attributed to the organization of the human mental lexicon (see (Kapatsinski, 2006) and references therein), self-organization of orthographic systems and certain properties of edit distance measure. The differences across the languages, perhaps, are an outcome of the specific orthographic features, such as the size of the alphabet. Another interesting observation is that the phonemic nature of the orthography strongly correlates with the difficulty of spell-checking. Among the three languages, Hindi has the most phonemic and English the least phonemic orthography. This correlation calls for further investigation.

Throughout the present discussion, we have focussed on spell-checkers that ignore the context; consequently, many of the aforementioned results, especially those involving spelling correction, are valid only for context-insensitive spell-checkers. Nevertheless, many of the practically useful spell-checkers incorporate context information and the current analysis on SpellNet can be extended for such spell-checkers by conceptualizing a network of words that capture the word co-occurrence patterns (Biemann, 2006). The word co-occurrence network can be superimposed on SpellNet and the properties of the resulting structure can be appropriately analyzed to obtain similar bounds on hardness of context-sensitive spell-checkers. We deem this to be a part of our future work. Another way to improve the study could be to incorporate a more realistic measure for the orthographic similarity between the words. Nevertheless, such a modification will have no effect on the analysis technique, though the results of the analysis may be different from the ones reported here.

Appendix A: Derivation of the Probability of RWE

We take a *noisy channel approach*, which is a common technique in NLP (for example (Brown et al., 1993)), including spellchecking (Kernighan et al., 1990). Depending on the situation. the channel may model typing or OCR errors. Suppose that a word w, while passing through the channel, gets transformed to a word w'. Therefore, the aim of spelling correction is to find the $w^* \in \Lambda$ (the lexicon), which maximizes $p(w^*|w')$, that is

$$\underset{w \in \Lambda}{argmax}\ p(w|w') = \underset{w \in \Lambda}{argmax}\ p(w'|w)p(w) \tag{9}$$

The likelihood $p(w'|w)$ models the noisy channel, whereas the term $p(w)$ is traditionally referred to as the *language model* (see (Jurafsky and Martin, 2000) for an introduction). In this equation, as well as throughout this discussion, we shall assume a unigram language model, where $p(w)$ is the normalized frequency of occurrence of w in a standard corpus.

We define the probability of RWE for a word w,

$p_{rwe}(w)$, as follows

$$p_{rwe}(w) = \sum_{\substack{w' \in \Lambda \\ w \neq w'}} p(w'|w) \tag{10}$$

Stated differently, $p_{rwe}(w)$ is a measure of the probability that while passing through the channel, w gets transformed into a form w', such that $w' \in \Lambda$ and $w' \neq w$. The probability of RWE in the language, denoted by $p_{rwe}(\Lambda)$, can then be defined in terms of the probability $p_{rwe}(w)$ as follows.

$$\begin{aligned} p_{rwe}(\Lambda) &= \sum_{w \in \Lambda} p_{rwe}(w)p(w) \\ &= \sum_{w \in \Lambda} \sum_{\substack{w' \in \Lambda \\ w \neq w'}} p(w'|w)p(w) \end{aligned} \tag{11}$$

In order to obtain an estimate of the likelihood $p(w'|w)$, we use the concept of *edit distance* (also known as Levenstein distance (Levenstein, 1965)). We shall denote the edit distance between two words w and w' by $ed(w, w')$. If we assume that the probability of a single error (i.e. a character deletion, substitution or insertion) is ρ and errors are independent of each other, then we can approximate the likelihood estimate as follows.

$$p(w'|w) = \rho^{ed(w,w')} \tag{12}$$

Exponentiation of edit distance is a common measure of word similarity or likelihood (see for example (Bailey and Hahn, 2001)).

Substituting for $p(w'|w)$ in Eqn 11, we get

$$p_{rwe}(\Lambda) = \sum_{w \in \Lambda} \sum_{\substack{w' \in \Lambda \\ w \neq w'}} \rho^{ed(w,w')} p(w) \tag{13}$$

References

R. Albert and A. L. Barabási. 2002. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97.

Todd M. Bailey and Ulrike Hahn. 2001. Determinants of wordlikeness: Phonotactics or lexical neighborhoods? *Journal of Memory and Language*, 44:568 – 591.

A. Bhatt, M. Choudhury, S. Sarkar, and A. Basu. 2005. Exploring the limits of spellcheckers: A comparative study in bengali and english. In *Proceedings of the Symposium on Indian Morphology, Phonology and Language Engineering (SIMPLE'05)*, pages 60–65.

C. Biemann. 2006. Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the COLING/ACL 2006 Student Research Workshop*, pages 7–12.

P. F. Brown, S. A. D. Pietra, V. J. D. Pietra, and R. L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312.

M. Choudhury, A. Mukherjee, A. Basu, and N. Ganguly. 2006. Analysis and synthesis of the distribution of consonants over languages: A complex network approach. In *Proceedings of the COLING/ACL Main Conference Poster Sessions*, pages 128–135.

G. Hirst and A. Budanitsky. 2005. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11:87 – 111.

R. Ferrer i Cancho and R. V. Solé. 2004. Patterns in syntactic dependency networks. *Physical Review E*, 69:051915.

D. Jurafsky and J. H. Martin. 2000. *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall.

V. Kapatsinski. 2006. Sound similarity relations in the mental lexicon: Modeling the lexicon as a complex network. *Speech research Lab Progress Report*, 27:133 – 152.

M. D. Kernighan, K. W. Church, and W. A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING*, pages 205–210, NJ, USA. ACL.

K. Kukich. 1992. Technique for automatically correcting words in text. *ACM Computing Surveys*, 24:377 – 439.

V. I. Levenstein. 1965. Binary codes capable of correcting deletions, insertions and reversals. *Doklady Akademii Nauk SSSR*, 19:1 – 36.

M. E. J. Newman. 2003. The structure and function of complex networks. *SIAM Review*, 45:167–256.

M. S. Vitevitch. 2005. Phonological neighbors in a small world: What can graph theory tell us about word learning? Spring 2005 Talk Series on Networks and Complex Systems, Indiana University.

Vertex Degree Distribution for the Graph of Word Co-Occurrences in Russian

Victor Kapustin
Faculty of Philology
Saint-Petersburg State University
Saint-Petersburg, Russia 199178
vak@icape.nw.ru

Anna Jamsen
Faculty of Philology
Saint-Petersburg State University
Saint-Petersburg, Russia 199178
anna_zheleznova@mail.ru

Abstract

Degree distributions for word forms co-occurrences for large Russian text collections are obtained. Two power laws fit the distributions pretty good, thus supporting Dorogovtsev-Mendes model for Russian. Few different Russian text collections were studied, and statistical errors are shown to be negligible. The model exponents for Russian are found to differ from those for English, the difference probably being due to the difference in the collections structure. On the contrary, the estimated size of the supposed kernel lexicon appeared to be almost the same for the both languages, thus supporting the idea of importance of word forms for a perceptual lexicon of a human.

1 Introduction

Few years ago Ferrer and Solé (2001a) draw the attention of researchers to the fact that the lexicon of a big corpus (British National Corpus – BNC –in the case) most probably consists of two major components: a compact kernel lexicon of about 10^3 – 10^4 words, and a cloud of all other words. Ferrer and Solé studied word co-occurrence in BNC in (2001b). Two word forms[1] in BNC were considered as "interacting" when they appeared in the same sentence and the words' distance didn't exceed 2. Ferrer and Solé (2001b) treated also some other notions of word interaction, but the results obtained don't differ qualitatively. The interacting words form a graph, where the vertices are the words themselves, and the edges are the words' co-occurrences. The fact of the collocation considered to be important, not the number of collocations of the same pair of words. Ferrer and Solé (2001b) studied vertices degree distribution and found two power laws for that distribution with a crossover at a degree approximately corresponding to the previously found size of the supposed kernel lexicon of about 10^3 – 10^4 words. In (Solé et al, 2005) word co-occurrence networks were studied for small (about 10^4 lines of text) corpora of English, Basque, and Russian. The authors claim the same two-regime word degree distribution behavior for all the languages.

Dorogovtsev and Mendes (2001, 2003: 151-156) offered an abstract model of language evolution, which provides for two power laws for word degree distribution with almost no fitting, and also explains that the volume of the region of large degrees (the kernel lexicon) is almost independent of the corpus volume. Difference between word (lemma) and word form for an analytic language (e.g. English) seems to be small. Dorogovtsev-Mendes model certainly treats word forms, not lemmas, as vertices in a corpus graph. Is it really true for inflecting languages like Russian? Many researchers consider a word form, not a word (lemma) be a perceptual lexicon unit (Zasorina, 1977; Ventsov and Kassevich, 1998; Verbitskaya et. al., 2003; Ventsov et. al., 2003). So a hypothesis that word forms in a corpus of an inflecting language should exhibit degree distribution similar to that of BNC looks appealing. An attempt to investigate word frequency rank sta-

[1] Strictly speaking, word forms, not words.

TextGraphs-2: Graph-Based Algorithms for Natural Language Processing, pages 89–92,
Rochester, April 2007

tistics for Russian was made by Gelbukh and Sidorov (2001), but they studied only Zipf law on too small texts to reveal the kernel lexicon effects. To study the hypothesis one needs a corpus or a collection[2] of texts comparable in volume with the BNC part that was examined in (Ferrer and Solé, 2001b), i.e. about 4.10^7 word occurrences. Certainly, texts that were analyzed in (Solé et al, 2005) were much smaller.

Recently Kapustin and Jamsen (2006) and Kapustin (2006) studied a big (~5.10^7 word occurrences) collection of Russian. The collection exhibited power law behavior similar to that of BNC except that the vertex degree at the crossover point and the average degree were about 4-5 times less than that of BNC. These differences could be assigned either to a collection nature (legislation texts specifics) or to the properties of the (Russian) language itself. We shall reference the collection studied in (Kapustin and Jamsen, 2006; Kapustin, 2006) as "**RuLegal**".

In this paper we present a study of another big collection of Russian texts. We have found that degree distributions (for different big sub-collections) are similar to those of BNC and of RuLegal. While the exponents and the kernel lexicon size are also similar to those of BNC, the average degree for these collections are almost twice less than the average degree of BNC, and the nature of this difference is unclear still.

The rest of the paper has the following structure. **Technology** section briefly describes the collection and the procedures of building of co-occurrence graph and of calculation of exponents of power laws. In **Discussion** section we compare the results obtained with those of Kapustin and Jamsen (2006), Kapustin (2006), and (Ferrer and Solé, 2001b). In **Conclusion** some considerations for future research are discussed.

2 Technology

At present Russian National Corpus is unavailable for bulk statistical research due to copyright considerations. So we bought a CD ("World Literature in Russian") in a bookstore – a collection of fiction translations to Russian. We'll call the collection **WLR**. The size of the whole collection is more than 10^8 word occurrences. The source format of the collection is HTML, but its files contain essentially no formatting, just plain paragraphs. We made three non-overlapping samples from WLR (WLR1–3). The samples were approximately of the same size. Each sample was processed the same way. The idea behind using more than one sample was to estimate statistical errors.

We used open source Russian grapheme analysis module (Sokirko, 2001) to strip HTML and to split the texts into words and sentences. Word co-occurrences were defined as in (Ferrer and Solé, 2001b): two words are "interacting" if and only if they: (a) appear in the same sentence, and (b) the word distance is either 1 (adjacent words) or 2 (one word or a number or a date in-between). A found co-occurred pair of words was tried out against MySQL database of recorded word pairs, and if it wasn't found in the database, it was put there. Then we use a simple SQL query to get a table of count of vertices **p(k)** vs. vertex degree **k**.

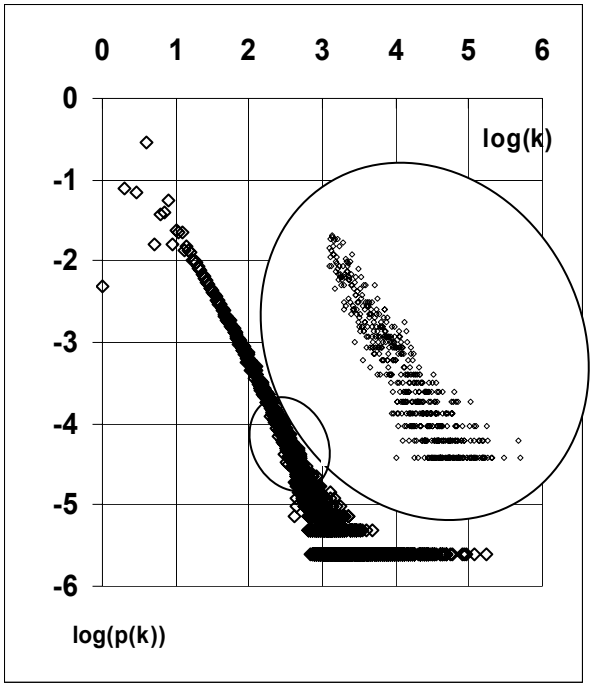


Figure 1. Raw degree distribution for WLR1.

The raw results for one of the samples are shown on Fig. 1. For the two other samples the distributions are similar. All distributions are almost linear (in log-log coordinates, that means that they obey power law), but fitting is impossible due to high fluctuations. As noted by Dorogovtsev and Mendes (2003: 222-223), cumulative distribution

[2] We consider a corpus to be a special type of a text collection, which comprises text samples chosen for language research purposes, while a more general term "collection " refers to a set of full texts brought together for some other purpose.

$P(k) = \Sigma_{K \geq k}\ p(K)$ fluctuates much less, so we calculated the cumulative degree distributions (Fig.2). Cumulative degree distributions for all three WLR samples are very similar.

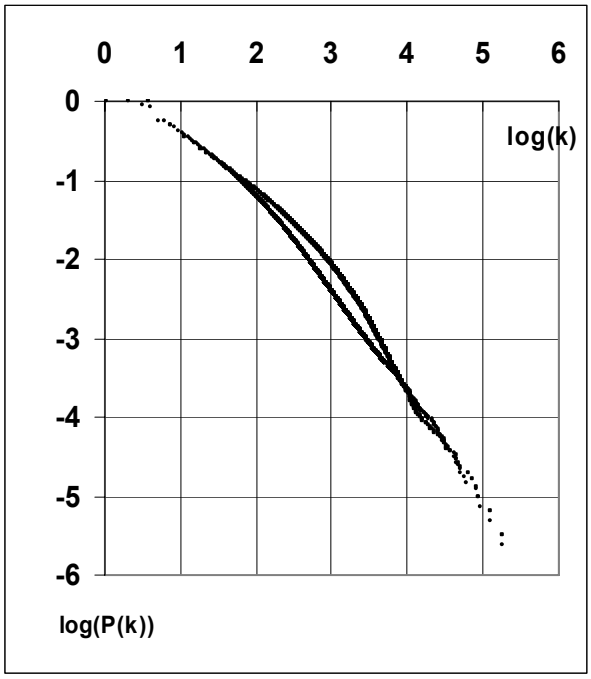


Figure 2. Cumulative degree distributions for WLR1 (lower curve) and RuLegal (upper curve).

3 Discussion

To estimate statistical errors we have normalized the distributions to make them comparable: the degree ranges were reduced to the largest one, then the cumulative degree distribution was sampled with the step of 1/3, as in (Ferrer and Solé, 2001a, Dorogovtsev and Mendes, 2003: 222-223). When we use WLR samples only, the statistical errors are less than 7% in the middle of the curves and reach a margin of 77% in small degrees region. With the inclusion of RuLegal sample, difference between samples becomes larger – up to 13% in the middle of the curves), but are still small enough.

In both cases (with and without RuLegal) we attempted to fit either a single power law (a straight line in log-log coordinates) or two/three power laws with one/two crossover points. Strong changes and large statistical errors of the distributions in the low degree region prevent meaningful usage of these points for fitting. We have made attempts to fit all three approximations for all points, and omitting one or two points with the lowest degrees. To choose between the hypotheses we minimized Schwarz information criterion (Schwarz, 1978):

$$SIC = N * \ln\left(\sum_i (p_i - \hat{p}_i)^2 / N\right) - m * \ln(N)$$

where p_i – cumulative distribution at i-th point;
$\hat{p}_i$ – fitting law at the same point;
N – number of sampling points (13–15, depending on the number of omitted points);
m – number of fitting parameters (2, 4 or 6)

Omitted points	SIC (1/2/3 power laws)	
	WLR1–3	WLR1–3 + RuLegal
0	–44 / –85 / –68	–42 / –93 / –77
1	–46 / **–85** / –65	–43 / **–93** / –73
2	–47 / –80 / –60	–44 / –86 / –67

Table 1. Fitting power laws to averaged degree distributions – Schwarz information criterion

	WLR1–3	WLR1–3 + RuLegal	RuLegal	BNC
γ_1	–0.95	–0.95	–0.95	–0.5
γ_2	–1.44	–1.46	–1.75	–1.7
k_{cross}	670	670	510	2000
V_{kernel}	4.10^3	4.10^3	4.10^3	5.10^3
$k_{average}$	36	31	15	72
Collection size	3.10^7	14.10^7	5.10^7	4.10^7

Table 2. Parameters of the best fit two power laws for the cumulative distributions

Clearly two power laws fit the curves better. The exponents, the crossover degree and estimated size of the kernel lexicon (number of vertices with high degrees above the crossover) for the best fits (two powers, zero/one omitted point) are shown in Table 2. The exponents for the raw distributions are γ_1 and γ_2 minus 1.

Disagreement between English and Russian seems to exist. Probably, the differences are still due to the collections' nature (the difference between different Russian collections is noticeable).

4 Conclusion

We found that ergodic hypothesis for word form degree distribution seems to work for large text collections – differences between the distributions

are small (except for the few smallest degrees). At least, a single big enough sample permits reliable calculation of degree distribution parameters.

Dorogovtsev-Mendes model, which yields two power laws for the degree distribution for the word forms graph, gives pretty good explanation both for an analytic language (English) and for an inflecting one (Russian), though numeric parameters for both languages differ. The estimated sizes of the supposed kernel lexicons for the both languages are almost the same, the fact supports the point that word form is a perceptual lexicon unit.

To make more rigorous statements concerning statistical properties of various languages, we plan to calculate other important characteristics of the co-occurrence graph for Russian: clustering coefficient and average shortest path. Also we hope that legal obstacles to Russian National Corpus usage will have been overcome. Other statistical language graph studies are also interesting; among them are investigation of networks of lemmas, and statistical research of agglutinated languages.

Acknowledgements

The authors are grateful to the anonymous reviewers, the comments of whom were of much help.

The work is supported in part by Russian Foundation for Basic Research, grants 06-06-80434 and 06-06-80251.

References

Sergey N. Dorogovtsev., José F. Mendes, 2001. Language as an evolving word web. *Proceedings of the Royal Society of London B*, 268(1485): 2603-2606

Sergey N. Dorogovtsev., José F. Mendes, 2003. Evolution of Networks: From Biological Nets to the Internet and WWW. Oxford University Press, Oxford.

Ramon Ferrer and Ricard V. Solé. 2001a. Two regimes in the frequency of words and the origin of complex lexicons. *Journal of Quantitative Linguistics* 8: 165-173.

Ramon Ferrer and Ricard V. Solé. 2001b. The Small-World of Human Language. *Proceedings of the Royal Society of London B*, 268(1485): 2261-2266

Victor Kapustin, 2006. Капустин В.А. Ранговые статистики совместной встречаемости словоформ в большой монотематической коллекции. Труды третьей международной конференции «Корпусная лингвистика», 11–13 октября 2006 г., – СПб.: Изд-во С. Петерб. ун-та, 2006. – С. 135-142 (*Rank Statistics of Word Co-Occurrences in a Big Monothematic Collection*. Proc. 3rd International Conf. "Corpus Linguistics", Oct. 11-13, 2006. Saint-Petersburg State Publishing: 135-142).

Alexander Gelbukh and Grigory Sidorov, 2001. *Zipf and Heaps Laws' Coefficients Depend on Language*.. Proc. CICLing-2001, Conference on Intelligent Text Processing and Computational Linguistics (February 18–24, 2001, Mexico City), Lecture Notes in Computer Science, Springer-Verlag. (2004): 332-335. (ISSN 0302-9743, ISBN 3-540-41687-0)

Victor Kapustin and Anna Jamsen. 2006. Ранговая статистика встречаемости слов в большой текстовой коллекции. Труды 8ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» – RCDL'2006, Суздаль, Россия, 2006. – С. 245–251 (*Rank Statistics of Word Occurrence in a Big Text Collection*. Proc. 8th National Russian Research Conference "Digital libraries: advanced methods and technologies, digital collections", Oct. 17-19, 2006: 245-251).

Alexey Sokirko, 2001. *A short description of Dialing Project*. http://www.aot.ru/docs/sokirko/sokirko-candid-eng.html

Ricard V. Solé, Bernat Corominas, Sergi Valverde and Luc Steels. 2005. Language Networks: their structure, function and evolution. SFI-WP 05-12-042, SFI Working Papers

Gideon Schwarz, 1978. *Estimating the dimension of a model*. Annals of Statistics 6(2): 461-464.

Anatoly V. Ventsov and Vadim B. Kassevich, 1998. Венцов А.В., Касевич В.Б. *Словарь для модели восприятия речи*. Вестник Санкт-Петербургского университета, сер. 2, вып. 3, с. 32-39 (*A Dictionary for a Speech Perception Model*. Vestnik Sankt-Peterbugskogo Universiteta, 2(3): 32-39).

Anatoly V. Ventsov, Vadim B. Kassevich and Elena V. Yagounova, 2003. Венцов А.В., Касевич В.Б., Ягунова Е.В. *Корпус русского языка и восприятие речи*. Научно-техническая информация.– Серия 2, № 6, с.25-32 (*Russian Corpus and Speech Perception*. Research and Technical Information, 2(6): 25-32).

Liudmila A. Verbitskaya, Nikolay N. Kazansky and Vadim B. Kassevich, 2003. Вербицкая Л.А., Казанский Н.Н., Касевич В.Б. Некоторые проблемы создания национального корпуса русского языка. Научно-техническая информация.– Серия 2, № 5, с.2-8 (*On Some Problems of Russian National Corpus Development*. Research and Technical Information, 2(5): 2-8).

Lidia N. Zasorina, ed., 1977. *Частотный словарь русского языка*. Под ред. Л.Н. Засориной. М.: Русск. яз. (*Frequency Dictionary of Russian*. Russian Language, Moscow, 1977).

Author Index