Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm

Der Fakultät für Mathematik und Informatik
der Universität Leipzig
eingereichte

DISSERTATION

zur Erlangung des akademischen Grades

DOCTOR rerum naturalium
(Dr. rer. nat.)

im Fachgebiet

Informatik

vorgelegt

von Dipl. Inf. Christian Biemann

geboren am 3. März 1977 in Ingolstadt

Die Annahme der Dissertation haben empfohlen:

1. Prof. Dr. Antal van den Bosch, Universität Tilburg
2. Prof. Dr. Gerhard Heyer, Universität Leipzig
3. Prof. Dr. Stefan Wrobel, Universität Bonn

Die Verleihung des akademischen Grades erfolgt auf Beschluss des Rates der Fakultät für Mathematik und Informatik vom 19 November 2007 mit dem Gesamtprädikat *summa cum laude*.

Dedicated to my parents, who unsupervised me in a good way

Abstract

After almost 60 years of attempts to implement natural language competence on machines, there is still no automatic language processing system that comes even close to human language performance.

The fields of Computational Linguistics and Natural Language Processing predominantly sought to teach the machine a variety of subtasks of language understanding either by explicitly stating processing rules or by providing annotations the machine should learn to reproduce. In contrast to this, *human* language acquisition largely happens in an unsupervised way – the mere exposure to numerous language samples triggers acquisition processes that learn the generalisation and abstraction needed for understanding and speaking that language.

Exactly this strategy is pursued in this work: rather than telling machines how to process language, one instructs them how to discover structural regularities in text corpora. Shifting the workload from specifying rule-based systems or manually annotating text to creating processes that employ and utilise structure in language, one builds an inventory of mechanisms that – once being verified on a number of datasets and applications – are universal in a way that allows their execution on unseen data with similar structure. This enormous alleviation of what is called the "acquisition bottleneck of language processing" gives rise to a unified treatment of language data and provides accelerated access to this part of our cultural memory.

Now that computing power and storage capacities have reached a sufficient level for this undertaking, we for the first time find ourselves able to leave the bulk of the work to machines and to overcome data sparseness by simply processing larger data.

In Chapter 1, the *Structure Discovery* paradigm for Natural Language Processing is introduced. This is a framework for learning structural regularities from large samples of text data, and for making these regularities explicit by introducing them in the data via self-annotation. In contrast to the predominant paradigms, Structure Discovery involves neither language-specific knowledge nor supervision and is therefore independent of language, domain and data

representation. Working in this paradigm rather means to set up discovery procedures that operate on raw language material and iteratively enrich the data by using the annotations of previously applied Structure Discovery processes. Structure Discovery is motivated and justified by discussing this paradigm along Chomsky's levels of adequacy for linguistic theories. Further, the vision of the complete Structure Discovery Machine is sketched: A series of processes that allow analysing language data by proceeding from the generic to the specific. At this, abstractions of previous processes are used to discover and annotate even higher abstractions. Aiming solely to identify structure, the effectiveness of these processes is judged by their utility for other processes that access their annotations and by measuring their contribution in application-based settings.

Since graphs are used as a natural and intuitive representation for language data in this work, Chapter 2 provides basic definitions of graph theory. As graphs built from natural language data often exhibit scale-free degree distributions and the Small World property, a number of random graph models that also produce these characteristics are reviewed and contrasted along global properties of their generated graphs. These include power-law exponents approximating the degree distributions, average shortest path length, clustering coefficient and transitivity.

When defining discovery procedures for language data, it is crucial to be aware of quantitative language universals. In Chapter 3, Zipf's law and other quantitative distributions following power-laws are measured for text corpora of different languages. The notion of word co-occurrence leads to co-occurrence graphs, which belong to the class of scale-free Small World networks. The examination of their characteristics and their comparison to the random graph models as discussed in Chapter 2 reveals that none of the existing models can produce graphs with degree distributions found in word co-occurrence networks.

For this, a generative model is needed, which accounts for the property of language being among other things a time-linear sequence of symbols. Since previous random text models fail to explain a number of characteristics and distributions of natural language, a new random text model is developed, which introduces the notion of sentences in random text and generates sequences of words with a higher probability, the more often they have been generated before. A comparison with natural language text reveals that this model successfully explains a number of distributions and local word order restrictions in a fully emergent way. Also, the co-occurrence graphs of

its random corpora comply with the characteristics of their natural language counterparts. Due to its simplicity, it provides a plausible explanation for the origin of these language universals without assuming any notion of syntax or semantics.

For discovering structure in an unsupervised way, language items have to be related via similarity measures. Clustering methods serve as a means to group them into clusters, which realises abstraction and generalisation. Chapter 4 reviews clustering in general and graph clustering in particular. A new algorithm, Chinese Whispers graph partitioning, is described and evaluated in detail. At cost of being non-deterministic and formally not converging, this randomised and parameter-free algorithm is very efficient and especially suited for Small World graphs. This allows its application to graphs of several million vertices and edges, which is intractable for most other graph clustering algorithms. Chinese Whispers is parameter free and finds the number of parts on its own, making brittle tuning obsolete. Modifications for quasi-determinism and possibilities for obtaining a hierarchical clustering instead of a flat partition are discussed and exemplified. Throughout this work, Chinese Whispers is used to solve a number of language processing tasks.

Chapter 5 constitutes the practical part of this work: Structure Discovery processes for Natural Language Processing using graph representations.

First, a solution for sorting apart multilingual corpora into monolingual parts is presented, involving the partitioning of a multilingual word co-occurrence graph. The method has shown to be robust against a skewed distribution of the sizes of monolingual parts and is able to distinguish between all but the most similar language pairs. Performance levels comparable to trained language identification are obtained without providing training material or a preset number of involved languages.

Next, an unsupervised part-of-speech tagger is constructed, which induces word classes from a text corpus and uses these categories to assign word classes to all tokens in the text. In contrast to previous attempts, the method introduced here is capable of building significantly larger lexicons, which results in higher text coverage and therefore more consistent tagging. The tagger is evaluated against manually tagged corpora and tested in an application-based way. Results of these experiments suggest that the benefit of using this unsupervised tagger or a traditional supervised tagger is equal for most applications, rendering unnecessary the tremendous annotation efforts for creating a tagger for a new language or domain.

A Structure Discovery process for word sense induction and disambiguation is briefly discussed and illustrated.

The conclusion in Chapter 6 is summarised as follows: Unsupervised and knowledge-free Natural Language Processing in the Structure Discovery paradigm has shown to be successful and capable of producing a processing quality equal to systems that are built in a traditional way, if just sufficient raw text can be provided for the target language or domain. It is therefore not only a viable alternative for languages with scarce annotated resources, but might also overcome the acquisition bottleneck of language processing for new tasks and applications.

Acknowledgements

For the successful completion of a scientific work, many factors come together and many pre-conditions have to be met. Even if – like with dissertations – the work is authored by only one person, it is always the case that a work like this is not possible without support by other people. In an acknowledgement section, it is difficult to draw a line between those who had the largest influence and those who cannot be mentioned due to space restrictions. May the latter group be forgiving for that. Especially all my friends that helped me to keep up a good mood are an important part of those unnamed.

First and foremost, I would like to thank my supervisor, Gerhard Heyer, for his outstanding conceptional and institutional support. His judgements on interesting issues versus dead ends and his swift reactions in times of need influenced the quality of this work tremendously and helped me to see interpretations in what otherwise would have remained a harddisk full of numbers.

Further, a great thanks goes to Uwe Quasthoff, who was always available for in-depth discussions about new ideas, algorithms and methodologies. He enormously strenghtend my feeling for language and mathematical formalisms. Also, most of the text data I used was taken from his projects Leipzig Corpora Collection and Deutscher Wortschatz.

For numerous corrections and suggestions on earlier versions of this manuscript, I am indebted to thank Gerhard Heyer, Hans Friedrich Witschel, Uwe Quasthoff, Jutta Schinscholl, Stefan Bordag, Wibke Kuhn and Ina Lauth. Also, I would like to thank countless anonymous reviewers and fellow conference attendees from the scientific community for valuable remarks, suggestions and discussions about NLP topics.

Matthias Richter is acknowledged for help with the typesetting and I thank Renate Schildt for coffee, cigarettes and other office goodies.

Having dedicated this thesis to my parents Gerda and Horst Biemann, I must thank them deeply for bringing me up in a free, but not unguided way, and to open up this whole lot of possibilities to me that enabled me to stay in school and university for such a long time.

Finally, on practical matters and funding, I acknowledge the sup-

port of Deutsche Forschungsgemeinschaft (DFG) via a stipend in the Graduiertenkolleg Wissensrepräsentation, University of Leipzig. This grant covered my expenses for life and travel in a period of three years. I would like to thank Gerhard Brewka and Herrad Werner for supporting me in this framework.

Additionally, this work was funded for four months by MULTILINGUA, University of Bergen, supported by the European Commission under the Marie Curie actions. In this context, my thanks goes to Christer Johansson and Koenraad de Smedt.

Above all, my warmest thanks goes to my love Jutta. She always made my life worth living, never complained about me sometimes being a workaholic, and always managed to distract my attention to the nice things in life. Without her constant presence and support, I would have gone nuts already.

Contents

List of Tables

List of Figures

1 Introduction

> "Linguistics accordingly works continuously with concepts forged by grammarians without knowing whether or not the concepts actually correspond to the constituents of the system of language. But how can we find out? And if they are phantoms, what realities can we place in opposition to them?" (de Saussure, 1966, p. 110)

In the past, Natural Language Processing (NLP) has always been based on the *explicit* or *implicit* use of linguistic knowledge. Explicit rule based approaches prevail in classical linguistic applications, while machine learning algorithms use implicit knowledge for generating linguistic annotations.

The question behind this work is: how far can we go in NLP without assuming any linguistic knowledge? How much effort in annotation and resource building is needed for what level of sophistication in natural language analysis?

1.1 Structure Discovery for Language Processing

Working in what I call the *Structure Discovery* (SD) paradigm, the claim being made here is that the knowledge needed can largely be acquired by knowledge-free and unsupervised methods. By employing knowledge about language universals (cf. Chapter 3) it is possible to construct Structure Discovery processes, which operate on a raw text corpus[1] in iterative fashion to unveil structural regularities in the text and to make them explicit in a way that further SD processes can build upon it.

A frequent criticism on work dealing with unsupervised methods in NLP is the question: "Why not take linguistic knowledge into account?" The simple answer to this is that for many languages and applications, the appropriate linguistic knowledge just is not available. While annotated corpora, classification examples, sets of rules and lexical semantic word nets of high coverage exist for English,

[1] the terms text, text data, corpus, text corpus, language and language data are used interchangeably throughout this work

this does not reflect the situation for most of even the major world languages. Further, handmade and generic resources often do not fit the application domain, whereas resources created *from and for* the target data inherently do not suffer from these discrepancies.

Structure in this context is any kind of automatically acquired annotation that relates elements of language along arbitrary kinds of similarity. It is not restricted to a single language level, but encompasses labels with the scope of e.g. sentences, clauses, words or substrings. Since the processes that discover and mark structural regularities are not allowed to choose from a predefined inventory of labels, the names of the labels are meaningless and receive their interpretation merely through the elements marked by them.

Unlike other works that proceed by teaching the machine directly how to solve certain tasks – be it by providing explicit rules or implicitly by training the machine on handcrafted examples – the topic of this work is the unsupervised, knowledge-free acquisition of structural regularities in language. Unsupervised means that no labelled training material is provided as input. The machine is exposed to language only, without being told what its output should look like. Knowledge-free means that no knowledge about the specific language, such as e.g. word order constraints or a list of personal pronouns, is given to the system. The work of a Structural Discovery engineer is rather to provide a suitable collection of natural language data and to set up procedures that make use of it.

All knowledge about how to conduct this augmentation of structure is encoded procedurally in methods and algorithms. This keeps the paradigm entirely language independent and applicable without further effort to all human languages and sub-languages for which data is available. Given the fact that several thousand human languages are spoken in the world and the ongoing specialisation in science, production and culture, which is reflected in the respective sub-languages or domains, this paradigm provides a cheaper, if not the only way of efficiently dealing with this variety.

Shifting the workload from creating rich resources manually to developing generic, automatic methods, a one-size-fits-all solution needing only minimal adaptation to new domains and other languages comes into reach.

In the remainder of this section, the paradigms followed by the fields of Computational Linguistics (CL) and Natural Language Processing (NLP) are shortly contrasted after laying out the SD paradigm in more detail. Further, the benefits and drawbacks of knowledge-free as opposed to knowledge-intensive approaches,

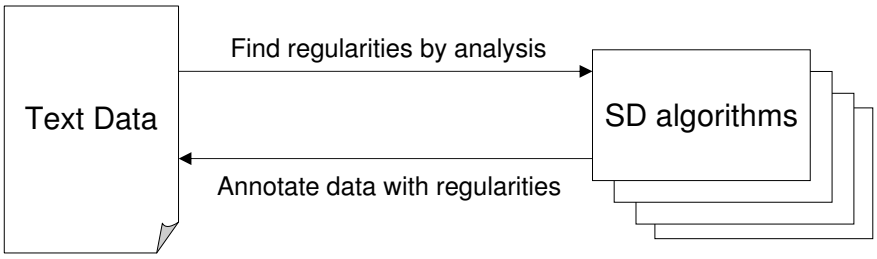


Figure 1.1: Iterative methodology of the Structure Discovery paradigm: SD algorithms find and annotate regularities in text data, which can serve as input for further SD algorithms

as well as degrees of supervision are elaborated on and the SD paradigm is compared to other paradigms.

1.1.1 Structure Discovery Paradigm

The Structure Discovery (SD) paradigm is the research line of algorithmic descriptions that find and employ structural regularities in natural language data. The goal of SD is to enable machines to grasp regularities, which are manifold and necessary in data used for communication, politics, entertainment, science and philosophy, purely from applying operationalised procedures on data samples. Structural regularities, once discovered and marked as such in the data, allow an abstraction process by subsuming structural similarities of basic elements. These complex bricks constitute the building block material for meta-regularities, which may themselves be subject of further exploration.

The iterative methodology of the SD paradigm is laid out in Figure 1.1. It must be stressed that working in the SD paradigm means to proceed in two directions: using the raw data for arriving at generalisations and using these generalisations to structurally enrich the data. While the first direction is conducted by all clustering approaches, the second direction of feeding the results back into the data to perform self-annotation is only rarely encountered.

Unlike other schools that provide knowledge of some sort to realise language processing, a system following the SD paradigm must arrive at an adequate enrichment of language data with the instances of the discovered structures, realising self-annotation of the data, as merely the data in its raw form determines the kind of structures found and instances identified thereof.

Solely working on algorithmically discoverable structures means

to be consequently agnostic to linguistic theories. It is not possible for machines to create proposals for analysis based on intuition and introspection. Rather, the grammarian's knowledge can stimulate discovery process formulation. The native speaker's intuition about particular languages is replaced by a theory-independent intuition of how to discover structure. While as well created intellectually, the utility of the new discovery process can immediately be judged and measured. Further, it is applicable for all data exhibiting similar structure and thus more general.

1.1.2 Approaches to Automatic Language Processing

The discipline of automatically processing natural language is split into two well-established subfields that are aimed at slightly different goals. Computational Linguistics (CL) is mainly influenced by linguistic thinking and aims at implementing linguistic theories following a *rationalist* approach. Focus is set on linguistic problems rather than on robust and efficient processing. Natural Language Processing (statistical NLP, in the definition of Manning and Schütze (1999)), on the other hand, is not linked to any linguistic theory. The goal of NLP is not understanding of language structure as an end in itself. Rather, knowledge of language structure is used to build methods that help in processing language data. The criterion of NLP is the performance of the system, not the adequacy of the representation to human language processing, taking a pragmatic but theoretically poorly founded view. Regarding current research, the two fields are not easily separated, as they influence and fertilise each other, so there is rather a continuum than a sharp cutting edge between the two.

The history of automatic treatment of natural languages was dominated consecutively by two major paradigms: rule-based and statistical approaches. Rule-based approaches are in line with the CL tradition; statistical methods correspond roughly to the NLP view. Starting with the realm of computers, rule-based approaches tackled more and more problems of language processing. The leitmotiv was: given enough time to develop more and more sophisticated rules, eventually all phenomena of language will be encoded. In order to operationalise the application of grammar formalisms, large systems with several thousand grammar rules were built. Since these rules interact with each other, the process of adding sensible rules gets slower the more rules are already present, which makes the construction of rule-based systems expensive. What is inherent of the

top-down, introspection-driven construction of rule-based systems is that they can only work on the subset of language covered by the rules. The past showed that the size of this subset remained far from getting close to full coverage, resulting in only a fraction of sentences being processable.

Since the advent of large machine-readable text corpora starting with the Brown Corpus (Francis and Kučera, 1982) and the computing power necessary to handle them, statistical approaches to language processing received increased interest. By basing decisions on probabilistic models instead of rules, this *empiricist* approach early showed to be capable of reaching higher accuracy on language processing tasks than the rule-based approach. The manual labour in statistical methods is shifted from instructing the machines directly by rules how to process the data to labelling training examples that provide information on how a system's output should look like. At this point, more training means a richer basis for the induction of probabilistic models, which in turn leads to better performance.

For understanding language from a linguistic point of view and testing grammar theories, there does not seem to be another way than the rule-based approach. For building applications, however, statistical methods proved to be more robust and faster to set up, which is probably best contrasted for Machine Translation by opposing Martin Kay's essay "Machine Translation: The Disappointing Past and Present" (Kay, 1997) to Franz Josef Och's talk "Statistical Machine Translation: The Fabulous Present and Future" (Och, 2005).

This work is clearly rooted at the NLP end of the line, as knowledge-free methods by definition do not use theoretic results achieved by a linguist's introspection, but go a step further by not even allowing implicit knowledge to be provided for training.

1.1.3 Knowledge-intensive and Knowledge-free

Another scale, along which it is possible to classify language processing methods, is the distinction between knowledge-intensive and knowledge-free approaches, see also (Bordag, 2007). Knowledge-intensive approaches make excessive use of language resources such as dictionaries, phrase lists, terminological resources, name gazetteers, lexical-semantic networks such as WordNet (Miller *et al.*, 1990), thesauri such as Roget's Thesaurus (Roget, 1852), ontologies and alike. As these resources are necessarily incomplete (all resources leak), their benefit will cease to exist beyond a certain point and additional coverage can only be reached by substantial enlarge-

ment of the resource, which is often too much of a manual effort.

But knowledge-intensiveness is not only restricted to explicit resources: the rules in a rule-based system constitute a considerable amount of knowledge just as positive and negative examples in machine learning.

Knowledge-free methods seek to eliminate human effort and intervention. The human effort is not in specifying rules or examples, but in the method itself, lending the know-how by providing discovery procedures rather than presenting the knowledge itself. This makes knowledge-free methods more adaptive to other languages or domains, overcoming the brittleness of knowledge-intensive systems when exposed to an input substantially different from what they were originally designed for.

Like above, there rather is a continuum than a sharp border between the two ends of the scale. Methods that incorporate only little human intervention are sometimes labelled knowledge-weak, combining the benefits of not having to prepare too much knowledge with obtaining good results by using available resources.

1.1.4 Degrees of Supervision

Another classification of methods, which is heavily related to the amount of knowledge, is the distinction between supervised, semi-supervised, weakly supervised and unsupervised methods.

- In *supervised* systems, the data as presented to a machine learning algorithm is fully labelled. That means: all examples are presented with a classification that the machine is meant to reproduce. For this, a classifier is learned from the data, the process of assigning labels to yet unseen instances is called classification.

- In *semi-supervised* systems, the machine is allowed to additionally take unlabelled data into account. Due to a larger data basis, semi-supervised systems often outperform their supervised counterparts using the same labelled examples (see (Zhu, 2005) for a survey on semi-supervised methods and (Sarkar and Haffari, 2006) for a summary regarding NLP and semi-supervision). The reason for this improvement is that more unlabelled data enables the system to model the inherent structure of the data more accurately.

- Bootstrapping, also called self-training, is a form of learning that is designed to use even less training examples, therefore

sometimes called *weakly-supervised*. Bootstrapping (see (Biemann, 2006a) for an introduction) starts with a few training examples, trains a classifier, and uses thought-to-be positive examples as yielded by this classifier for retraining. As the set of training examples grows, the classifier improves, provided that not too many negative examples are misclassified as positive, which could lead to deterioration of performance.

- *Unsupervised* systems are not provided any training examples at all and conduct clustering. This is the division of data instances into several groups, (see (Manning and Schütze, 1999, Ch. 14) and (Berkhin, 2002) for an overview of clustering methods in NLP). The results of clustering algorithms are data driven, hence more 'natural' and better suited to the underlying structure of the data. This advantage is also its major drawback: without a possibility to tell the machine what to do (like in classification), it is difficult to judge the quality of clustering results in a conclusive way. But the absence of training example preparation makes the unsupervised paradigm very appealing.

To elaborate on the differences between knowledge-free and unsupervised methods, consider the example of what is called unsupervised word sense disambiguation (cf. Yarowsky, 1995). Word sense disambiguation (WSD) is the process of assigning one of many possible senses to ambiguous words in the text. This can be done supervisedly by learning from manually tagged examples. In the terminology of Senseval-3 (Mihalcea and Edmonds, 2004), an unsupervised WSD system decides the word senses merely based on overlap scores of the word's context and a resource containing word sense definitions, e.g. a dictionary or WordNet. Such a system is unsupervised, as it does not require training examples, but knowledge-intensive due to the provision of the lexicographic resource.

1.1.5 Contrasting Structure Discovery with Previous Approaches

To contrast the paradigm followed by this work with the two predominant paradigms of using explicit or implicit knowledge, Table 1.1 summarises their main characteristics.

Various combinations of these paradigms lead to hybrid systems, e.g. it is possible to construct the rules needed for CL by statistical methods, or to build a supervised standard NLP system on top of an unsupervised, knowledge-free system, as conducted in Section 5.2.9.

Paradigm	CL	NLP	SD
Approach	rule-based	statistics	statistics
Direction	top-down	bottom-up	bottom-up
Knowledge Source	manual resources	manual annotation	–
Knowledge Intensity	knowledge-intensive	knowledge-intensive	knowledge-free
Degree of Supervision	unsupervised	supervised	unsupervised
Corpus required	–	annotated text	raw text

Table 1.1: Characteristics of three paradigms for the computational treatment of natural language

As already discussed earlier, semi-supervised learning is located in between statistical NLP and the SD paradigm.

1.2 Relation to General Linguistics

Since the subject of examination in this work is natural language, it is inevitable to relate the ideas presented here to linguistics. Although this dissertation neither implements one of the dominating linguistic theories nor proposes something that would be dubbed 'linguistic theory' by linguists, it is still worthwhile looking at those ideas from linguistics that inspired the methodology of unsupervised natural language processing, namely linguistic structuralism and distributionalism. Further, the framework shall be examined along desiderata for language processing systems, which were formulated by Noam Chomsky.

Serving merely to outline the connection to linguistic science, this section does by no means raise a claim for completeness on this issue. For a more elaborate discussion of linguistic history that paved the way to the SD paradigm, see (Bordag, 2007).

1.2.1 Linguistic Structuralism and Distributionalism

Now, the core ideas of linguistic structuralism will be sketched and related to the SD paradigm. Being the father of modern linguistics, de Saussure (1966) introduced his negative definition of meaning: the signs of language (think of linguistic elements such as words for the remainder of this discussion) are solely determined by their relations to other signs, and not given by a (positive) enumeration of

characterisations, thereby harshly criticising traditional grammarians. This is to say, language signs arrange themselves in a space of meanings and their value is only differentially determined by the value of other signs, which are themselves characterised differentially.

De Saussure distinguishes two kinds of relations between signs: syntagmatic relations that hold between signs in a series in present (e.g. neighbouring words in a sentence), and associative relations for words in a "potential mnemonic series" (de Saussure, 1966, p. 123). While syntagmatic relationships can be observed from what de Saussure calls *langage* (which corresponds to a text corpus in terms of this work), all other relationships subsumed under 'associations' are not directly observable and can be individually different. A further important contribution to linguistic structuralism is attributed to Zelling Harris (1951, 1968), who attempts to discover some of these associative or paradigmatic relations. His *distributional hypothesis* states that words of similar meanings can be observed in similar contexts, or as popularised by J. R. Firth: "You shall know a word by the company it keeps!" (Firth, 1957, p.179)[2]. This quote can be understood as the main theme of *distributionalism*: determining the similarity of words by comparing their contexts.

Distributionalism does not look at single occurrences, but rather at a word's distribution, i.e. the entirety of contexts (also: global context) it can occur in. The notion of context is merely defined as language elements related to the word; its size or structure is arbitrary and different notions of context yield different kinds of similarities amongst the words sharing them. The consequences of the study of Miller and Charles (1991) allow to operationalise this hypothesis and to define the similarity of two signs as a function over their global contexts: the more contexts two words have in common, the more often they can be exchanged, and the more similar they are. This immediately gives rise to discovery procedures that compare linguistic elements according to their distributions, as e.g. conducted in Chapter 5.

Grouping sets of mutually similar words into clusters realises an abstraction process, which allows generalisation for both words and contexts via class-based models (cf. Brown *et al.*, 1992). It is this mechanism of abstraction and generalisation, based on contextual

[2]Ironically, Firth did not mean to pave the way to a procedure for statistically finding similarities and differences between words. He greatly objected to de Saussure's views and clearly preferred the study of a restricted language system for building a theory, rather than using discovery procedures for real, unrestricted data. In his work, the quote refers to assigning correct meanings to words in habitual collocations.

clues, that allows the adequate treatment and understanding of previously unseen words, provided their occurrence in well-known contexts.

1.2.2 Adequacy of the Structure Discovery Paradigm

This section aims at providing theoretical justification for bottom-up discovery procedures as employed in the SD paradigm. This is done by discussing them along the *levels of adequacy* for linguistic theories, set up in (Chomsky, 1964), and examining to what extent this classification applies to the procedures discussed in this work. For this, Chomsky's notions of linguistic theory and grammar have to be sketched briefly. For Chomsky, a (generative) grammar is a formal device that can generate the infinite set of grammatical (but no ungrammatical) sentences of a language. It can be used for deciding the grammaticality of a given sentence (see also Chomsky, 1957). A linguistic theory is the theoretical framework, in which a grammar is specified. Chomsky explicitly states that linguistic theory is only concerned with grammar rules that are identified by introspection, and rejects the use of discovery procedures of linguistic regularities from text corpora, since these are always finite and cannot, in his view, serve as a substitute for the native speaker's intuitions.

Having said this, Chomsky provides a hierarchy of three levels of adequacy to evaluate grammars.

- A grammar with *observational adequacy* accounts for observations by exhaustive enumeration. Such "item-and-arrangement grammars" (Chomsky, 1964, p. 29) can decide whether a given sentence belongs to the language described by the grammar or not, but does not provide any insights into linguistic theory and the nature of language as such.

- A higher level is reached with *descriptive adequacy*, which is fulfilled by grammars that explain the observations by rules that employ "significant generalizations that express underlying regularities of language" (Chomsky, 1964, p. 63).

- *Explanatory Adequacy* is the highest level a grammar can reach in this classification. Grammars on this level provide mechanisms to choose the most adequate of competing descriptions, which are equally adequate on the descriptive level. For this, "it aims to provide a principled basis, independent of any particular language" (Chomsky, 1964, p. 63).

According to Chomsky, the levels of descriptive and explanatory adequacy can only be reached by linguistic theories in his sense, as only theoretic means found by introspection based on the native speaker's intuition can perform the necessary abstractions and meta-abstractions. Criticising exactly this statement is the topic of the remainder of this section.

When restricting oneself to a rule-based description of universal language structure like Chomsky, there does not seem to be any other option than proceeding in an introspective, highly subjective and principally incomplete way: as already Sapir (1921, p. 38) stated: "all grammars leak", admitting the general defection of grammar theories to explain the entirety of language phenomena. Especially when proceeding in a top-down manner, the choice "whether the concept [an operational criterion] delimits is at all close to the one in which we are interested" (Chomsky, 1964, p. 57) is subjective and never guaranteed to mirror linguistic realities. But when abolishing the constraint on rules and attributing explanatory power to bottom-up discovery procedures, the levels of adequacy are also applicable to the framework of SD.

Admitting that operational tests are useful for soothing the scientific conscience about linguistic phenomena, Chomsky errs when he states that discovery procedures cannot cope with higher levels of adequacy (Chomsky, 1964, p. 59). By using clustering procedures as e.g. described in (Brown *et al.*, 1992) and in Chapters 4 and 5, abstraction and generalisation processes are realised that employ the underlying structure of language and thus serve as algorithmic descriptions of language phenomena. Further, these class-based abstractions allow the correct treatment of previously unobserved sentences, and a system as described in Section 5.2 would clearly attribute a higher probability (and therefore acceptability) to the sentence "colorless green ideas sleep furiously" than to "furiously sleep ideas green colorless" based on transition probabilities of word classes (examples taken from (Chomsky, 1964, p. 57)).

Unlike linguistic theories, the systems equipped with these discovery procedures can be evaluated either directly by examination or indirectly by performance within an application. It is therefore possible to decide for the most adequate, best performing discovery procedure amongst several available ones. Conducting this for various languages, it is even possible to identify to what extent discovery procedures are language independent, and thus to arrive at explanatory power, which is predictive in that sense that explanatory adequate procedures can be successfully used on previously unseen

languages.

The great advantage of discovery procedures is, that once defined algorithmically, they produce abstractions that are purely based on the data provided, being more objective and conclusive than rules found by introspection. While simply not fitting in the framework of linguistic theories, they are not phantoms but realities of language, so a complete description of language theory should account for them, see also (Abney, 1996) on this issue.

In terms of quantity and quality, the goals of linguistic theory and unsupervised discovery procedures are contrary to one another. Linguistic theory aims at accounting for most types of phenomena irrespective of how often these are observed, while application-based optimisation targets at an adequate treatment of the most frequent phenomena. Therefore, in applying unsupervised discovery procedures, one must proceed quantitatively, and not qualitatively, which is conducted in this work at all times.

1.3 Similarity and Homogeneity in Language Data

1.3.1 Levels in Natural Language Processing

Since the very beginning of language studies, it is common ground that language manifests itself by interplay of various levels. The classical level hierarchy in linguistics, where levels are often studied separately, is the distinction of (see e.g. Grewendorf *et al.*, 1987) the phonological, morphological, syntactic, semantic and pragmatic level. Since this work is only dealing with digitally available written language, levels that have to do with specific aspects of spoken language like phonology are not considered here. Merely considering the technical data format for the text resources used here, basic units and levels of processing in the SD paradigm are (cf. Heyer *et al.*, 2006):

- character level
- token level
- sentence level
- document level

The character level is concerned with the alphabet of the language. In case of digitally available text data, the alphabet is defined by the encoding of the data and consists of all valid characters, such as letters, digits and punctuation characters. The (white)space character

is a special case, as it is used as delimiter of tokens. Characters as the units of the character level form words by concatenation. The units of the token level are tokens, which roughly correspond to words. Hence, it is not at all trivial what constitutes a token and what does not. The sentence level considers sentences as units, which are concatenations of tokens. Sentences are delimited by sentence separators, which are given by full stop, question mark and exclamation mark. Tokenisation and sentence separation are assumed to be given by a pre-processing step outside of the scope of this work. Technically, the tools of the Leipzig Corpora Collection (LCC, (Quasthoff *et al.*, 2006)) were used throughout for tokenisation and sentence separation.

A clear definition is available for documents, which are complete texts of whatever content and length, i.e. web pages, books, newspaper articles etc.

When starting to implement data-driven acquisition methods for language data, only these units can be used as input elements for SD processes, since these are the only directly accessible particles that are available in raw text data.

It is possible to introduce intermediate levels: morphemes as subword units, phrases as subsentential units or paragraphs as subdocument units. However, these and other structures have to be found by the SD methodology first, so they can be traced back to the observable units.

Other directly observable entities, such as hyperlinks or formatting levels in web documents, are genre-specific levels that might also be employed by SD algorithms, but are not considered for now.

1.3.2 Similarity of Language Units

In order to group language units into meaningful sets, there must be a means to compare them and assign similarity scores for pairs of units. Similarity is determined by two kinds of features: *internal features* and *contextual features*. Internal features are obtained by only looking at the unit itself, i.e. tokens are characterised internally by the letters and letter combinations (such as character N-grams) they contain, sentences and documents are internally described with the tokens they consist of. Contextual features are derived by taking the context of the unit into consideration.

The context definition is arbitrary and can consist of units of the same and units of different levels. For example, tokens or character N-grams are contextually characterised by other tokens or character

N-grams preceding them, sentences are maybe similar if they occur in the same document, etc. Similarity scores based on both internal and contextual features require exact definition and a method to determine these features, as well as a formula to compute the similarity scores for pairs of language units.

Having computational means at hand to compute similarity scores, language units can eventually be grouped into homogeneous sets.

1.3.3 Homogeneity of Sets of Language Units

While the notion of language unit similarity provides a similarity ranking of related units with respect to a specific unit, a further abstraction mechanism is needed to arrive at classes of units that can be employed for generalisation. Thus, it is clear that a methodology is needed for grouping units into meaningful sets. This is realised by clustering, which will be discussed in depth in Chapter 4. In contrast to the similarity ranking centred on a single unit, a cluster consists of an arbitrary number of units. The advantage is that all cluster members can be subsequently subsumed under the cluster, forming a new entity that can give rise to even more complex entities.

Clustering can be conducted based on the pair wise similarity of units, based on arbitrary features. In order to make such clustering and at the same time generalisation processes successful, the resulting sets of units must exhibit *homogeneity* in some dimensions. Homogeneity here means that the cluster members agree in a certain property, which constitutes the abstract concept of the cluster. Since similarity can be defined along many features, it is to be expected that different dimensions of homogeneity will be found in the data, each covering only certain aspects, e.g. on syntactic, semantic or morphological levels. Homogeneity is, in principle, a measurable quantity and expresses the plausibility of the grouping. In reality, however, this is very often difficult to quantify, thus different groupings will be judged on their utility in the SD paradigm: on one hand by their ability to generalise in a way that further structure can be discovered with these abstract concepts forming the building blocks, on the other hand by their utility as features in task-driven applications.

1.4 Vision: The Structure Discovery Machine

The remainder of this chapter is dedicated to an outline of the ultimate goal of SD: a set of algorithmic procedures that encompasses

the discovery of the entirety of structure that can be discovered in a data-driven way. Viewing the practical results in Chapter 5 as being only a starting point, I will now envision a number of capabilities of such a system and discuss them along the usual pipeline of processing steps for language processing. The 'traditional' terms shall serve as an aid for imagination – without doubt, the discovery procedures will not reproduce theoretically pure sub-levels, as indicated above. Nevertheless, linguistic theory can play the role of a source of what is necessary and which phenomena are to be accounted for.

When exhibited to language data, the Structure Discovery Machine (SDM) identifies the basic word and sentence units it will operate on and finds a suitable representation – no matter whether the SDM is exposed to text, speech or other encodings of language. Already here, an abstraction process is involved that groups e.g. different phonetic variants in speech or different character sets in written language. Then, different languages are identified in the stream and the corresponding parts are grouped (see Section 5.1). In a similar way, domain-specific subsets of the monolingual material are marked as such, e.g. by techniques as employed in document clustering (see (Steinbach *et al.*, 2000) for an overview).

For each language, a syntactic model is set up that encompasses parts of speech (cf. Section 5.2) for basic word units, chunking to phrasal units of several words (as in e.g. Cohen *et al.*, 2007) and syntactic dependencies between basic and complex units[3] such as in (Klein, 2005; Bod, 2006; Olney, 2007, inter al.). This requires a procedure handling derivation, inflection and compounding of units, realised by a component similar to those compared in the MorphoChallenge (Kurimo *et al.*, 2006). Contextual analysis of content-bearing units allows to hypothesise different meanings for units with semantically ambiguous use and to disambiguate their occurrences (cf. Section 5.3). Having circumnavigated the pitfalls of lexical ambiguity, units are classified according to their semantic function and relation. Semantic classes have been previously learned in (e.g. Cardie and Weischedel, 1997; Berland and Charniak, 1999; Thelen and Riloff, 2002); for semantic relations, (Turney, 2006) shows ways how to extract them from massive corpora. The works of Davidov and Rappoport (2006); Davidov *et al.* (2007) illustrate how to extract sets of semantically similar words and their typical relations from web data. Reoccurring relations between units of similar or different semantic function will be marked as such, serving as something similar

[3] cf. approaches to parsing, such as HPSG (Pollard and Sag, 1994), LFG (Horn, 1983) and dependency parsing (Nivre, 2006)

to FrameNet (Baker *et al.*, 2003) annotations. Coreference has been successfully learnt in an unsupervised way by (Haghighi and Klein, 2007).

In its highest and purest form, the SDM will not even be engineered by humans plugging together discovery procedures based on their intuitions, but will self-assemble by trying out interplays of a parameterisable inventory of procedures, thus optimising the overall structural description in terms of generalisation power. This, however, is a much larger project than could be treated by a single dissertation at the time being and raises several yet unanswered research questions.

Output of the SDM is a multidimensional annotation of the raw input data with labels that denote abstractions of structurally similar phenomena. Some kinds of annotations are orthogonal to each other, others can be arranged hierarchically and many will be dependent on other annotations. The scope of annotations is not limited, ranging from the most basic to the most complex units. This holistic approach to data-driven self-annotation rather overgenerates and is not restricted to a small set of structural phenomena. To determine which generalisations are useful, these have to be tested for utility in task-based evaluations.

Applications of these annotations are twofold: firstly, they can be employed as features in machine learning for a task-based adaptation: stemming, parts-of-speech tagging, chunking, named entity recognition, information extraction and parsing in their present form will greatly benefit from this rich inventory of features, which will reduce the necessary amount of training instances significantly. Overlapping representations of complex facts give rise to mapping queries to answers in a Question Answering system (as e.g. conducted in (Ramakrishnan *et al.*, 2004)), which is currently regarded as one of the next steps to enhance today's search engines. Since the names and values of the features are irrelevant in this setting besides their ability to grasp the differentiation granularity needed for the task, the annotations generated by the SDM can be easily incorporated and evaluated in such an application-based way. Feature selection mechanisms (cf. Kohavi and John, 1997, inter al) allow to choose those annotations that correlate best with the task-defined structural regularities.

Secondly, when examining what annotations are the most important for solving language processing tasks and tracing back their creation to their data-driven origin, one will find that the procedures of the SDM provide insights into the system of constituents of natural

language and could play the role of de Saussure's desperately sought realities in the initial quote of this chapter.

2 Graph Models

This chapter provides basic definitions of graph theory, which is a well-established field in mathematics, dealing with properties of graphs in their abstract form. Graph models are a way of representing information by encoding it in vertices and edges. In the context of language processing, vertices will denote language units, whereas edges represent relations between these. This way, units and their similarities are naturally and intuitively translated into a graph representation.

After revisiting notions of graph theory in Section 2.1, the focus is set on large-scale properties of graphs occurring in many complex systems, such as the Small World property and scale-free degree distributions. A variety of random graph generation models exhibiting these properties on their graphs will be discussed in Section 2.2.

The study of large-scale characteristics of graphs that arise in Natural Language Processing using graph representations are an essential step towards approaching the data, in which structural regularities shall be found. Structure Discovery processes have to be designed with awareness about these properties. Examining and contrasting the effects of processes that generate graph structures similar to those observed in language data sheds light on the structure of language and their evolution.

2.1 Graph Theory

2.1.1 Notions of Graph Theory

This section contains basic definitions of graph theory that will be necessary in all later chapters. The notation follows (Bollobas, 1998).

Graph, Vertices, Edges

A graph G is a pair of finite sets (V, E) such that E is a subset of unordered pairs of $V \times V$. V is the set of vertices, E is the set of edges. If G is a graph, then $V = V(G)$ is the vertex set of G, and $E = E(G)$ is the edge set. An edge $\{x, y\}$ is said to join the vertices x and y and is denoted by xy, also $e(x, y)$. Thus, xy and yx mean

exactly the same edge, x and y are called endpoints of that edge, x and y are said to be adjacent vertices. Equivalently, it is said that an edge connects x and y. In the following, the graphs are restricted to having no self-loops (irreflexive), i.e. edges with the same vertex for both endpoints are not allowed, if not explicitly stated otherwise.

Subgraph

A subgraph $G(S)$ of a graph G is induced by a set of vertices $S \subset V(G)$ and contains all edges of G that connect $s_i, s_j \in S$. The set of edges of $G(S)$ is denoted by $E(S)$.

Order and Size

The order of G is the number of vertices in G, denoted by $|G|$, also written $|V(G)|$. Here, $|.|$ is the size of a set. The size of G is the number of edges, denoted by $|E(G)|$. We write G^n for a graph of order n. $G(n, m)$ denotes a graph of order n and size m.

Subsets of Vertices

Considering two disjoint subsets U and W of the vertex set of a graph $V(G)$, the set of edges joining vertices from U and W is written $E(U, W)$. The number of edges in $E(U, W)$ is $|E(U, W)|$.

Neighbourhood of a Vertex

The neighbourhood of a vertex $v \in V(G)$ consists of the vertices adjacent to v, that is $neigh(v) = \{x | \{v, x\} \in E(G)\}$. Sometimes this is called open neighbourhood in contrast to the closed neighbourhood formed by $neigh(v) \cup \{v\}$.

Path

A path is a graph P of the form $V(P) = \{x_0, x_1, ..., x_l\}$, $E(P) = \{x_0x_1, x_1x_2, ..., x_{l-1}x_l\}$. The path is denoted by $x_0x_1..x_l.$, l is the length of the path P. P is said to be a path from x_0 to x_l or an $x_0 - x_l$ path.

Distance

The distance $d(x, y)$ between vertices x and y is the minimal length of an $x - y$ path. If no such path exists, $d(x, y) = \infty$. The distance between a vertex and itself is $d(x, x) = 0$.

Component

A graph is connected, if for every pair $\{x, y\}$ of distinct vertices $x \in V(G)$, $y \in V(G)$, $x \neq y$ there exists a path from x to y. A maximal connected subgraph of G is called a component of G.

Partition

A partition of a graph $G(V, E)$ is a set of disjoint sets of vertices $\{P_1, P_2, ...P_n\}$ with $P_i \subset V$ such that for all $i, j \in \{1..n\}$, $i \neq j$: $P_i \cap P_j = \emptyset$ and $\bigcup_{i=1..n} P_i = V$. The sets P_i are called parts, also clusters. These two terms are used interchangeably in the remainder of this work.

Cut

Any subset of vertices $S \subset V$ creates a cut, which is a partition of V into two disjoint subsets S and $V \setminus S$. The size of a cut S of graph G is defined as $c_G(S) = e(S, V \setminus S)$. and measures the number of edges that have to be eliminated in order to obtain the two components S and $V \setminus S$ from G.

A cutvertex is a vertex whose deletion increases the number of components. An edge is called bridge, if its deletion increases the number of components.

Bipartite Graph

A graph G is bipartite with vertex subsets V_1 and V_2 if $V(G) = V_1 \cup V_2, V_1 \cap V_2 = \emptyset$ and every edge joins a vertex from V_1 and a vertex from V_2. Similarly, G is r-partite with vertex classes $V_1, V_2, ...V_r$, if $V(G) = \bigcup_{1..r} V_i$ and $V_i \cap V_j = \emptyset$ for all $i, j \in \{1..n\}$, $i \neq j$, and no edge joins vertices of the same vertex part.

Complete Graph

A graph of order n and size $\binom{n}{2}$ is called a complete n-graph and is denoted by K^n (recall that self-loops are excluded). In K^n, all possible pairs of vertices are adjacent to each other. K^n is also called *n-clique*.

Directed Graph

If the edges are ordered pairs of vertices (x, y) instead of sets $\{x, y\}$, the graph is called directed. An ordered pair (x, y) is said to be a

directed edge from x to y, or beginning at x and ending at y, denoted by xy.

Edge and Vertex Weight

A graph G is called edge-weighted if there exists a function $ew : E(G) \rightarrow \mathbb{R}^+$ that assigns an edge weight w_{xy} to every edge $\{x, y\} \in E(G)$. $\{x, y\}$ is said to have the weight $ew(x, y) = w_{xy}$. In analogy, G is vertex-weighted, if there is a function $vw : V(G) \rightarrow \mathbb{R}^+$ that assigns a vertex weight $vw(v)$ to each vertex $v \in V(G)$. Unweighted (also: simple) graphs are a special case of weighted graphs with all vertex and edge weights set to 1.

Size of Cut in Weighted Graphs

The size of a cut $c_G w(S)$ induced by a subset of vertices $S \subset V$ in a weighted graph is defined as the sum of edge weights that cross cluster boundaries: $c_G w(S) = \sum_{a \in S} \sum_{b \in V \setminus S} ew(a, b)$. In case of $ew(a, b) \notin E$, $ew(a, b) = 0$.

Edge and Vertex Type

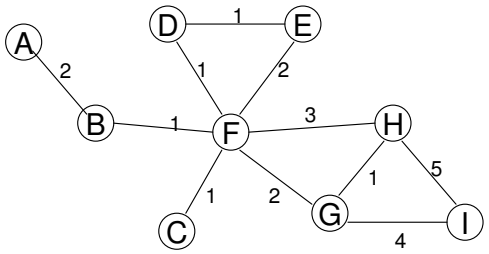
A graph G is edge-typed, if there exists a function $et : E(G) \rightarrow S$ with S set of edge types that assigns a type to every edge. A graph is vertex-typed, if there is a function $vt : V(G) \rightarrow T$ with T set of vertex types, such that every vertex $v \in V(G)$ is assigned a type from T. The entirety of vertex type assignments is called vertex-typisation. A vertex typisation of G induces a partition of $G : V(G) = V_1 \cup .. \cup V_n, V_i \cap V_j = \emptyset$ for all $i, j \in \{1..n\}$, $i \neq j$, and for all $x \in V(G), y \in V(G)$: if $x \in V_i$ and $y \in V_i$, then $vt(x) = vt(y)$.

Adjacency Matrix

The adjacency matrix of a graph G is a matrix A_G associated with G where $(a_{ij}) = 1$ if there exists an edge between vertices v_i and v_j, $(a_{ij}) = 0$ otherwise. For edge-weighted graphs, $(a_{ij}) = ew(i, j)$.

Geometrical Representation

A more intuitive way of specifying a graph is to depict it in a geometrical representation. Figure 2.1 illustrates various concepts of



Geometrical representation

a_{ij}	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*
A	0	2	0	0	0	0	0	0	0
B	2	0	0	0	0	1	0	0	0
C	0	0	0	0	0	1	0	0	0
D	0	0	0	0	1	1	0	0	0
E	0	0	0	1	0	2	0	0	0
F	0	1	1	1	2	0	2	3	0
G	0	0	0	0	0	2	0	1	4
H	0	0	0	0	0	3	1	0	5
I	0	0	0	0	0	0	4	5	0

Adjacency matrix

G(V,E)

V={A,B,C,D,E,F,G,H,I}

E={AB,BF,CF,DE,DF,EF,FG,FH,GH,GI,HI}

ew(BF)=ew(CF)=ew(DE)=ew(DF)=ew(GH)=1;

ew(AB)=ew(EF)=ew(FG)=2; ew(FH)=3;

ew(GI)=4; ew(HI)=5;

Set representation

Figure 2.1: Example for representing a graph in geometrical, adjacency matrix and set representation. B and F are cutvertices, AB, BF and CF are bridges. A further possible representation, the incidence matrix that stores vertices per edges, is not shown

graph theory in different notations. Notice that a graphical representation defines a graph unambiguously, but there are many possible graphical representations that define the same graph.

2.1.2 Measures on Graphs

There exist a variety of measures that characterise certain properties of graphs. The overview here is by no means complete and merely defines measures that are needed at a later stage. For an exhaustive treatment of graph measures the reader is referred to (Merris (2001), inter al.).

Measures on graphs can be divided into local and global measures. Local measures characterise single vertices. In global measures, characteristics of the graph as a whole are combined into a single coefficient or into a distribution.

Degree, Indegree, Outdegree

The number of edges including v, called degree of vertex v, is $k(v) = |neigh(G)|$. A vertex of degree 0 is called isolated, also singleton. For vertices x of directed graphs, the indegree $k_{in}(x)$ is defined as the number of edges ending at x, the outdegree $k_{out}(x)$ is the number of edges starting with x.

The average vertex degree $< k >$ of graph $G(V, E)$ is obtained by averaging over the degrees of all vertices: $< k >= \frac{\sum_{v \in V} k(v)}{|V|}$. Notice that a distinction between average indegree and average outdegree is superfluous.

Shortest Path Length and Diameter

The shortest path length between two vertices v and w is defined as the distance $d(v, w)$ between v and w. An efficient way to compute the shortest path length is the *Dijkstra-algorithm*, see (Dijkstra, 1959).

The average shortest path length L of a graph $G(V, E)$ is given by averaging the shortest path lengths over all vertex pairs in the same component. For components $V_1, ..., V_n$, it is defined as:

$$L = \frac{\sum_{v \in V} \sum_{w \in V, d(v,w) < \infty} d(v, w)}{\sum_{i=1..n} |V_i| \cdot (|V_i| - 1)}$$

The diameter D of a graph $G(V, E)$ is defined as the length of the longest of all shortest paths of G: $D = \max_{v \in V, w \in V, d(v,w) < \infty} d(v, w)$.

Clustering Coefficient

Introduced by Watts and Strogatz (1998), the vertex clustering coefficient c for a vertex v with $k(v) \geq 2$ in graph G measures the connectivity of vertices in the neighbourhood of v:

$$c(v) = \frac{2 \cdot |\{\{u, w\} \in E(G), u \in neigh(v), w \in neigh(v)\}|}{|neigh(v)| \cdot (|neigh(v)| - 1)}.$$

The clustering coefficient $C(G)$ of graph G is defined as the average vertex clustering coefficient:

$$C(G) = \frac{1}{|V'|} \sum_{v \in V, k(v) \geq 2} c(v)$$

where V' is the set of vertices with degree $\geq$2. Vertices with degree 1 or 0 are not taken into account.

Transitivity

The transitivity $T(G)$ of a graph G as defined by Newman *et al.* (2002) is the number of fully connected triplets of vertices divided by the total number of vertex triplets:

$$T(G) = \frac{\sum_{v \in V} \delta(v)}{\sum_{v \in V} \binom{k(v)}{2}}$$

with $\delta(v) = |\{\{u, w\} \in E \text{ and } \{v, u\} \in E \text{ and } \{v, w\} \in E\}|$.

Relation between Clustering Coefficient and Transitivity

Despite other claims, transitivity is not equivalent to the clustering coefficient, only for graphs where all vertices have the same degree or the same local clustering coefficient, see (Schank and Wagner, 2004). This originates from the fact that $C(G)$ is computed by averaging over the vertex clustering coefficients, with each vertex contributing equally to $C(G)$. For $T(G)$, vertices with higher degrees take part in more vertex triplets and thus contribute more in relative terms to $T(G)$. In (Schank and Wagner, 2004), a weighted clustering coefficient that equals transitivity is proposed: here, vertices v are weighted by the number of "possible opposite edges", that is $k(v)(k(v) - 1)/2$. The study further discusses cases of graphs where $C(G)$ and $T(G)$ differ considerably. For the random graph models as discussed in Section 2.2, the most interesting case is a high value for $C(G)$ coupled with a low value of $T(G)$. Figure 2.2 exemplifies a graph where $C \to 1$ and $T \to 0$ as $n \to \infty$. Less extreme cases for high $C(G)$ and low $T(G)$ are characterised by low vertex clustering coefficients for vertices with high degrees.

For determining $C(G)$ and $T(G)$, directed graphs are transformed into undirected graphs. Throughout this work, the efficient implementation as described in (Schank and Wagner, 2005) is used for computation of these measures.

Degree Distribution

The degree distribution $P(k)$ of graph $G(V, E)$ is a probability distribution that provides the probability of choosing a vertex with degree k when randomly selecting a vertex from V with uniform probability. The indegree and outdegree distributions are defined in analogy. When plotting the degree distribution as in Section 2.2, the number of vertices per degree is depicted. This can be transformed into the degree distribution by normalising with $|V|$.

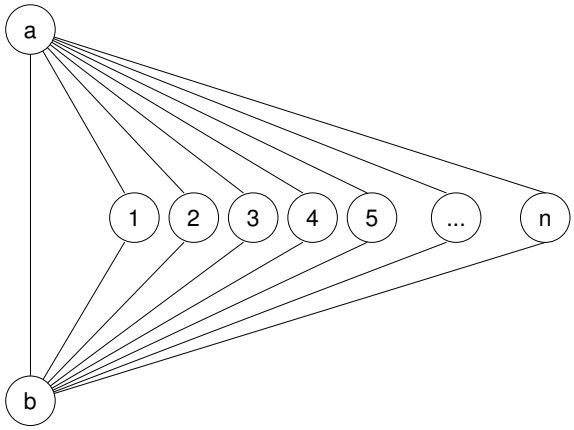


Figure 2.2: Example of a graph with $C \rightarrow 1$ and $T \rightarrow 0$ as $n \rightarrow \infty$. Figure adopted from (Schank and Wagner, 2004)

Symbol	Explanation
n	number of vertices
L	average shortest path length
D	diameter of the graph
C	clustering coefficient
T	transitivity
k, k_{in}, k_{out}	the degree, the in-degree, and out-degree
$P(k), P(k_{in}), P(k_{out})$	degree distributions
$< k >$	average degree

Table 2.1: Symbols referring to graph measures as used in this work

Component Size Distribution

The component size distribution $Comp(x)$ of graph $G(V, E)$ is a probability distribution that provides the probability of choosing a component of size x when randomly selecting a component from V with uniform probability. When plotting the component size distribution, the number of components per size is depicted. This can be transformed into the component size distribution by normalising.

Table 2.1 summarises the symbols used throughout this chapter, largely following the notation of Steyvers and Tenenbaum (2005).

The graph in Figure 2.1 has the following values for the characteristics introduced: $n = 9$, $L = 2$, $D = 4$, $C = 0.638095$, $T = 0.45$,$< k >= \frac{22}{9}$ and the degree distribution is given in Table 2.2.

Degree k	1	2	3	4	5	6
Vertices per degree	2	4	2	0	0	1
$P(k)$	$\frac{2}{9}$	$\frac{4}{9}$	$\frac{2}{9}$	0	0	$\frac{1}{9}$

Table 2.2: Degree distribution for graph in Figure 2.1

2.2 Random Graphs and Small World Graphs

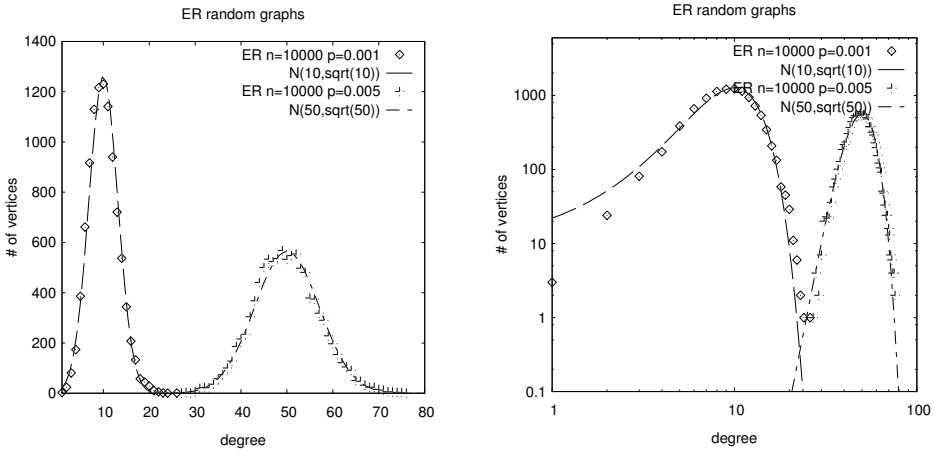
The previous section provided definitions and measures that characterise graphs. Now we turn to the question, how graphs can be generated and how properties of the generation process influence the characteristics of the resulting graphs. As graphs can be seen as abstractions of entities and their relations, a generative process is concerned with describing how new entities and relations are added to existing graphs in a systematic way. Different generation principles lead to graphs with different characteristics, and various graph generation models have been proposed in order to yield graphs that resemble observations on real-world data. Now, the most prominent graph generation models in the literature are reviewed. The connection to language as the matter of interest will be indicated, but carried out more thoroughly in Chapter 3.

2.2.1 Random Graphs: Erdős-Rényi Model

The earliest and most basic model of random graph generation in graph theory is the Erdős-Rényi-model (ER-model, Erdős and Rényi (1960)). The generation process starts with n vertices and no edges. Each of the possible number of $\frac{n(n-1)}{2}$ undirected vertex pairs gets connected by an edge with probability p. The higher p, the denser gets the graph, with the vertex degrees forming a binomial distribution around their mean $< k >= p(n-1)$.

For comparing graph models, the measures as defined in Section 2.1.2 are employed. Figure 2.3 shows the characteristics of two ER random graphs with 10,000 vertices and $p = 0.001$ ($p = 0.005$). Not surprisingly, the values for transitivity and clustering coefficient hardly differ, as most values for vertex degrees $k(v)$ are close to their mean and the vertex clustering coefficient is directly dependent on p and not on $k(v)$.

The ER model triggered the theory of random graphs. Virtually all studies in this field before the mid-nineties of the 20th century are based on this model, see (Bollobas, 1985) for an extensive overview. As shall be clear soon, however, the ER model does not capture the



Characteristic	ER $p = 0.001$	ER $p = 0.005$
n	10,000	10,000
L	4.261	2.770
D	7	4
C	9.242E-4	5.010E-3
T	9.939E-4	5.018E-3
$< k >$	10	50

Figure 2.3: Characteristics of two ER random graphs. The degree distributions are given in a linear scale and in a log-log plot and follow a binomial distribution, which is approximated by a normal distribution $N(< k >, \sqrt{< k >})$ in the plots

characteristics of many graph abstractions of natural phenomena. Another class of graphs that comes closer to what is observed in natural and artificial networks are Small World graphs (SWGs), which will be discussed throughout the remainder of this section.

2.2.2 Small World Graphs: Watts-Strogatz Model

The first report on Small World phenomena was released by Milgram (1967), who performed his well-known experiment using the social network graph between people: addressing 60 letters to a stockbroker living in Boston, he instructed various recruits in Omaha, Nebraska, to merely hand the letter to an acquaintance they would think who could most likely know the receiver. Surprisingly, after six transfers in average, a part of the letters had reached their destination, which was popularised as "six degrees of separation": anyone knows all people on earth over six connections (at least within the USA). In subsequent years, this experiment was repeated under various conditions, confirming this claim. Observations on social networks (as e.g. the entirety of acquaintances in Milgram's experiment) showed, that their corresponding graphs are characterised by a high clustering coefficient C while retaining similar or slightly larger values of L as compared to a Erdős-Rényi random graph with similar numbers of vertices and edges.

This could only be obtained in the Erdős-Rényi model by a probability p that would render us all 'famous party animals': low values of L in hand with high values of C can only be realised using high p values in graph generation, leading to a large $< k >$.

To account for this discrepancy, SWGs were first defined by Watts and Strogatz (1998), where a number of interesting graphs are described as having the following property: $n \gg k \gg ln(n) \gg 1$, where $k \gg ln(n)$ guarantees that the graph is connected. They provide a rewiring procedure that constructs a SWG by rewiring edges of a ring lattice with all vertices having the same degree (cf. Figure 2.4) with a certain probability to randomly chosen vertices. When incrementing p, the characteristic path length L drops dramatically, since shortcuts are introduced in the network. The high clustering coefficient C in the regular ring lattice graph drops very slowly with higher p, resulting in the desired characteristics for SWGs.

Since their discovery, graphs with Small World structure have been observed in graphs corresponding to data as diverse as the topology of food webs, electrical power grids, cellular and metabolic networks, the World Wide Web, the internet backbone, neural networks,

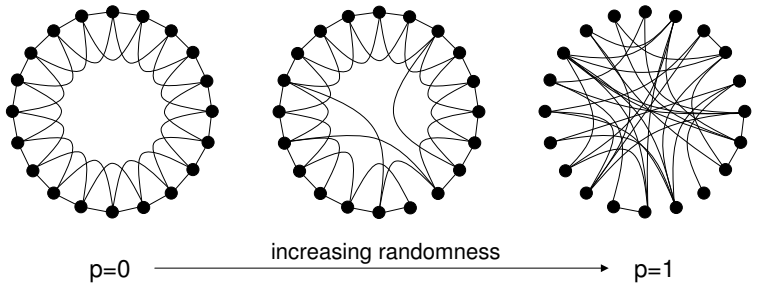


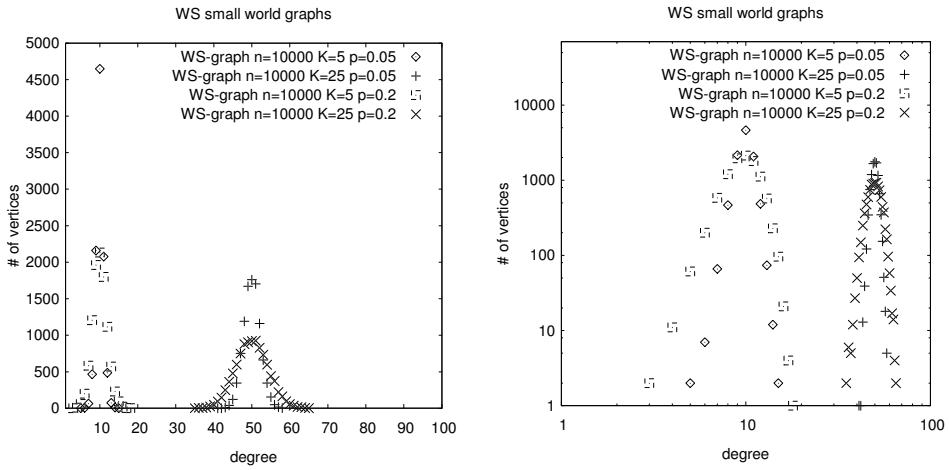
Figure 2.4: Building a SWG by rewiring edges of a regular ring lattice graph (figure adopted from Watts and Strogatz (1998))

telephone call graphs, co-authorship and citation in networks of scientists, movie actor networks and overlapping boards of large companies, see (Strogatz, 2001) for reference. The omnipresence of SWGs in networks whose growth is not governed by a centralised organisation but emerge 'naturally' indicates that understanding their formation processes unveils general principles of nature.

The resulting SWGs are proposed as more accurate models for a large number of self-organising systems, as clustering coefficients are higher while retaining similar L values (cf. Figure 2.3). Figure 2.5 shows characteristics of WS-graphs. Again, transitivity and clustering coefficient highly agree for the same reasons as for the ER model.

2.2.3 Preferential Attachment: Barabási-Albert Model

Starting from the observation that in many self-organising systems the degree distribution decays in a power-law, following $P(k) \sim k^{-\gamma}$ (γ = slope of the degree distribution when plotted on log-log scale), Barabási and Albert (1999) introduce their graph growing model. Unlike in the ER and WS models, where the number of vertices is fixed in advance, the Barabási-Albert (BA) model starts with a small number of vertices and no edge and iteratively introduces new vertices to the graph, connecting them to a fixed number of existing vertices with a probability based on the existing vertex' degree. Increasing the probability of connection to 'popular' vertices (i.e. vertices with high degrees) is called *preferential attachment*. Barabási and Albert (1999) show that preferential attachment makes the difference between an exponentially decreasing degree distribution and a power-law distribution with $\gamma = 3$ for the standard BA-model as proven in (Bollobas *et al.*, 2001). Further, they give indications how to obtain scale-free SWGs with different power-law exponents. Figure



Charact.	K=5 p=0.05	K=5 p=0.2	K=25 p=0.05	K=25 p=0.2
n	10,000	10,000	10,000	10,000
L	6.148	4.671	3.186	2.852
D	10	7	5	4
C	0.4953	0.1789	0.5417	0.1957
T	0.4859	0.1679	0.5396	0.1933
$< k >$	10	10	50	50

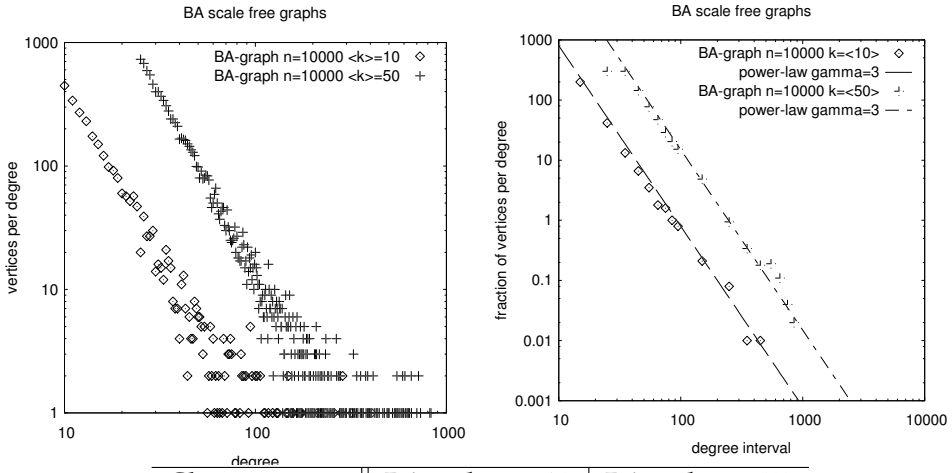
Figure 2.5: Characteristics of two graphs generated by the WS model in linear and log-log plot. The degree distribution decreases exponentially with k, as in the ER-model

2.6 shows the characteristics of BA-modelled graphs. Also in graphs generated by the BA model, the values for transitivity and clustering coefficient are about similar, yet very low as compared to the WS model.

Graphs with a power-law degree distribution are called scale-free for the following reason: in scale-free graphs, there is a significant number of vertices with very high degrees, called *hubs*, whereas in SWGs with an exponential tail of $P(k)$, no such hubs are observed. The scale-free-ness manifests itself in the non-existence of a characteristic vertex degree, i.e. all degrees are present at the same strength, the number of edges for groups of vertices of similar degree is approximately equal. Formally, the degree distribution $P(k) = Ak^{-\gamma}$ remains unchanged to within a multiplying factor when k is multiplied with a scaling factor, i.e. $P(ax) = bP(x)$. This can be seen in Figure 2.6, where the characteristics of graphs generated by the BA-model are depicted. In the figure the exact degree distribution is given on the left side. To obtain a smoother plot, the x-axis was divided into intervals exponentially increasing in size, and the fraction of vertices per degree is given in the figure on the right. For the remainder, this process called *logarithmic binning* is carried out for most figures depicting degree distributions.

2.2.4 Aging: Power-laws with Exponential Tails

Amaral *et al.* (2000) observe three kinds of Small World networks in nature: scale-free graphs as characterised by a power-law distribution of vertex degrees, broad-scale graphs as power-law distributed vertex connectivity with a sharp cut-off, and single-scale graphs with faster decaying tails of the degree distribution. These three different varieties of SWGs can be produced by introducing a limiting factor when adding new edges. In the BA-model, an early vertex can obtain an unlimited number of edges as the generation process goes along. In the real world that we seek to model, however, this might be impossible: whereas in a citation graph, a seminal paper can be cited by many following works without any problem, the capability of people to know a large number of acquaintances is limited naturally by their lifetime. Dependent on the limit, the scale-free property might hold for a certain range of connectivity. These findings are exemplified on the movie-actor network (broad-scale as the number of movies per actor is limited yet can reach a high number) and on high school friendships (single-scale because of a smaller number of participants). The authors provide a methodology of generating graphs



Characteristic	BA $< k >= 10$	BA $< k >= 50$
n	10,000	10,000
L	3.663	2.655
D	6	4
C	7.0716E-3	0.02244
T	5.4071E-3	0.02205
$< k >$	10	50
γ	3	3

Figure 2.6: The degree distributions of two undirected BA-graphs form a straight line in the log-log plots, indicating that $P(k)$ follows a power-law distribution $P(k) \sim k^{-3}$. Notice that the clustering coefficients are higher than in an ER model graph of the same size and order, but lower than in the graphs generated by the WS-model

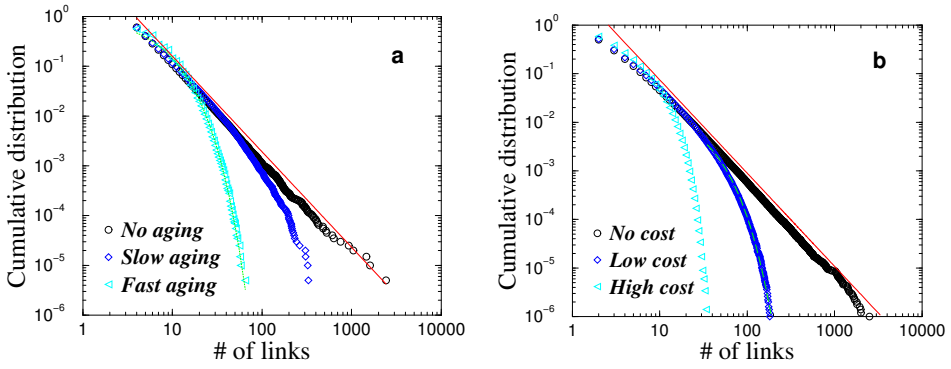


Figure 2.7: Dependent on aging (a) or cost for highly connected vertices (b), the SWG generation model of Amaral *et al.* (2000) produces scale-free, broad-scale and single-scale SWGs (figure taken from (Amaral *et al.*, 2000))

of all three varieties by introducing the concept of aging for vertices or alternatively adding cost to connections to vertices proportional to the vertex' degree into the BA-model. Figure 2.7 shows the characteristic degree distributions of these three varieties in log-log plots generated by various strengths of aging respectively cost.

2.2.5 Semantic Networks: Steyvers-Tenenbaum Model

Exploring the structure of semantic networks, which now links this section to language data, is the topic of (Steyvers and Tenenbaum, 2005). The authors analyse the graph structures of a human association network, Roget's Thesaurus (Roget, 1852) and WordNet (Miller *et al.*, 1990), finding that all three resources exhibit the characteristics of a scale-free SWG. As they observe much higher clustering coefficients C than predicted by the BA-model, they propose their own network growing algorithm, henceforth called ST-model. It generates scale-free SW graphs in the following way: we start with a small number of fully connected vertices. When adding a new vertex, an existing vertex u is chosen with a probability proportional to its degree. The new vertex is connected to M vertices in the neighbourhood of u. The generative model is parameterised by the number of vertices n and the network's mean connectivity, which approaches $2M$ for large n. Steyvers and Tenenbaum propose a directed and an undirected variant of this model and show high agreement with the characteristics of the semantic networks they examine. The directed variant is obtained from the undirected one by creating a directed

edge from the new vertex with a probability d and to the new vertex with a probability $(1 - d)$.

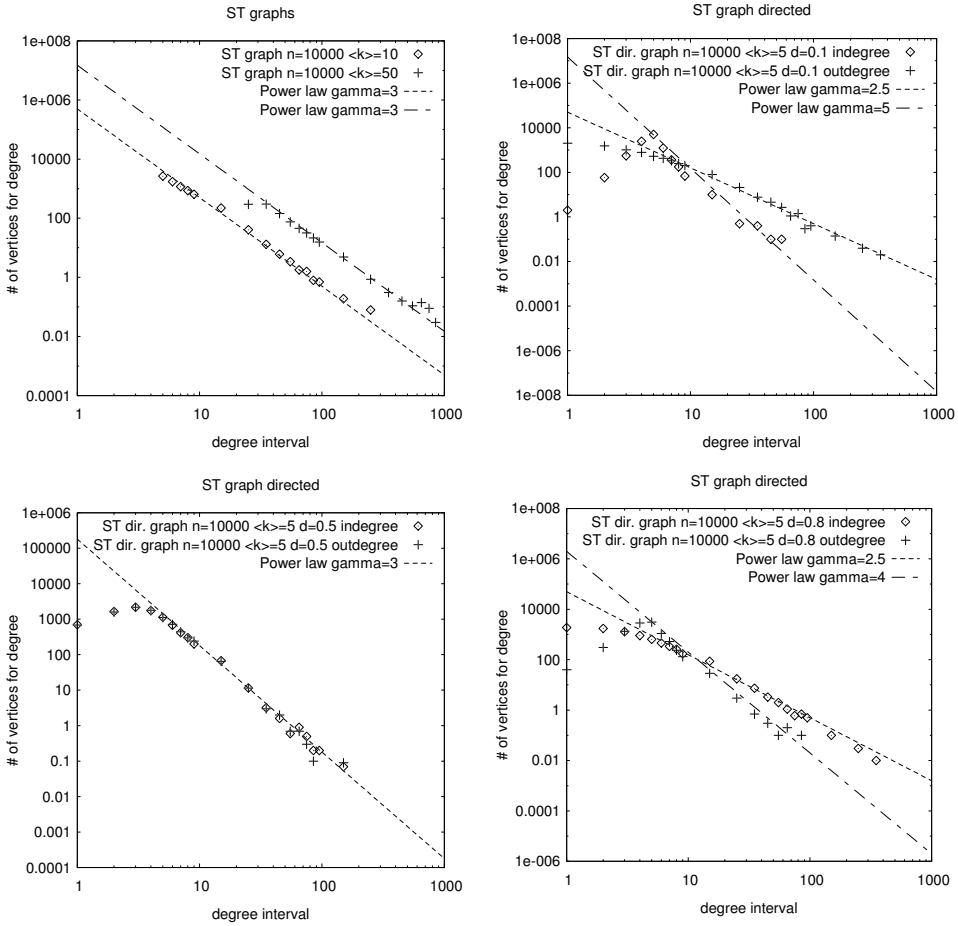
The main difference between the ST-model and the BA-model is that the ST-model enforces a high clustering coefficient by attaching all connections of a new vertex in the same neighbourhood. Figure 2.8 shows characteristics of graphs generated by the ST model.

In graphs generated by the ST-model, the transitivity values were measured at about one third of the clustering coefficient of the same graph, which indicates that the vertex clustering coefficients for vertices with high degrees are lower than those of low degrees. This follows from the construction principle: vertices in the neighbourhood of high degree vertices, which have themselves most probably a high degree, get linked to many new vertices, which are themselves interlinked scarcely. Therefore, the vertex clustering coefficient of the high degree vertices is lower than for newer, low-degree vertices, which mostly connect to highly interlinked old vertices.

2.2.6 Changing the Power-Law's Slope: (α, β) Model

The BA-model and the ST-model both produce power-law degree distributions with a slope of $\gamma = 3$, which is an invariant of their models rather than of SWGs in nature (see e.g. examples in Section 3.2 for language data). A more flexible model, called the (α, β)-model, is defined by Kumar *et al.* (1999). In this growth model for directed SWGs, α is a parameter for specifying the slope of the in-degree distribution $\gamma_{in} = 1/(1 - \alpha)$, β is its analogue for the out-degree distribution $\gamma_{out} = 1/(1 - \beta)$, with $\alpha, \beta \in [0, 1)$. However, slopes of $\gamma < 2$ cannot be generated, as determined empirically and proven for all preferential linking methods in (Dorogovtsev and Mendes, 2001b). A further contribution of Dorogovtsev and Mendes (2001b) is to explain power-law slope variation by a comparison between growth in the number of vertices and growth in the number of edges: if the total number of edges increases faster than the number of vertices – at this increasing the average degree – the exponent of the degree distribution deviates from $\gamma = 3$.

The (α, β)-model is motivated by the observation that the web graph contains a high number of bipartite cores, which are sets of pages that can be separated into two groups with many links between these groups. These structures emerge in the following building process: each time a new vertex v is created, a fixed number of edges is created as well following this method: two random numbers $r_1, r_2 \in [0, 1]$ are drawn. If r_1 falls within the interval $[0, \alpha]$, the



Charact.	M=5	M=25	M=5 d=0.1	M=5 d=0.5	M=5 d=0.8
dir.	undir.	undir.	dir.	dir.	dir.
n	10,000	10,000	10,000	10,000	10,000
L	4.123	2.697	6.262	5.231	5.798
D	8	4	16	11	13
C	0.2529	0.1024	0.2551	0.2639	0.2570
T	0.0845	0.0522	0.0826	0.0882	0.0870
$< k >$	10	50	5	5	5
γ_{in}/γ_{out}	3	3	5/2.5	3	2.5/4

Figure 2.8: Characteristics of graphs generated by the ST model

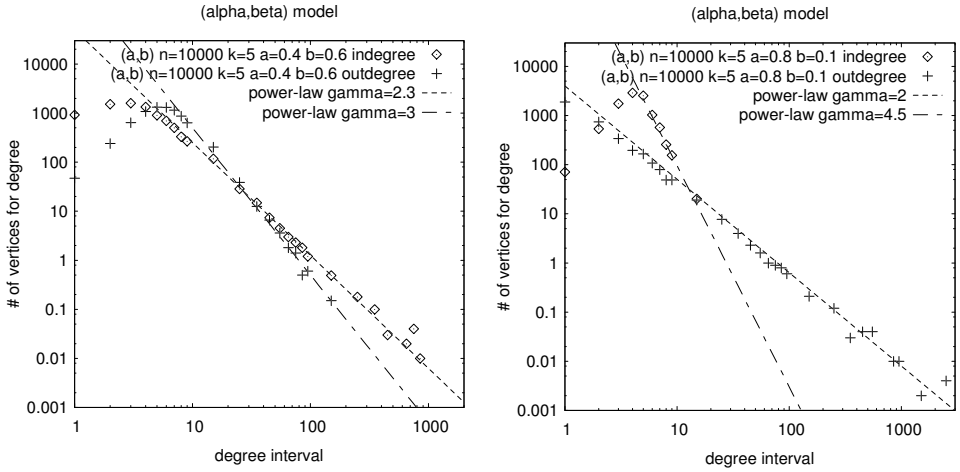
destination of the edge is the new vertex v, otherwise the destination is the destination of a randomly chosen edge. The source of the edge is determined using r_2: if r_2 is in $[0, \beta]$, then the source is v, otherwise it is the source of a randomly chosen edge.

Notice that randomly choosing edges in case of $r_1 > \alpha$ ($r_2 > \beta$) realises preferential attachment, as vertices with higher in-degree (out-degree) more likely become the destination (source) of the new edge. With the possibility to tailor the degree distributions directly to arbitrary slopes in $[2, \infty)$, the (α, β)-model captures the degree distributions of the web graph. This is the only graph generation model discussed here that is reflexive, i.e. edges are allowed to have the same vertex as source and destination. When measuring characteristics, these edges are ignored. The average degree $< k >$ is dependent on the values of α and β. Figure 2.9 shows characteristics for graphs with different α and β values. As a consequence of the construction process, a constant fraction of vertices with both $k_{in} = 0$ and $k_{out} = 0$ can be found in graphs generated by the (α, β)-model. For low values of either α or β, the power-law's slope for the undirected version of the graph is steep. With no mechanism in the construction process that connects vertices adjacent to vertices with high degrees, the discrepancy between C and T is very large.

2.2.7 Two Regimes: Dorogovtsev-Mendes Model

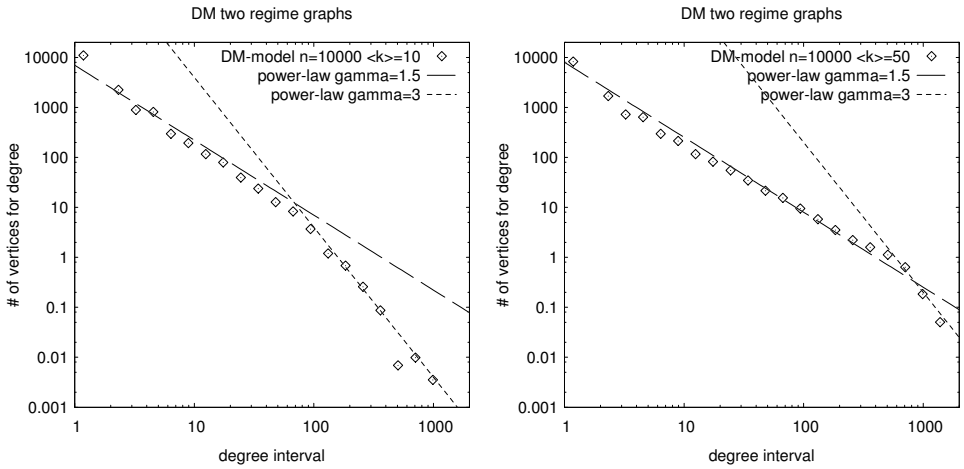
In co-occurrence networks of natural language, sometimes a degree distribution that follows power-laws with two different exponents is observed. The Dorogovtsev-Mendes (DM) model, proposed in (Dorogovtsev and Mendes, 2001a) captures this effect in a generation process for undirected graphs. A DM-generated graph is built in the following way: at each time step t, a new vertex is introduced and connected to one present vertex by preferential attachment, i.e. with a probability proportional to the old vertex' degree. This step is equal to the BA-model with $< k >= 2$. But in difference to the BA-model, ct edges between old vertices are introduced in every time step t between previously not connected old vertices i and j, with a probability according to the product of their vertex degrees $k(i) \cdot k(j)$.

It has been shown in (Dorogovtsev and Mendes, 2001a) that the average degree is dependent on the time step and is given by $< k >= 2 + ct$. For $1 \ll ct$, the degree distribution is governed by two power-laws with $\gamma_1 = 1.5$ for low degrees and $\gamma_2 = 3$ for high degrees, with a crossover point at $k_{cross} =\approx \sqrt{ct}(2 + ct)^{1.5}$. This is achieved by mixing two different growth rates of edges as compared to vertices (see



Characteristic	(α, β) $\alpha = 0.4$ $\beta = 0.6$	(α, β) $\alpha = 0.8$ $\beta = 0.1$
n	10,000	10,000
L	3.9640	6.1567
D	9	17
C	0.0426	0.3007
T	0.0217	3.962E-3
$< k >$	8.163	4.313

Figure 2.9: Characteristics of graphs generated by the (α, β)-model. For this directed model, in-degree and out-degree distributions are given separately.



Characteristic	DM $c = 0.0008$	DM $c = 0.0048$
n	10,000	10,000
L	3.4918	2.8432
D	8	5
C	0.0804	0.3715
T	0.0511	0.2679
$< k >$	10	50

Figure 2.10: Characteristics of graphs generated by the DM-model. Two power-law regimes can be observed. The higher the average degree $< k >$, the higher is the crossover degree k_{cross}

Dorogovtsev and Mendes, 2001b): a constant growth for edges involving the newly introduced vertices and an increasing growth rate for edges between old vertices. Dorogovtsev and Mendes successfully modelled the word web as described by Ferrer-i-Cancho and Solé (2001): word tokens represent vertices and edges are introduced if their corresponding endpoints are found in adjacent positions in the underlying text corpus. These *word co-occurrence graphs* will be subject to deeper exploration in Section 3.2.

Figure 2.10 displays the degree distribution and the characteristics of graphs generated by the DM-model. The DM-model generates scale-free Small World graphs with two power-law regimes. Values for C and T are much higher than in a BA-model graph. As opposed to the ST model, a higher $< k >$ in the DM-model leads to higher C and T values. Throughout, transitivity T is measured somewhat lower than the clustering coefficient C.

Model	ER	WS	BA	ST	(α, β)	DM
undirected	yes	yes	yes	yes	no	yes
directed	no	no	no	yes	yes	no
L	$\approx$5	$\approx$6	$\approx$4	$\approx$4	$\approx$4	$\approx$4
D	$\approx$7	$\approx$7	$\approx$6	$\approx$8	$\approx$9	$\approx$8
C	very low	high	low	high	high	high
T	very low	high	low	low	low	high
$C \lessgtr T$	$C = T$	$C = T$	$C = T$	$C \gg T$	$C \gg T$	$C > T$
$< k >$	10	10	10	10	$\approx$8	10
$P(k)$ tail	exp.	exp.	pl $\gamma = 3$	pl $\gamma = 3$ / $\gamma \in [2.5, \infty)$	pl $\gamma \in [2, \infty)$	pl $\gamma_1 = 1.5$ $\gamma_2 = 3$

Table 2.3: Comparison of graph models: ER-model, WS-model, BA-model, ST-model, (α,β)-model and DM-model for graphs with n=10,000 and $< k >= 10$. Approximate values are marked by $\approx$, pl denotes power-law

2.2.8 Further Remarks on Small World Graph Models

In (Aiello *et al.*, 2000), no generation model is provided, but statements about the component size distribution of scale-free graphs for different degree distribution slopes γ are proven. While for $\gamma < 1$, the graph is almost surely fully connected, a γ between 1 and 2 produces one very large component and some small components of constant size. The range of $2 < \gamma < 3.4785$[1] delimits a family of graphs with small components in a size order of the logarithm of the number of vertices, and $\gamma > 3.4785$ yields graphs with component size distributions following a power-law as well.

To compare the graph models discussed so far, Table 2.3 contrasts their characteristics. The ST-model, the (α, β)-model and the DM-model generate scale-free SWGs. Graphs generated by the ER-model and the BA-model exhibit a small clustering coefficient, ER-model and WS-model graphs are not scale-free.

Up to this point, only simple, i.e. unweighted models of SWGs have been discussed. As graph representations of natural phenomena are often weighted, extensions to weighted graphs are shortly mentioned in the following.

Vertex weights correspond to the importance of the represented entities, *edge weights* model the strength of interaction. If vertex weights are set to the degree of the vertex and edge weights to the product

[1] numerical result of a complex expression as given in (Aiello *et al.*, 2000)

of the weight of its two vertices, a scale-free graph exhibits also a power-law distribution of vertex and edge weights, as observed in real-world data by Li and Chen (2003) and Barrat *et al.* (2004a).

Barrat *et al.* (2004b) describe a generative model quite similar to the BA-model, only that vertex weights are the influencing factor for attachment rather than degree. When attaching a new vertex, the old vertex' weight is increased by the weight of the new vertex plus a parameter δ; for $\delta = 0$, this is equivalent to the BA-model. As $\delta \to \infty$, the power-law exponent decreases to $\gamma = 2$.

2.2.9 Further Reading

A survey of models for the directed web graph that compares several models on various properties can be found in (Bonato, 2005), including several models that not only model the addition of new vertices but also deletion of old ones. The works of Jon Kleinberg (Kleinberg, 1999; Kleinberg *et al.*, 1999, inter al.) convey the consequences for navigation and search algorithms when operating on web-like SWGs. Comprehensive surveys on SWGs in a World Wide Web context are (Deo and Gupta, 2001) and more recently (Chakrabarti and Faloutsos, 2006). On employing Small World network structures in distributed information retrieval, consult (Holz *et al.*, 2007).

Arguments against over-estimating the findings of omnipresent large-scale properties are given by Keller (2005). While this author is right in the respect that the scale-free property found in many complex systems not necessarily implies a common architecture, it is still important to account for these properties when processing data exhibiting them.

This chapter provided definitions and terminology of graph theory. Properties of Small World graphs were discussed and the most prominent graph generation models were reviewed. In the next chapter, SWGs occurring in natural language will be examined and a generation model for language will be developed.

3 Small Worlds of Natural Language

The previous chapter introduced notions of graph theory and reviewed several random graph generation models. In this chapter, power-law distributions and Small World Graphs originating from natural language data are examined in the fashion of Quantitative Linguistics. After giving several data sources that exhibit power-law distributions in rank-frequency in Section 3.1, graphs with Small World properties in language data are discussed in Section 3.2. We shall see that these characteristics are omnipresent in language data, and we should be aware of them when designing Structure Discovery processes. When knowing e.g. that a few hundreds of words make the bulk of words in a text, it is safe to use only these as contextual features without loosing a lot of text coverage. Knowing that word co-occurrence networks possess the scale-free Small World property has implications for clustering these networks.

An interesting aspect is whether these characteristics are only inherent to real natural language data or whether they can be produced with generators of linear sequences in a much simpler way than our intuition about language complexity would suggest – in other words, we shall see how distinctive these characteristics are with respect to tests deciding whether a given sequence is natural language or not.

Finally, a random text generation model that captures many of the characteristics of natural language is defined and quantitatively verified in Section 3.3.

3.1 Power-Laws in Rank-Frequency Distribution

G.K. Zipf (1935, 1949) described the following phenomenon: if all words in a corpus of natural language are arranged in decreasing order of frequency, then the relation between a word's frequency and its rank in the list follows a power-law. Since then, a significant amount of research has been devoted to the question how this property emerges and what kinds of processes generate such Zipfian distributions. Now, some datasets related to language will be pre-

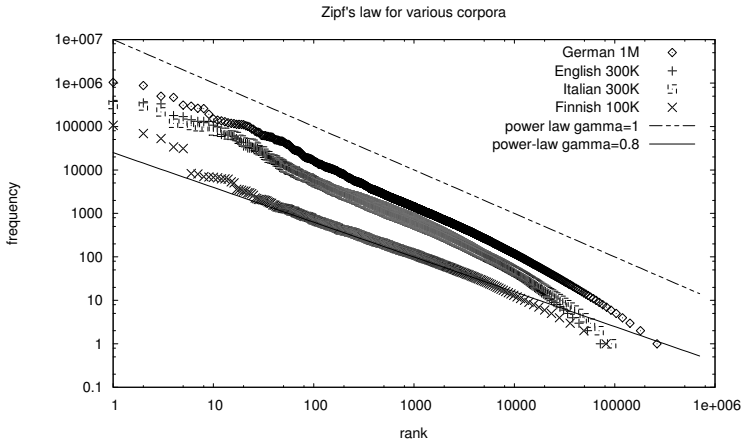


Figure 3.1: Zipf's law for various corpora. The numbers next to the language give the corpus size in sentences. Enlarging the corpus does not effect the slope of the curve, but merely moves it upwards in the plot. Most lines are almost parallel to the ideal power-law curve with $z = 1$. Finnish exhibits a lower slope of $\gamma \approx 0.8$, akin to higher morphological productivity

sented that exhibit a power-law on their rank-frequency distribution. For this, the basic units as given in Section 1.3.1 will be examined.

3.1.1 Word Frequency

The relation between the frequency of a word at rank r and its rank is given by $f(r) \sim r^{-z}$, where z is the exponent of the power-law that corresponds to the slope of the curve in a log-log plot. The exponent z was assumed to be exactly 1 by Zipf; In natural language data, also slightly differing exponents in the range of about 0.7 to 1.2 are observed (see Zanette and Montemurro, 2005). B. Mandelbrot (1953) provided a formula that closer approximates the frequency distributions in language data, noticing that Zipf's law holds only for the medium range of ranks, whereas the curve is flatter for very frequent words and steeper for high ranks. Figure 3.1 displays the word rank-frequency distributions of corpora of different languages taken from the Leipzig Corpora Collection[1].

There exist several exhaustive collections of research capitalising Zipf's law and related distributions[2] ranging over a wide area of

[1] LCC, see `http://www.corpora.uni-leipzig.de` [July 7th, 2007]

[2] e.g. `http://www.nslij-genetics.org/wli/zipf/index.html` [April 1st, 2007] or `http://linkage.rockefeller.edu/wli/zipf/index_ru.html` [April 1st, 2007]

datasets; here, only findings related to natural language will be reported. A related distribution which will play a role at a later stage is the *lexical spectrum* (see Ferrer-i-Cancho and Solé, 2002), which gives the probability of choosing a word from the vocabulary with a given frequency. For natural language, the lexical spectrum follows a power-law with slope $\gamma = \frac{1}{z} + 1$, where z is the exponent of the Zipfian rank-frequency distribution. For the relation between lexical spectrum, Zipf's law and Pareto's law, see (Adamic, 2000).

But Zipf's law in its original form is just the tip of the iceberg of power-law distributions in a quantitative description of language. While a Zipfian distribution for word frequencies can be obtained by a simple model of generating letter sequences with space characters as word boundaries (Mandelbrot, 1953; Miller, 1957), these models based on "intermittent silence" can neither reproduce the distributions on sentence length (see Sigurd *et al.*, 2004), nor explain the relations of words in sequence. But before elaborating further on this point in Section 3.3, more power-law distributions in natural language are discussed and exemplified.

3.1.2 Letter N-grams

To continue with a counterexample, letter frequencies do not obey a power-law in the rank-frequency distribution. This also holds for letter N-grams (including the space character), yet for higher N, the rank-frequency plots show a large power-law regime with exponential tails for high ranks. Figure 3.2 shows the rank-frequency plots for letter N-grams up to $N = 6$ for the first 10,000 sentences of the British National Corpus (BNC[3], Burnard (1995)).

Still, letter frequency distributions can be used to show that letters are not forming letter bigrams from single letters independently, but there are restrictions on their combination. While this intuitively seems obvious for letter combination, the following test is proposed for quantitatively examining the effects of these restrictions: from letter unigram probabilities, a text is generated that follows the letter unigram distribution by randomly and independently drawing letters according to their distribution and concatenating them. The letter bigram frequency distribution of this generated text can be compared to the letter bigram frequency distribution of the real text the unigram distribution was measured from. Figure 3.3 shows the generated and the real rank-frequency plot, again from the small BNC sample.

[3] `http://www.natcorp.ox.ac.uk/` [April 1st, 2007]

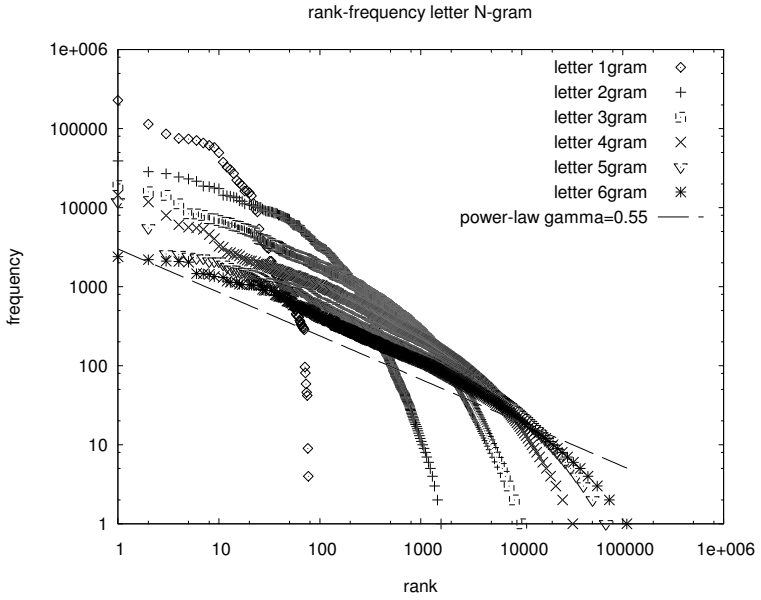


Figure 3.2: Rank-Frequency distributions for letter N-grams for the first 10,000 sentences in the BNC. Letter N-gram rank-frequency distributions do not exhibit power-laws on the full scale, but increasing N results in a larger power-law regime for low ranks

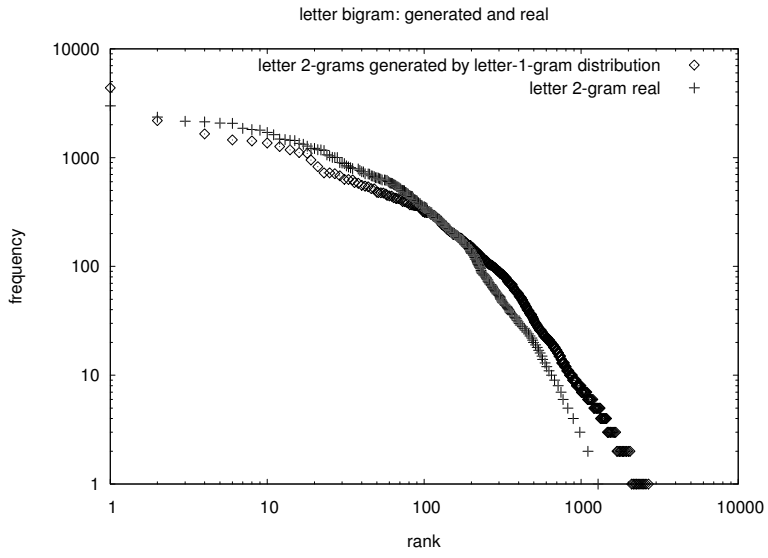


Figure 3.3: Rank-frequency plots for letter bigrams, for a text generated from letter unigram probabilities and for the BNC sample

The two curves clearly differ. The generated bigrams without restrictions predict a higher number of different bigrams and lower frequencies for bigrams of high ranks as compared to the real text bigram statistics. This shows that letter combination restrictions do exist, as not nearly all bigrams predicted by the generation process were observed, resulting in higher counts for valid bigrams in the sample.

3.1.3 Word N-grams

For word N-grams, the relation between rank and frequency follows a power-law, just as in the case for words (unigrams). Figure 3.4 (left) shows the rank-frequency plots up to $N = 4$, based on the first 1 million sentences of the BNC. As more different word combinations are possible with increasing N, the curves get flatter as the same total frequency is shared amongst more units, as previously observed by e.g. (Smith and Devine, 1985) and (Ha *et al.*, 2002). Testing concatenation restrictions quantitatively as above for letters, it might at the first glance seem surprising that the curve for a text generated with word unigram frequencies differs only very little from the word bigram curve, as Figure 3.4 (right) shows. Small differences are only observable for low ranks: more top-rank generated bigrams reflect that words are usually not repeated in the text. More low-ranked and less high-ranked real bigrams indicate that word concatenation takes place not entirely without restrictions, yet is subject to much more variety than letter concatenation. This coincides with the intuition that it is, for a given word pair, almost always possible to form a correct English sentence in which these words are neighbours. Regarding quantitative aspects, the frequency distribution of word bigrams can be produced by a generation process based on word unigram probabilities. In Section 3.3, a measure will be introduced that can better distinguish between a text generated in this way and a real text.

3.1.4 Sentence Frequency

In larger corpora that are compiled from a variety of sources, the number of duplicate sentences is not to be neglected. In the full BNC, which serves as data basis in this case, 7.3% of the sentences occur two or more times. The most frequent sentences are "Yeah.", "Mm.", "Yes." and "No.", which are mostly found in the section of spoken language. But also longer expressions like "Our next bulletin is at 10.30 p.m." have a count of over 250. The sentence frequencies

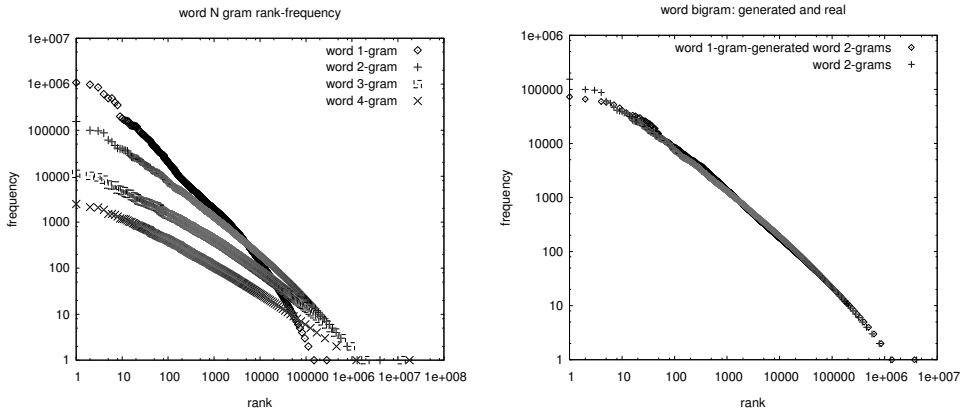


Figure 3.4: Left: rank-frequency distributions for word N-grams for the first 1 Million sentences in the BNC. Word N-gram rank-frequency distributions exhibit power-laws. (Right: rank-frequency plots for word bigrams, for a text generated from letter unigram probabilities and for the BNC sample

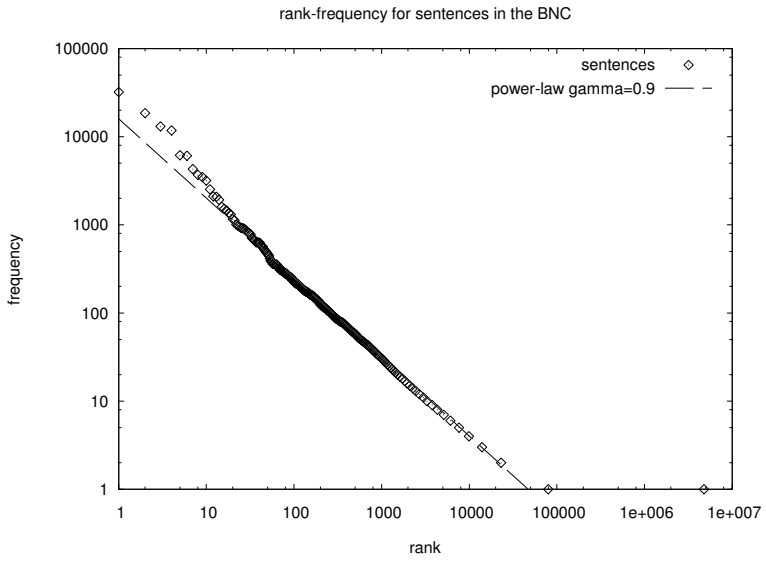


Figure 3.5: Rank-frequency plot for sentence frequencies in the full BNC, following a power-law with $\gamma \approx 0.9$, but with a high fraction of sentences occurring only once

also follow a power-law with an exponent close to 1 (cf. Figure 3.5), indicating that Zipf's law also holds for sentence frequencies.

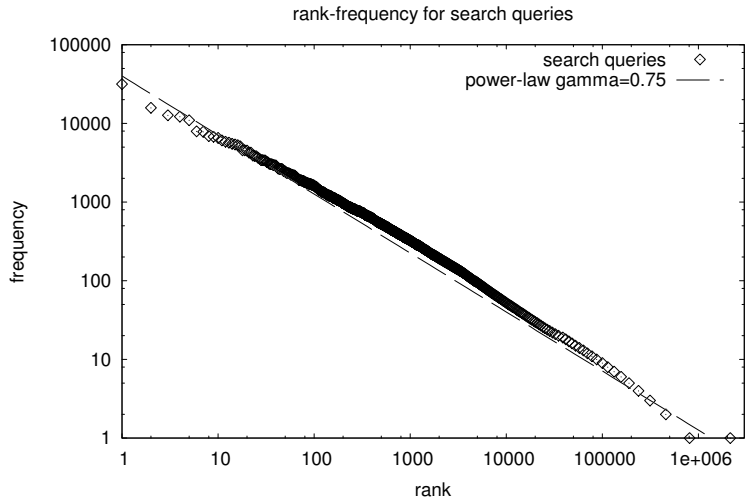


Figure 3.6: Rank-frequency plot for AltaVista search queries, following a power-law with $\gamma \approx 0.75$

3.1.5 Other Power-Laws in Language Data

The results above strongly suggest that when counting document frequencies in large collections such as the World Wide Web, another power-law distribution would be found, but an analysis has not been carried out and would require access to the index of a web search engine.

Further, there are more power-laws in language-related areas, some of which are provided briefly to illustrate their omnipresence:

- Web page requests follow a power-law, which was employed for a caching mechanism in (Glassman, 1994).
- Related to this, frequencies of web search queries during a fixed time span also follow a power-law, as exemplified in Figure 3.6 for a 7 million queries log of AltaVista[4] as used by Lempel and Moran (2003).
- The number of authors of Wikipedia[5] articles was found to follow a power-law with $\gamma \approx 2.7$ for a large regime in (Voss, 2005), who also discusses other power-laws regarding the number of links.

[4] `http://www.altavista.com`
[5] `http://www.wikipedia.org`

3.2 Scale-Free Small Worlds in Language Data

Whereas the previous section discussed the shape of rank-frequency distributions for natural language units, now the properties of graphs with units represented as vertices and relations between them as edges will be in the focus of interest. As already stated in Section 1.3.2, internal as well as contextual features can be employed for computing similarities between language units that are represented as (possibly weighted) edges in the graph. Some of the graphs discussed here can be classified in being scale-free Small World graphs; others differ from these characteristics and represent other, but related graph classes.

3.2.1 Word Co-occurrence Graph

The notion of *word co-occurrence* is used to model dependencies between words. If two words X and Y occur together in some contextual unit of information (as neighbours, in a word window of 5, in a clause, in a sentence, in a paragraph), they are said to co-occur. When regarding words as vertices and edge weights as the number of times two words co-occur, the *word co-occurrence graph* of a corpus is given by the entirety of all word co-occurrences.

In the following, specifically two types of co-occurrence graphs are considered: the graph as induced by neighbouring words, henceforth called neighbour-based graph, and the graph as induced by sentence-based co-occurrence, henceforth called sentence-based graph. The neighbour-based graph can be undirected or directed with edges going from the left to the right words as found in the corpus, the sentence-based graph is undirected. To illustrate co-occurrence graphs, Figure 3.7 displays the sentence-based co-occurrence graph and the co-occurrence graph based on neighbouring words for a song by Radiohead.

To find out whether the co-occurrence of two specific words A and B is merely due to chance or exhibits a statistical dependency, measures are used that compute, to what extent the co-occurrence of A and B is statistically significant. Many significance measures can be found in the literature, for extensive overviews consult e.g. (Evert, 2004) or (Bordag, 2007). In general, the measures compare the probability of A and B to co-occur under the assumption of their statistical independence with the number of times A and B actually co-occurred in the corpus. In this work, the log likelihood ratio (Dunning, 1993) is used to sort the chaff from the wheat. It is given in expanded form by Bordag (2007):

Exit Music (for a film)

Wake from your sleep
The drying of your tears
Today we escape
We escape.

Pack and get dressed
Before your father hears us
Before all hell breaks loose.

Breathe keep breathing
Don't lose your nerve.
Breathe keep breathing
I can't do this alone.

Sing us a song
A song to keep us warm
There is such a chill
Such a chill.

You can laugh
A spineless laugh
We hope that your rules
and wisdom choke you
Now we are one
In everlasting peace

We hope that you choke,
that you choke
We hope that you choke,
that you choke
We hope that you choke,
that you choke

Figure 3.7: The co-occurrence graphs for the song "Exit Music (for a film)" by Radiohead. Upper graph: words co-occurring in the same verse. Lower graph: words co-occurring as neighbours. Edge weights are omitted. Notice that the neighbouring relation does not cross verse boundaries

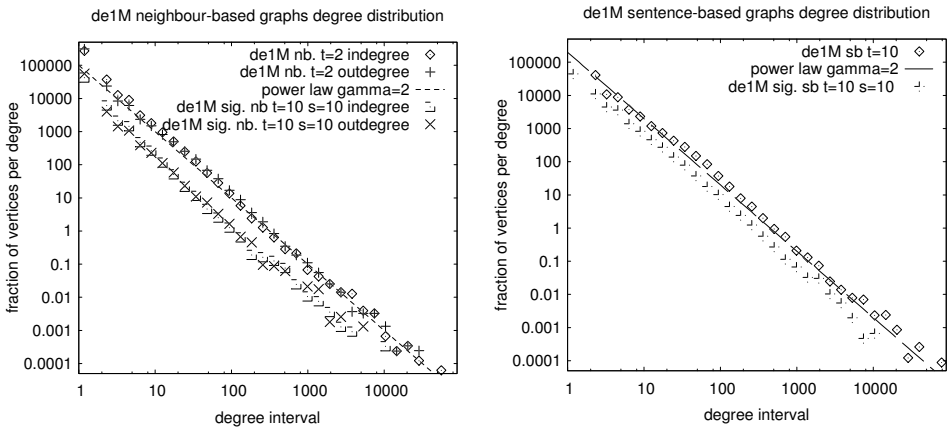
$$-2\log\lambda = 2\left[\begin{array}{l} n\log n - n_A\log n_A - n_B\log n_B + n_{AB}\log n_{AB} \\ +(n-n_A-n_B+n_{AB})\log(n-n_A-n_B+n_{AB}) \\ +(n_A-n_{AB})\log(n_A-n_{AB}) + (n_B-n_{AB})\log(n_B-n_{AB}) \\ -(n-n_A)\log(n-n_A) - (n-n_B)\log(n-n_B) \end{array}\right]$$

where n is the total number of contexts, n_A the frequency of A, n_B the frequency of B and n_{AB} the number of co-occurrences of A and B. As pointed out by Moore (2004), this formula overestimates the co-occurrence significance for small n_{AB}. For this reason, often a frequency threshold t on n_{AB} (e.g. a minimum of $n_{AB} = 2$) is applied. Further, a significance threshold s regulates the density of the graph; for the log likelihood ratio, the significance values correspond to the χ^2 tail probabilities (Moore, 2004), which makes it possible to translate the significance value into an error rate for rejecting the independence assumption. For example, a log likelihood ratio of 3.84 corresponds to a 5% error in stating that two words do not occur by chance, a significance of 6.63 corresponds to 1% error.

The operation of applying a significance test results in pruning edges being in existence due to random noise and keeping almost exclusively edges that reflect a true association between their endpoints. Graphs that contain all significant co-occurrences of a corpus, with edge weights set to the significance value between its endpoints, are called *significant co-occurrence graphs* in the remainder. For convenience, no singletons in the graph are allowed, i.e. if a vertex is not contained in any edge because none of the co-occurrences for the corresponding word is significant, then the vertex is excluded from the graph.

As observed by Ferrer-i-Cancho and Solé (2001) and Quasthoff *et al.* (2006), word co-occurrence graphs exhibit the scale-free Small World property. This goes in line with co-occurrence graphs reflecting human associations (see Rapp, 1996) and human associations in turn forming Small World graphs (see Steyvers and Tenenbaum, 2005). The claim is confirmed here on an exemplary basis with the graph for LCC's 1 million sentence corpus for German. Figure 3.8 gives the degree distributions and graph characteristics for various co-occurrence graphs.

The shape of the distribution is dependent on the language, as Figure 3.9 shows. Some languages – here English and Italian – have a hump-shaped distribution in the log-log plot where the first regime follows a power-law with a lower exponent than the second regime, as observed by Ferrer-i-Cancho and Solé (2001). For the Finnish and German corpora examined here, this effect could not be found in the



Characteristic	nb t=2	sig. nb t=10 s=10	sb t=10	sig. sb t=10, s=10
n	217,491	37,728	82,008	42,507
L	3.1005	3.3947	1.9997	2.8437
D	8	10	2	7
C	0.56087	0.20372	0.99188	0.71327
T	0.00563	0.00429	0.01274	0.01788
$< k >$	9.8034	6.4109	29.4379	12.8376

Figure 3.8: Graph characteristics for various co-occurrence graphs of LCC's 1 Million sentence German corpus. Abbreviations: nb = neighbour-based, sb = sentence-based, sig. = significant, t = co-occurrence frequency threshold, s = co-occurrence significance threshold. While the exact shapes of the distributions are language and corpus dependent, the overall characteristics are valid for all samples of natural language of sufficient size. The slope of the distribution is invariant to changes of thresholds. Characteristic path length and a high clustering coefficient at low average degrees are characteristic for SWGs

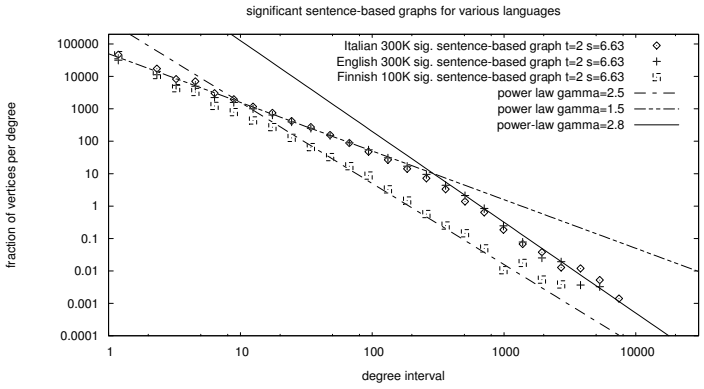


Figure 3.9: Degree distribution of significant sentence-based co-occurrence graphs of similar thresholds for Italian, English and Finnish

data. This property of two power-law regimes in the degree distribution of word co-occurrence graphs motivated the DM-model (see Section 2.2.7, (Dorogovtsev and Mendes, 2001a)). There, the cross-over-point of the two power-law regimes is motivated by a so-called *kernel lexicon* of about 5,000 words that can be combined with all words of a language.

The original experiments of Ferrer-i-Cancho and Solé (2001) operated on a word co-occurrence graph with window size 2: an edge is drawn between words if they appear together at least once in a distance of one or two words in the corpus. Reproducing their experiment with the first 70 Million words of the BNC and corpora of German, Icelandic and Italian of similar size reveals that the degree distribution of the English and the Italian graph is in fact approximated by two power-law regimes. In contrast to this, German and Icelandic show a single power-law distribution, just as in the experiments above, see Figure 3.10. These results suggest that two power-law regimes in word co-occurrence graphs with window size 2 are not a language universal, but only hold for some languages.

To examine the hump-shaped distributions further, Figure 3.11 displays the degree distribution for the neighbour-based word co-occurrence graphs and the word-co-occurrence graphs for connecting only words that appear in a distance of 2. As it becomes clear from the plots, the hump-shaped distribution is mainly caused by words co-occurring in distance 2, whereas the neighbour-based graph shows only a slight deviation from a single power-law. Together with the observations from sentence-based co-occurrence graphs of different languages in Figure 3.9, it becomes clear that a

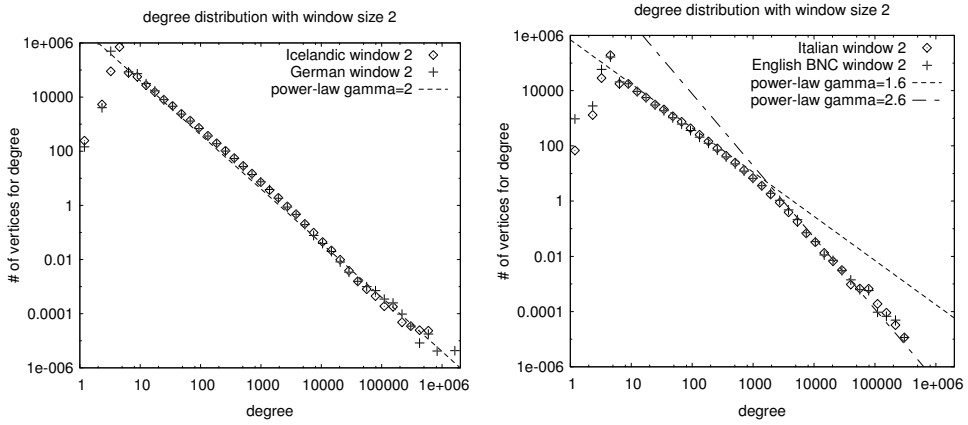


Figure 3.10: Degree distributions in word co-occurrence graphs for window size 2. Left: the distribution for German and Icelandic is approximated by a power-law with $\gamma = 2$. Right: for English (BNC) and Italian, the distribution is approximated by two power-law regimes

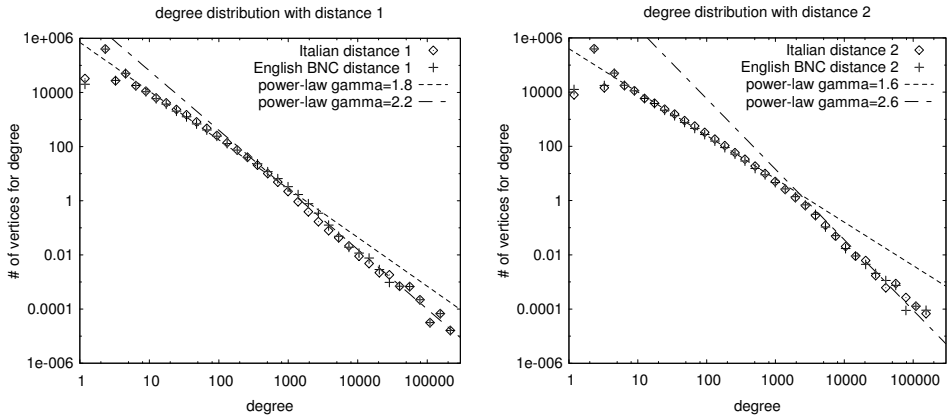


Figure 3.11: Degree distributions in word co-occurrence graphs for distance 1 and distance 2 for English (BNC) and Italian. The hump-shaped distribution is much more distinctive for distance 2

hump-shaped distribution with two power-law regimes is caused by long-distance relationships between words, if present at all.

Applications of Word Co-occurrences

Word co-occurrence statistics are an established standard and have been used in many language processing systems. My work includes including bilingual dictionary acquisition as in (Biemann and

Quasthoff, 2005) and (Cysouw *et al.*, 2007), semantic lexicon extension (see Biemann *et al.*, 2004c) and visualisation of concept trails in (Biemann *et al.*, 2004b). In Chapter 5, they will be used as building blocks in various Structure Discovery processes.

3.2.2 Co-occurrence Graphs of Higher Order

The significant word co-occurrence graph of a corpus represents words that are likely to appear near to each other. When interested in words co-occurring with similar other words, it is possible to transform the above defined (first order) co-occurrence graph into a second order co-occurrence graph by drawing an edge between two words A and B if they share a common neighbour in the first order graph. Whereas the first order word co-occurrence graph represents the global context per word, the corresponding second order graph contains relations between words which have similar global contexts. The edge can be weighted according to the number of common neighbours, e.g. by $weight = |neigh(A) \cap neigh(B)|$. Figure 3.12 shows neighbourhoods of the significant sentence-based first-order word co-occurrence graph from LCC's English web corpus[6] for the words *jazz* and *rock*. Taking into account only the data depicted, *jazz* and *rock* are connected with an edge of weight 3 in the second-order graph, corresponding to their common neighbours *album*, *music* and *band*. The fact that they share an edge in the first order graph is ignored.

In general, a graph of order $N + 1$ can be obtained from the graph of order N, using the same transformation. The higher order transformation without thresholding is equivalent to a multiplication of the unweighted adjacency matrix A with itself, then zeroing the main diagonal by subtracting the degree matrix of A. Since the average path length of scale-free SW graphs is short and local clustering is high, this operation leads to an almost fully connected graph in the limit, which does not allow to draw conclusions about the initial structure. Thus, the graph is pruned in every iteration N in the following way: For each vertex, only the max_N outgoing edges with the highest weights are taken into account. Notice that this vertex degree threshold max_N does not limit the maximum degree, as thresholding is asymmetric. This operation is equivalent with only keeping the max_N largest entries per row in the adjacency matrix $A = (a_{ij})$, then $A_t = (sign(a_{ij} + a_{ji}))$, resulting in an undirected graph.

[6] `http://corpora.informatik.uni-leipzig.de/?dict=en` [April 1st, 2007]

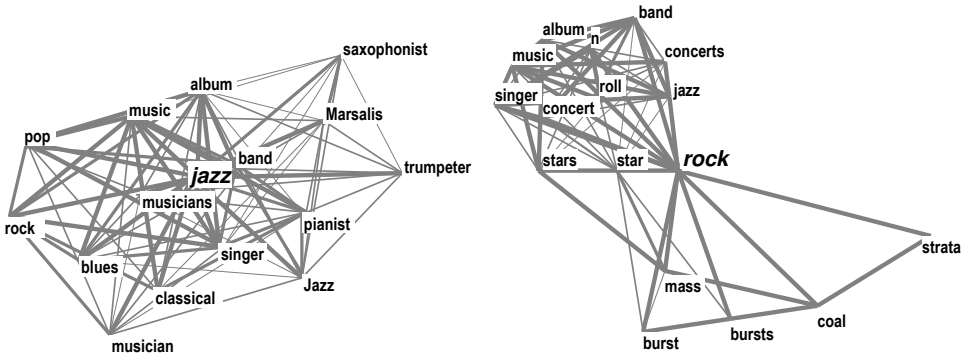


Figure 3.12: Neighbourhoods of *jazz* and *rock* in the significant sentence-based word co-occurrence graph as displayed on LCC's English Corpus website. Both neighbourhoods contain *album*, *music* and *band*, which leads to an edge weight of 3 in the second-order graph

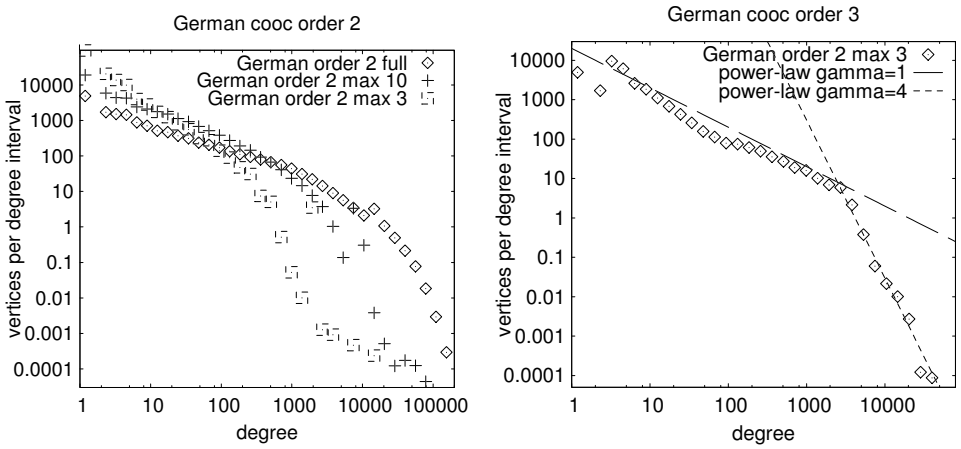


Figure 3.13: Degree distributions of word-co-occurrence graphs of higher order. The first order graph is the sentence-based word co-occurrence graph of LCC's 1 million German sentence corpus ($s = 6.63$, $t = 2$). Left: $N = 2$ for $max_2 = 3$, $max_2 = 10$ and $max_2 = \infty$. Right: $N = 3$ for $t_2 = 3$, $t_3 = \infty$

To examine quantitative effects of the higher order transformation, the sentence-based word co-occurrence graph of LCC's 1 million German sentence corpus ($s = 6.63$, $t = 2$) underwent this operation. Figure 3.13 depicts the degree distributions for $N = 2$ and $N = 3$ for different max_N.

Applying the max_N threshold causes the degree distribution to change, especially for high degrees. In the third order graph, two

power-law regimes are observable.

Studying the degree distribution of higher order word co-occurrence graphs revealed that the characteristic of being governed by power-laws is invariant to the higher order transformation, yet the power-law exponent changes. This indicates that the power-law characteristic is inherent at many levels in natural language data.

To examine what this transformation yields on the graphs generated by the random graph models in the previous chapter, Figure 3.14 shows the degree distribution of second order and third order graphs as generated by the models described in Section 2.2. The underlying first order graphs are the undirected graphs of order 10,000 and size 50,000 ($< k >= 10$) from the BA-model, the ST-model, and the DM-model.

While the thorough interpretation of second order graphs of random graphs might be subject to further studies, the following should be noted: the higher order transformation reduces the power-law exponent of the BA-model graph from $\gamma = 3$ to $\gamma = 2$ in the second order and to $\gamma \approx 0.7$ in the third order. For the ST-model, the degree distribution of the full second order graph shows a maximum around $2M$, then decays with a power-law with exponent $\gamma \approx 2.7$. In the third order ST-graph, the maximum moves to around $4M^2$ for sufficient max_2. The DM-model second order graph shows, like the first order DM-model graph, two power-law regimes in the full version, and a power-law with $\gamma \approx 2$ for the pruned versions. The third order degree distribution exhibits many more vertices with high degrees than predicted by a power-law.

In summary, all random graph models exhibit clear differences to word co-occurrence networks with respect to the higher order transformation. The ST-model shows maxima depending on the average degree of the first oder graph. The BA-model's power law is decreased with higher orders, but is able to explain a degree distribution with power-law exponent 2. The full DM model exhibits the same two power-law regimes in the second order as observed for German sentence-based word co-occurrences in the third order.

Applications of Co-occurrence Graphs of Higher Orders

In (Biemann *et al.*, 2004a) and (Mahn and Biemann, 2005), the utility of word co-occurrence graphs of higher orders are examined for lexical semantic acquisition. The highest potential for extracting paradigmatic semantic relations can be attributed to second- and third-order word co-occurrences. Second-order graphs are evaluated

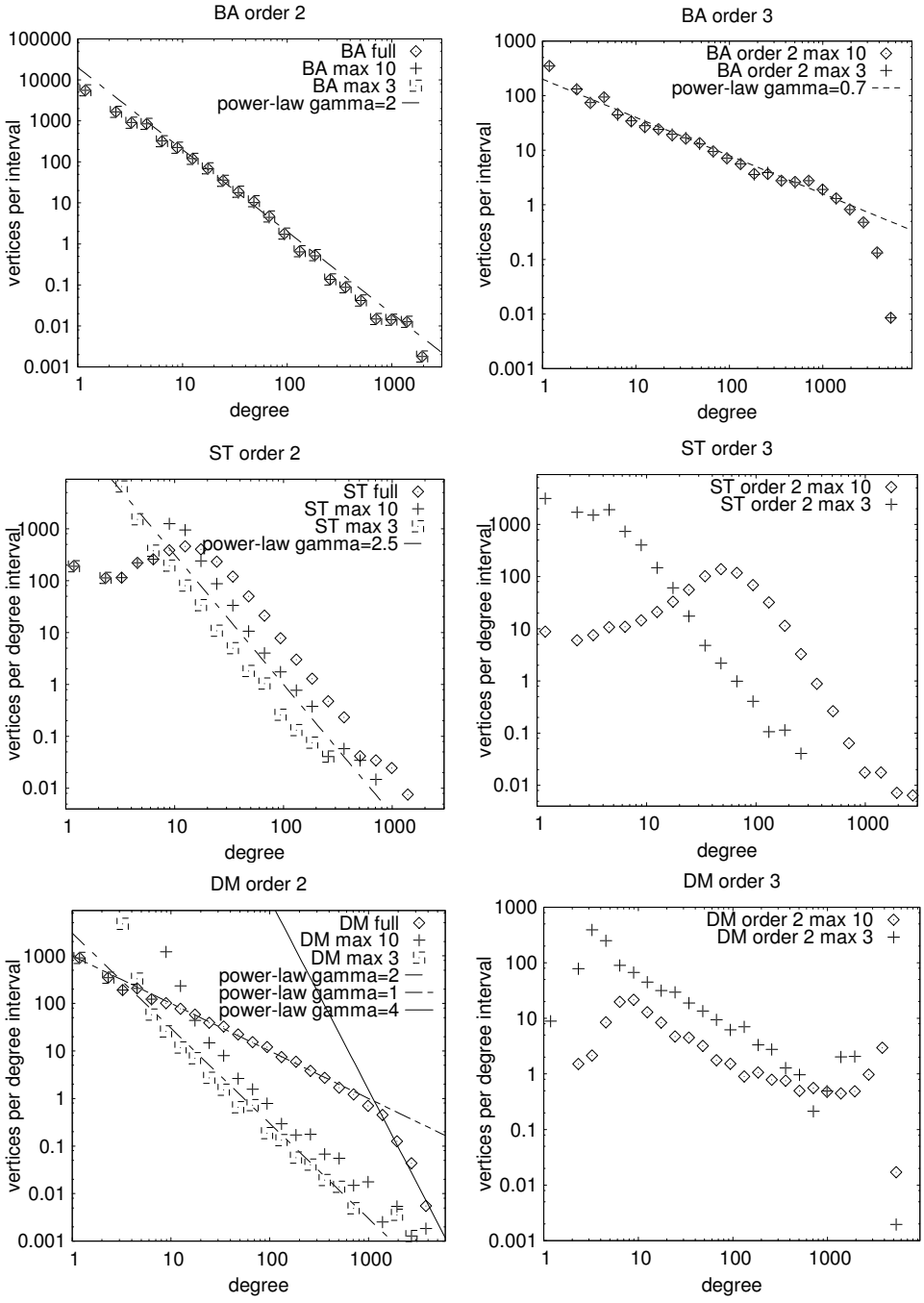


Figure 3.14: Second and third order graph degree distributions for BA-model, ST-model and DM-model graphs

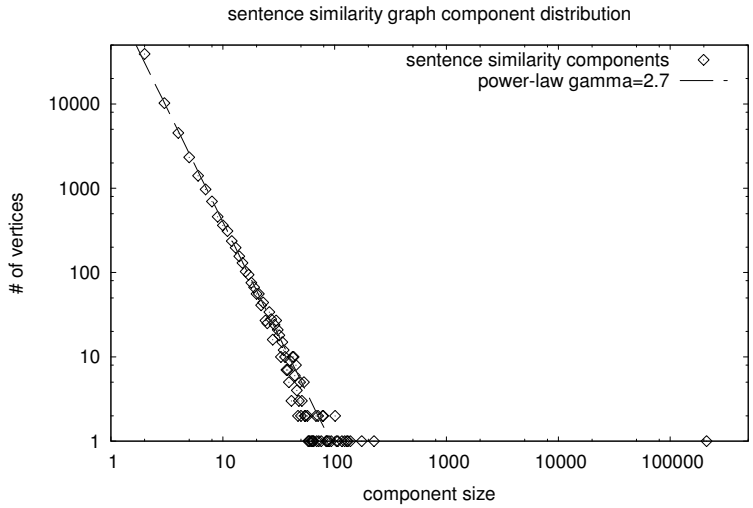
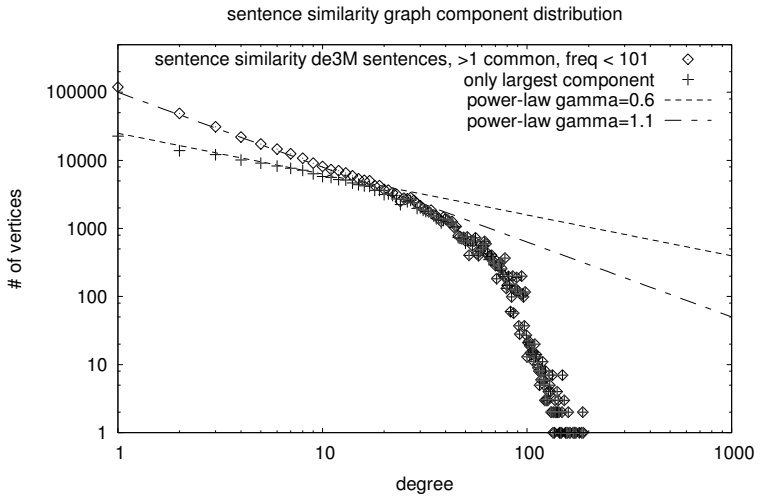


Figure 3.15: Component size distribution for the sentence similarity graph of LCC's 3 Million sentence German corpus. The component size distribution follows a power-law with $\gamma \approx 2.7$ for small components, the largest component comprises 211,447 out of 416,922 total vertices. The component size distribution complies to the theoretical results of Aiello *et al.* (2000), see Section 2.2.8

by Bordag (2007) against lexical semantic resources. In Section 5.2, a second-order graph based on neighbouring words on both sides will be, amongst other mechanisms, used for inducing syntactic word classes.

3.2.3 Sentence Similarity

Using words as internal features, the similarity of two sentences can be measured by how many common words they share. Since the few top frequency words are contained in most sentences as a consequence of Zipf's law, their influence should be downweighted or they should be excluded to arrive at a useful measure for sentence similarity. Here, the sentence similarity graph of sentences sharing at least two common words is examined, with the maximum frequency of these words bounded by 100. The corpus of examination is here LCC's 3 Million sentences of German. Figure 3.15 shows the component size distribution for this sentence-similarity graph, Figure 3.16 shows the degree distributions for the entire graph and for its largest component.



Characteristic	Sentence Similarity Graph	Largest Component Only
n	416,922	211,447
L	11.4850	11.4850
D	34	34
C	0.8836	0.8812
T	0.8807	0.8769
$< k >$	10.5968	17.3982

Figure 3.16: Degree distribution for the sentence similarity graph of LCC's 3 Million sentence German corpus and its largest component. An edge between two vertices representing sentences is drawn if the sentences share at least two words with corpus frequency <101, singletons are excluded

The degree distribution of the entire graph follows a power-law with γ close to 1 for low degrees and decays faster for high degrees, the largest component's degree distribution plot is flatter for low degrees. This resembles the broad-scale degree distribution as introduced in Section 2.2.4 and can be attributed to limited sentence length: as sentences are not arbitrarily long (see more on sentence length distribution in Section 3.3.6), they cannot be similar to an arbitrary high number of other sentences with respect to the measure discussed here, as the number of sentences per feature word is bound by the word frequency limit. Cost in terms of Section 2.2.4 corresponds to sentence length restrictions here. However, the extremely high values for transitivity and clustering coefficient and the low γ values for the degree distribution for low degree vertices and comparably long average shortest path lengths indicate that the sentence similarity graph belongs to a different graph class than the ones presented in Section 2.2.

Applications of the Sentence Similarity Graph

A similar measure is used in (Biemann and Quasthoff, 2007) for document similarity, obtaining well-correlated results when evaluating against a given document classification. A precision-recall tradeoff arises when lowering the frequency threshold for feature words or increasing the minimum number of common feature words two documents must have in order to be connected in the graph: both improves precision but results in many singleton vertices which lowers the total number of documents that are considered.

3.2.4 Summary on Scale-Free Small Worlds in Language Data

The examples above confirm the claim that graphs built on various aspects of natural language data often exhibit the scale-free Small World property or similar characteristics. Experiments with generated text corpora suggest that this is mainly due to the power-law frequency distribution of language units, cf. also Section 3.3. The slopes of the power-law approximating the degree distributions can often not be produced using the random graph generation models of Section 2.2: especially, all previously discussed generation models fail on explaining the properties of word co-occurrence graphs, where $\gamma \approx 2$ was observed as power-law exponent of the degree distribution. Of the generation models inspired by language data, the ST-model exhibits $\gamma = 3$, whereas the universality of the DM-model

to capture word co-occurrence graph characteristics can be doubted after examining data from different languages.

Therefore, a new generation model is described in the following section, which succeeds in modelling the co-occurrence graph and other properties of natural language. For this, the graph is not generated directly, but obtained from a linear sequence of generated text.

3.3 A Random Generation Model for Language

Now, a graph-based generation model for language will be given that reproduces the large-scale behaviour of language, i.e. a Zipfian distribution of word frequency and a power-law distribution with $\gamma = 2$ in the co-occurrence graph. Further, the model will produce sentence-like concatenations of words. At this, these characteristics emerge in a generation process, rather than being directly constrained by them. The model is inspired by SWG generation models as discussed in Section 2.2 and reproduces the language-characteristic graphs discussed above. This is the main contribution of the generation model and its distinguishing feature from previous random text models. The random text model first appeared in (Biemann, 2007).

The study of random text models unveils, which of the characteristics of natural language can be produced by well-understood mechanisms. When keeping these mechanisms as simple as possible, they can be perceived as plausible explanations for the origin of language structure. Common characteristics of random text and natural language indicate the amount of structure that is explained by the generation process, differences indicate missing aspects in the random generator. These discrepancies can inspire both more complex random text models and Structure Discovery processes that employ the yet unmodelled regularities that cause them.

3.3.1 Review of Random Text Models

B. Mandelbrot (1953) provided a word generation model that produces random words of arbitrary average length in the following way: with a probability w, a word separator is generated at each step, with probability $(1 - w)/N$, a letter from an alphabet of size N is generated, each letter having the same probability. This is sometimes called the "monkey at the typewriter" (Miller, 1957). The frequency distribution follows a power-law for long streams of words, yet the equiprobability of letters causes the plot to show a step-wise rather

than a smooth behaviour, as examined in (Ferrer-i-Cancho and Solé, 2002) and shown in Figure 3.18. In the same study, a smooth rank distribution could be obtained by setting the letter probabilities according to letter frequencies in a natural language text. But the question of how these letter probabilities emerge remained unanswered[7]. Li (1992) formally proves that the power-law frequency distribution in this random text model is a consequence of the transformation from the word's length to its rank, stretching the exponential distribution to a power-law.

Another random text model is given in (Simon, 1955), which does not take an alphabet of single letters into consideration. Instead, at each time step, a previously unseen new word is added to the stream with probability α, whereas with probability $(1 - \alpha)$, the next word is chosen amongst the words at previous positions. As words with higher frequency in the already generated stream have a higher probability of being added again, this imposes a strong competition among different words, resulting in a frequency distribution that follows a power-law with exponent $\gamma = (1 - \alpha)$. This was taken up by Zanette and Montemurro (2005), who slightly modify Simon's model. They introduce sublinear vocabulary growth by additionally making the new word probability dependent onto the time step. Furthermore, they apply a threshold on the maximal probability a previously seen word is generated with, being able to modify the exponent z as well as to model the flatter curve for high frequency words.

Neither the Mandelbrot nor the Simon generation model takes the sequence of words into account. Simon treats the previously generated stream as a bag of words, and Mandelbrot does not consider the previous stream at all. This is certainly an over-simplification, as natural language exhibits structural properties within sentences and texts that are not grasped by a bag of words approach.

The work by Kanter and Kessler (1995) is, to my knowledge, the only piece of research with word order awareness for random text generation. They show that a 2-parameter Markov process gives rise to a stationary distribution that exhibits the word or letter frequency distribution characteristics of natural language. However, the Markov process is initialised such that any state has exactly two successor states, which means that after each word only two different succeeding words are possible. This is certainly not reflecting natural language properties, where in fact successor frequencies of words

[7] A suitable random initialisation of letter frequencies can produce a smooth rank distribution, yet is still not emergent

follow a power-law and more successors can be observed for more frequent words. But even when allowing a more realistic number of successor states, the transition probabilities of a Markov model need to be initialised a priori in a sensible way. Further, the fixed number of states does not account for arbitrary large, extensible vocabularies.

While these previous random text models mainly aim at producing a Zipfian distribution of word frequencies, the new model presented here also takes the properties of neighbouring co-occurrence into account and introduces the notion of sentences in random text. After having pointed out the deficiencies of related models, a generation process is provided that takes neither the Zipf distribution on word frequencies nor the Small World property of the neighbouring co-occurrence graph as a constraint. Nevertheless, these distributions emerge in the process. The distributions obtained with the random generation model are compared to a sample of natural language data, showing high agreement also on word length and sentence length.

3.3.2 Desiderata for Random Text Models

The construction of the novel random text generation model proceeds according to the following guidelines (see Kumar *et al.*, 1999):

- *simplicity*: a generation model should reach its goal using the simplest mechanisms possible but results should still comply to characteristics of real language
- *plausibility*: without claiming that the model is an exhaustive description of what makes human brains generate and evolve language, there should be at least a possibility that similar mechanisms can operate in human brains.
- *emergence*: rather than constraining the model with the characteristics we would like to see in the generated stream, these features should emerge in the process.

The model described here is basically composed of two parts that will be described separately: a *word generator* that produces random words composed of letters, and a *sentence generator* that composes random sentences of words. Both parts use an internal graph structure, where traces of previously generated words and sentences are memorised. A key notion is the strategy of following 'beaten tracks': letters, words and sequences of words that have been generated before are more likely to be generated again in the future. But before

laying out both generators in detail, ways of testing agreement of a random text model with natural language text are introduced.

3.3.3 Testing Properties of Word Streams

All previous approaches aimed at reproducing a Zipfian distribution on word frequency, which is a criterion that certainly has to be fulfilled. But there are more characteristics that should be obeyed to make a random text more language-like than previous models:

- *Lexical spectrum*: the smoothness or step-wise shape of the rank-frequency distribution affects the lexical spectrum, which is the probability distribution on word frequency. In natural language texts, this distribution follows a power-law with an exponent close to 2 (cf. Ferrer-i-Cancho and Solé, 2002).
- *Distribution of word length*: according to Sigurd *et al.* (2004), the distribution of word frequencies by length follows a variant of the gamma distribution, which is given by $F(x) = x^a \cdot b^x$.
- *Distribution of sentence length*: the random text's sentence length distribution should resemble natural language. In (Sigurd *et al.*, 2004), the same variant of the gamma distribution as for word length is fitted to sentence length.
- *Significant neighbour-based co-occurrence*: at random generation without word order awareness, the number of word pairs that are significantly co-occurring in neighbouring positions should be very low. In text generation, the number of significant co-occurrences in natural language as well as the graph structure of the neighbour-based co-occurrence graph should be reproduced.

The latter characteristic refers to the distribution of words *in sequence*. As written language is rather an artefact of the most recent millennia than a realistic sample of everyday language, the beginning of the spoken language section of the BNC is used to test the model against. For simplicity, all letters are capitalised and special characters are removed, such that merely the 26 letters of the English alphabet are contained in the sample. Being aware of the fact that letter transcription is in itself an artefact of written language, this is chosen as a good-enough approximation. The sample contains 1 million words in 125,395 sentences with an average length of 7.975 words, which are composed of 3.502 letters in average.

3.3.4 Word Generator

The word generator emits sequences of letters, which are generated randomly in the following way: the word generator starts with a directed graph G_l with all N letters it is allowed to choose from as vertices. Initially, all vertices are connected to themselves with weight 1. When generating a word, the generator chooses a letter x according to the letter's probability $P(letter = x)$, which is computed as the normalised weight sum of outgoing edges:

$$P(letter = x) = \frac{weightsum(x)}{\sum_{i \in G_l} weightsum(i)}$$

where $weightsum(x) = \sum_{v \in V(G_l)} ew(x, v)$.

Then the word generator proceeds with the next position. At every position, the word ends with a probability $w \in (0, 1)$. Otherwise, the next letter is generated according to the letter production probability given above. For every letter bigram, the weight of the directed edge between the preceding and current letter in the letter graph is increased by one. This results in self-reinforcement of letter probabilities: the more often a letter is generated, the higher its weight sum will be in subsequent steps, leading to an increased generation probability. Figure 3.17 shows how a word generator with three letters A, B, C changes its weights during the generation of the words AA, BCB and ABC. The word generating function is given in Algorithms 1 and 2: here, `concatenate(.,.)` returns a string concatenation of its two string arguments, `random-number` returns a random number between 0 and 1. In the implementation, the Mersenne Twister (Matsumoto and Nishimura, 1998) is used for random number generation.

Algorithm 1 initialise-letter-graph G_l

insert $X_1, X_2, ..X_N$ vertices in $V(G_l)$
insert edges $e(X_i, X_j)$ for $i, j \in \{1, 2, ..N\}$ into $E(G_l)$ with weight 0
insert edges $e(X_i, X_i)$ for $i \in \{1, 2, ..N\}$ into $E(G_l)$ with weight 1

The word generator alone does produce a smooth Zipfian distribution on word frequencies and a lexical spectrum following a power-law. Figure 3.18 shows the frequency distribution and the lexical spectrum of 1 million words as generated by the word generator with $w = 0.3$ on 26 letters in comparison to a Mandelbrot generator with the same parameters. As depicted in Figure 3.18, the word generator fulfils the requirements on Zipf's law and the lexical spectrum, yielding a Zipfian exponent of around 1 and a power-law exponent of around 2 in the lexical spectrum, both matching the values

Algorithm 2 generate-word

```
word=(empty string)
previous-letter=(empty string)
repeat
  next-letter=pick letter randomly according to P(letter)
  word=concatenate(word,next-letter)
  if not previous-letter equals (empty string) then
    increase weight of e(previous-letter,next-letter) in G_l by 1
  end if
  previous-letter=next-letter
until random-number < w
return word
```

as observed previously in natural language in e.g. (Zipf, 1949) and (Ferrer-i-Cancho and Solé, 2002). In contrast to this, the Mandelbrot model has a step-wise rank-frequency distribution and a distorted lexical spectrum. The word generator itself is therefore already an improvement over previous models as it produces a smooth Zipfian distribution and a lexical spectrum following a power-law, incorporating a notion of letters with emergent letter probabilities. But to comply to the other requirements, the process has to be extended by a sentence generator.

The reader might notice that a similar behaviour could be reached by just setting the probability of generating a letter according to its relative frequency in previously generated words. The graph seems an unnecessary complication for that reason. But retaining the letter graph with directed edges gives rise to a more plausible morphological production in future extensions of this model, probably in a way similar to the sentence generator.

3.3.5 Sentence Generator

The sentence generator model retains a directed graph G_w. Here, vertices correspond to words and edge weights correspond to the number of times two words were generated in a sequence. The word graph is initialised with a begin-of-sentence (BOS) and an end-of-sentence (EOS) symbol, with an edge of weight 1 from BOS to EOS. When generating a sentence, a random walk starts at the BOS vertex. With a probability $(1 - s)$, an existing edge is followed from the current vertex to the next vertex according to its weight: the probability of choosing endpoint X from the endpoints of all outgoing edges from the current vertex C is given by

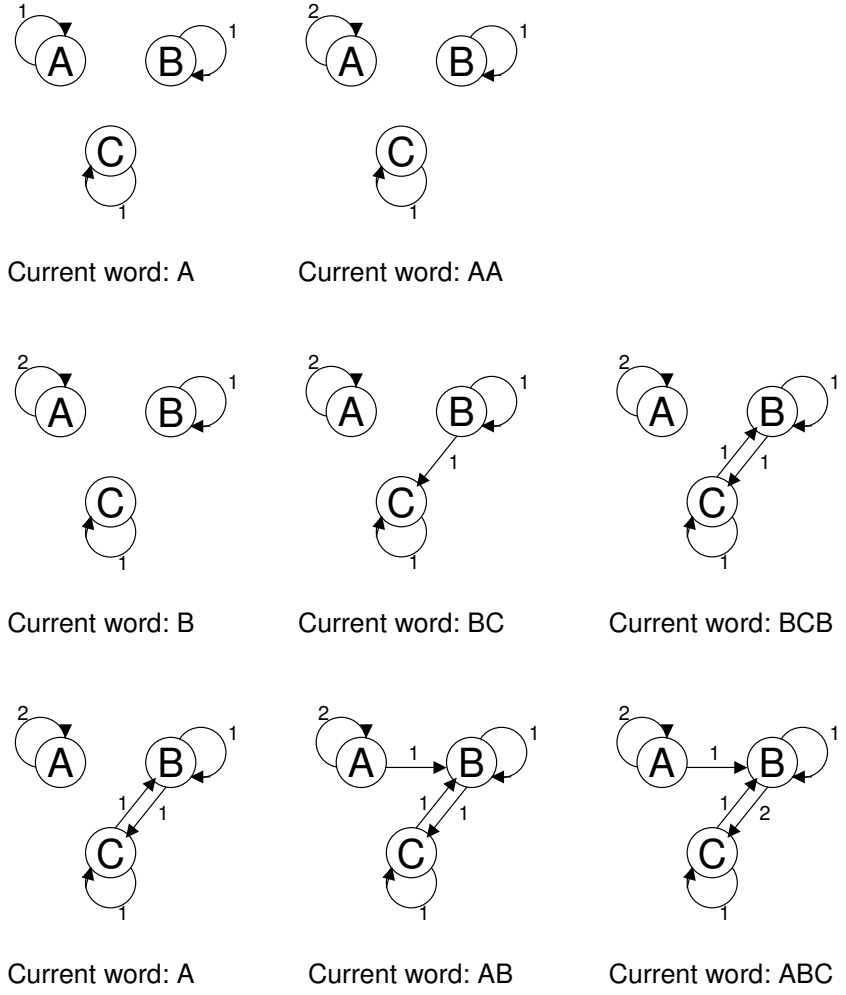


Figure 3.17: Letter graph of the word generator during the generation of three words. The small numbers next to edges are edge weights. The probability for the letters for the next step are $P(letter = A) = 0.375$, $P(letter = B) = 0.375$ and $P(letter = C) = 0.25$

$$P(word = X|C) = \frac{ew(C, X)}{\sum_{N \in neigh(C)} ew(C, N)}$$

Otherwise, with probability $s \in (0, 1)$, a new word is generated by the word generator, and a next word is chosen from the word graph in proportion to its weighted indegree: the probability of choosing an existing vertex E as successor of a newly generated word N is

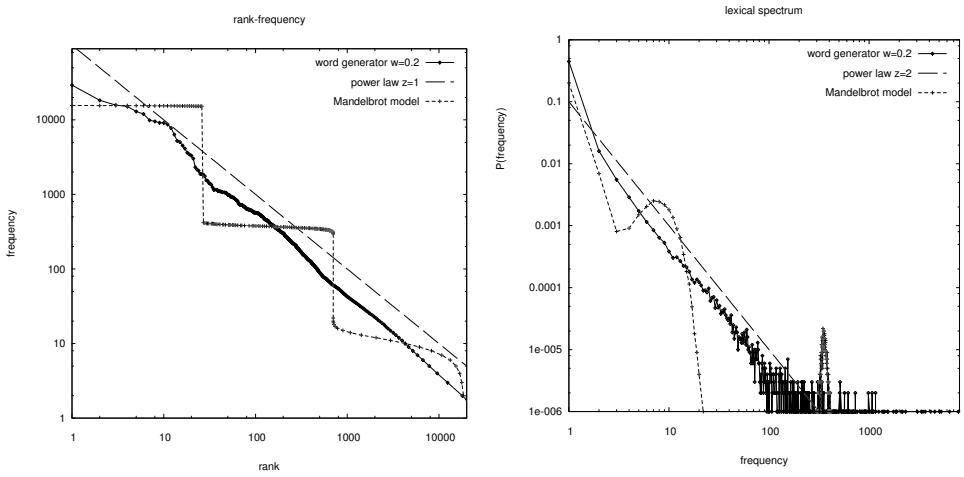


Figure 3.18: Rank-frequency distribution and lexical spectrum for the word generator in comparison to the Mandelbrot model

given by

$$P_{succ}(word = E) = \frac{indgw(E)}{\sum_{v \in V} indgw(v)}$$

with

$$indgw(X) = \sum_{v \in V} ew(v, X)$$

For each generated sequence of two words, the weight of the directed edge between them is increased by 1. Figure 3.19 shows the word graph for generating in sequence: (empty sentence), AA, AA BC, AA, (empty sentence), AA CA BC AA, AA CA CA BC. The sentence generating function is given in Algorithms 3 and 4.

Algorithm 3 initialise-word-graph G_w

insert BOS vertex into $V(G_w)$
insert EOS vertex into $V(G_w)$
insert e(BOS,EOS) into $E(G_w)$ with weight 1

During the generation process, the word graph grows and contains the full vocabulary used so far for generating in every time step. It is guaranteed that a random walk starting from BOS will finally reach the EOS vertex. It can be expected that sentence length will slowly increase during the course of generation as the word graph grows and the random walk has more possibilities before finally arriving at the EOS vertex. A similar effect, however, can be observed in human

Algorithm 4 generate-sentence

```
current-vertex=BOS
repeat
  r=random-number
  if r < s then
    new-word=generate-word
    if new-word ∈ V(G_w) then
      increase weight of e(current-vertex,new-word) by 1
    else
      insert new-word into V(G_w)
      insert e(current-vertex,new-word) into E(G_w) with weight 1
    end if
    sentence=concatenate(sentence, new-word)
    successor=pick word randomly according to P_succ(word)
    increase weight of e(new-word,successor) by 1
    sentence=concatenate(sentence,successor)
    current-vertex=successor
  else
    next-word=pick word randomly according to P(word | current-
    vertex)
    sentence=concatenate(sentence,next-word)
    increase weight of e(current-vertex,next-word) by 1
    current-vertex=next-word
  end if
until current-vertex equals EOS
return sentence
```

language learning, where children start with one- and two-word sentences and finally produce sentences of more than 1000 words in case of growing up to philosophers (see Meyer, 1967). The sentence length is influenced by both parameters of the model: the word end probability w in the word generator and the new word probability s in the sentence generator. By feeding the word transitions back into the generating model, a reinforcement of previously generated sequences is reached. Figure 3.20 illustrates the sentence length growth for various parameter settings of w and s.

When setting the new word probability s to 0 and not updating weights in G_w, the model is equivalent to a stationary Markov process with a horizon of 1. However, in this case, an initialisation with a fixed vocabulary and given transition probabilities would be necessary. Both word and sentence generator can be viewed as weighted finite automata (cf. Allauzen *et al.*, 2003) with self training.

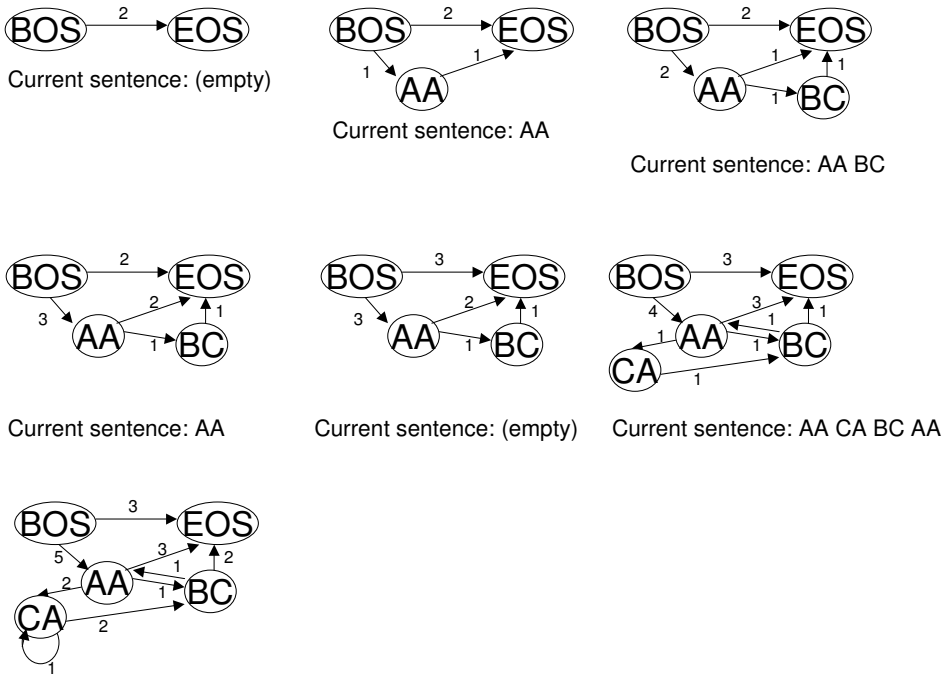


Figure 3.19: Word graph of the sentence generator model. Notice that in the last step, the second CA was generated as a new word from the word generator, and so was the last AA in the previous sentence. The generation of empty sentences (omitted in output) happens frequently

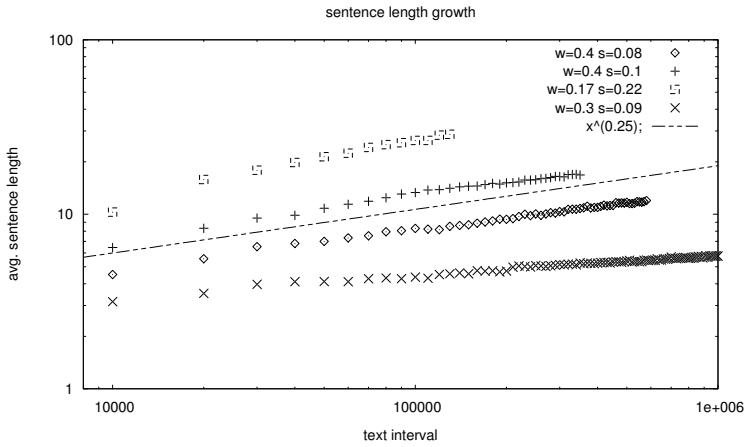


Figure 3.20: Sentence length growth, plotted in average sentence length per intervals of 10,000 sentences. The straight lines in the log-log plot indicate polynomial growth

After having defined a random text generation model, the subsequent section is devoted to testing it according to the criteria given above.

3.3.6 Measuring Agreement with Natural Language

To measure agreement with the BNC sample, random text was generated with the sentence generator with $w = 0.4$ and $N = 26$ to match the English average word length and setting $s = 0.08$. The first 50,000 sentences were skipped to reach a relatively stable sentence length throughout the sample. To make the samples comparable, 1 million words totalling 12,552 sentences with an average sentence length of 7.967 are used.

Rank-Frequency

The comparison of the rank-frequency distributions for English and the sentence generator is depicted in Figure 3.21.

Both curves follow a power-law with γ close to 1.5, in both cases the curve is flatter for high frequency words as observed by Mandelbrot (1953). This effect could not be observed to this extent for the word generator alone (cf. Figure 3.18).

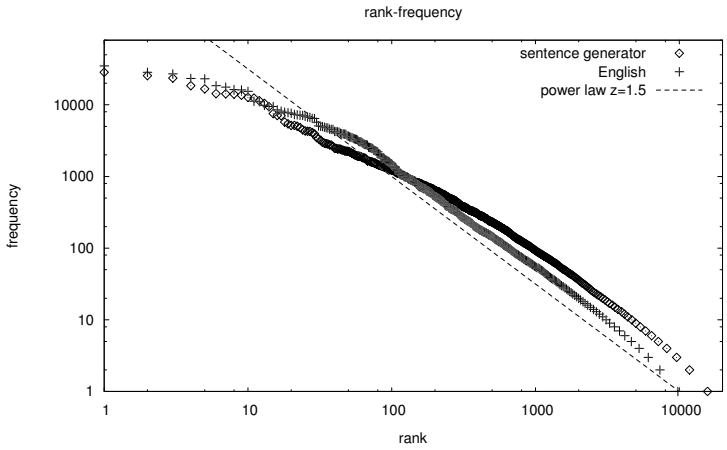


Figure 3.21: Rank-frequency plot for English and the sentence generator

Word Length

While the word length in letters is the same in both samples, the sentence generator produced more words of length 1, more words of length>10 and less words of medium length. The deviation in single letter words can be attributed to the writing system being a transcription of phonemes – and few phonemes being expressed with only one letter. However, the slight quantitative differences do not oppose the similar distribution of word length in both samples, which is reflected in a curve of similar shape in Figure 3.22 and fits well the gamma distribution variant of Sigurd *et al.* (2004).

Sentence Length

The comparison of sentence length distribution shows again a high capability of the sentence generator to model the distribution of the English sample. As can be seen in Figure 3.23, the sentence generator produces less sentences of length>25 but does not show much differences otherwise.

In the English sample, there are surprisingly many two-word sentences. Inspection of the English sample revealed that most of the surprisingly long sentences are prayers or similar items, which have been marked as a single sentence.

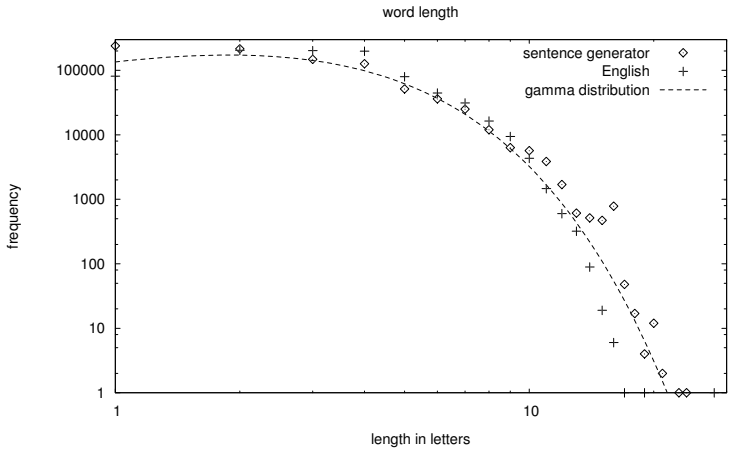


Figure 3.22: Comparison of word length distributions. The dotted line is the function as introduced in (Sigurd *et al.*, 2004) and given by $f(x) \sim x^{1.5} \cdot 0.45^x$

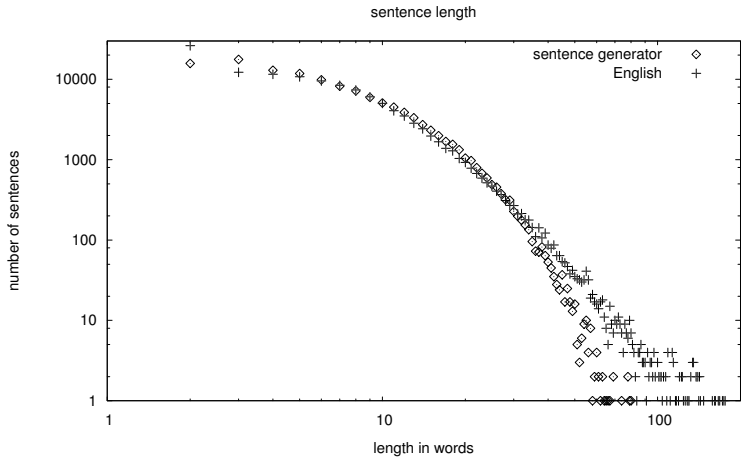
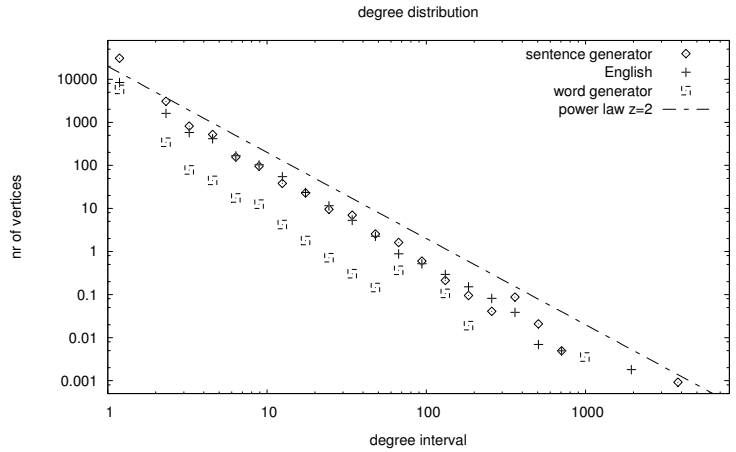


Figure 3.23: Comparison of sentence length distributions



Text	BNC sample	sent. gen.	word gen.	ER graph
n	7,154	14,234	3,498	10,000
L	2.933	3.147	3.601	4.964
$< k >$	9.445	6.307	3.069	7
C	0.2724	0.1497	0.0719	6.89E-4

Figure 3.24: Characteristics of the significant neighbour-based co-occurrence graphs of English and the generated samples of word and sentence generator

Neighbour-based Co-occurrence

Next, the structure of the significant neighbour-based co-occurrence graphs is examined. The significant neighbour-based co-occurrence graph contains all words as vertices that have at least one co-occurrence to another word exceeding a certain significance threshold. The edges are undirected and weighted by significance. Recall that the neighbour-based co-occurrence graph is scale-free and the Small World property holds, cf. Section 3.2.1.

For comparing the sentence generator sample to the English sample, log likelihood statistics on neighbouring words are computed with $t = 2$ and $s = 3.84$. For both graphs, the number of vertices, the average shortest path length, the average degree, the clustering coefficient and the degree distribution are given in Figure 3.24. Further, the characteristics of a comparable random graph as generated by the ER-model and the graph obtained from the 1 million words generated by the word generator model are shown.

From the comparison with the random graph it is clear that both neighbour-based graphs exhibit the Small World property as their clustering coefficient is much higher than in the random graph while

the average shortest path lengths are comparable. In quantity, the graph obtained from the generated sample has about twice as many vertices as the English sample, but its clustering coefficient is about half as high. This complies to the steeper rank-frequency distribution of the English sample (see Figure 3.21), which is, however, much steeper than the average exponent found in natural language. The degree distributions clearly match with a power-law with $\gamma = 2$. The word generator data produced a number of significant co-occurrences that lies in the range of what can be expected from the 5% error of the statistical test. The degree distribution plot appears shifted downwards about one decade, clearly not producing the distribution of words in sequence of natural language. Considering the analysis of the significant neighbour-based co-occurrence graph, the claim is supported that the sentence generator model reproduces the characteristics of word sequences in natural language on the basis of bigrams.

Sentence-based Co-occurrence

It is not surprising that the sentence generator produces high agreement with natural language regarding neighbour-based significant word co-occurrences, since the random text model incorporates a notion of local word order. But also when measuring sentence-based word co-occurrence, the agreement is high, as Figure 3.25 shows: the sentence-based word co-occurrence graph ($t = 2$ and $s = 6.63$) for the generated text and the BNC sample show a very similar degree distribution, whereas the word generator produces less significant co-occurrences, although the difference is not as large as for word bigrams. For low degrees, the word generator graph's degree distribution follows the power-law distribution much more closely than the sentence generator and the natural language graphs.

3.3.7 Summary for the Generation Model

A random text generation model was introduced that fits well with natural language with respect to frequency distribution, word length, sentence length and neighbouring co-occurrence. The model was not constrained by any a priori distribution – the characteristics emerged from a two-level process involving one parameter for the word generator and one parameter for the sentence generator. This is the first random text generator that models sentence boundaries beyond inserting a special blank character at random: rather, sentences are modelled as a path between sentence beginnings and sentence

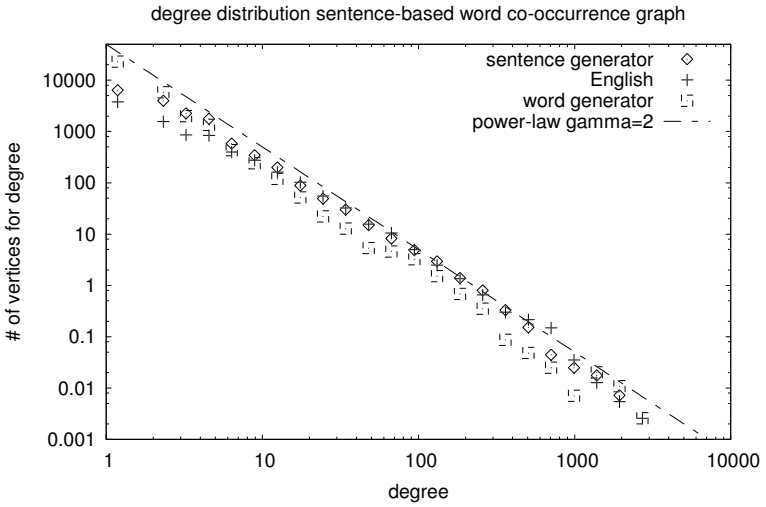


Figure 3.25: Degree distributions of the significant sentence-based co-occurrence graphs of English, sentence generator and word generator. The differences are similar to measurements on neighbour-based co-occurrences, but not as distinctive

ends which imposes restrictions on the words at sentence beginnings and endings. Considering its simplicity, a plausible model for the emergence of large-scale characteristics of language is proposed without assuming a grammar or semantics. After all, the model produces gibberish – but gibberish that is well distributed.

The studies of Miller (1957) rendered Zipf's law un-interesting for linguistics, as it is a mere artefact of language – as it emerges when putting a monkey in front of a typewriter – rather than playing an important role in its production. The model introduced here does not only explain Zipf's law, but many other characteristics of language, which are obtained with a monkey that follows beaten tracks. These additional characteristics can be thought of as artefacts as well, but I strongly believe that the study of random text models can provide insights in the process that lead to the origin and the evolution of human languages.

For further work, an obvious step is to improve the word generator so that it produces phonologically plausible sequences of letters and to intertwine both generators for the emergence of word categories. At this, letters could be replaced by phonemes, allowing only possible transitions as studied in the field of phonotactics (see Ladefoged and Maddieson, 1996). This would not mean an a-priori initialisation of the model but a necessary adaptation to the human phonetic

apparatus.

Another suggestion is to introduce more levels with similar generators, e.g. for morphemes, phrases and paragraphs. For evaluation of these additional generators, however, much more precise quantitative measures would be needed.

Furthermore, it is desirable to embed the random generator in models of communication where speakers parameterise language generation of hearers and to examine, which structures are evolutionary stable (see Jäger, 2003). This would shed light on the interactions between different levels of human communication. In this context, it would be desirable to ground the utterances to real-world situations, as carried out in the talking heads experiments of Steels (2003) and to simulate on a realistic social network, as proposed in Mehler (2007).

4 Graph Clustering

Having shown in the previous chapter that many phenomena in natural language can be modelled quantitatively with Small World graphs, this chapter is now devoted to discovering structure in these graphs by grouping their vertices together in meaningful clusters.

Clustering is the process of grouping objects in a way that similar objects are found within the same group. Informally, the similarity between objects within a group should be higher than the similarity between objects of different groups. These groups are called clusters. Clustering is data exploration contrary to classification, which is the learning of a classifier that assigns objects to a given set of classes. Clustering is also known as unsupervised machine learning.

Among researchers, there is no consensus of what exactly constitutes a cluster. In (Meilă, 2005), different measures are given that determine the quality of clusterings, yet the optimisation of a clustering with respect to one measure often results in a performance drop of another. Therefore the question cannot be how to produce an optimal clustering by itself, but to produce the best clustering for a given task, where the grouping of objects is used to achieve an improvement over not using a grouping or over an inferior way of grouping.

In Section 4.1, clustering in its broadness is briefly reviewed in general. The discipline of graph clustering is embedded into the broad field of clustering and discussed in detail and a variety of graph clustering algorithms are examined in terms of mechanism, algorithmic complexity and adequacy for Small World graphs. Taking their virtues and drawbacks into consideration, an efficient graph partitioning algorithm called Chinese Whispers is developed in Section 4.2. This algorithm will be used throughout this work to solve several NLP tasks.

4.1 Review on Graph Clustering

This section provides a review on graph clustering methods and their properties. After a introduction to clustering in general and graph clustering in particular, the distinction between spectral and non-spectral methods is drawn and several algorithms are described in

more detail.

4.1.1 Introduction to Clustering

To be able to set up a clustering method, a measure is needed that defines a similarity (or dissimilarity) value for pairs of objects, represented by features. Independent of the clustering method used, the similarity measure has a large influence on the result. Depending on the representation of objects (i.e. what kind of features are used), a plethora of similarity measures are possible and have to be chosen in order to meet the invariants of objects. For example, when clustering handwritten digits for optical character recognition, the shape of digits is more discriminative than their horizontal or vertical displacement or their size. Traditionally, objects are given as data points in a high-dimensional space, where the dimensions represent features that can be determined by examining the object. For the handwritten digits, every pixel could play the role of a feature, with the value set to the colour intensity of the pixel. The object's data point in the *feature space* is often referred to as *feature vector*.

Here, only the most basic differences of clustering methods are displayed rather than elaborated, to merely put the discipline of graph clustering into the right place. For recent extensive overviews of clustering methods, the reader is referred to (Everitt *et al.*, 2001; Berkhin, 2002).

Different methods of clustering can be classified into hierarchical methods and partitionings. Hierarchical methods arrange the objects in nested clusters, following divisive (top-down) or agglomerative (bottom-up) strategies: In divisive clustering, all objects are regarded as being contained in one large cluster in the beginning. Clusters are iteratively split until either only single objects are found in the clusters of the lowest hierarchy level or a stopping criterion is met. Agglomerative methods proceed the opposite way: in initialisation, every object constitutes its own cluster. Iteratively, clusters are joined to form larger clusters of clusters until all objects are in one big cluster or the process stops because none of the intermediate clusters are similar enough to each other. For the notion of similarity of clusters, again, various strategies are possible to compute cluster similarity based on the similarity of the objects contained in the clusters, e.g. single-link, complete-link and average-link cluster similarity, which are defined as the highest, lowest or the average similarity of pairs of objects from different clusters. Commonly used is also the distance of centroids: a cluster centroid is an artificial data point whose co-

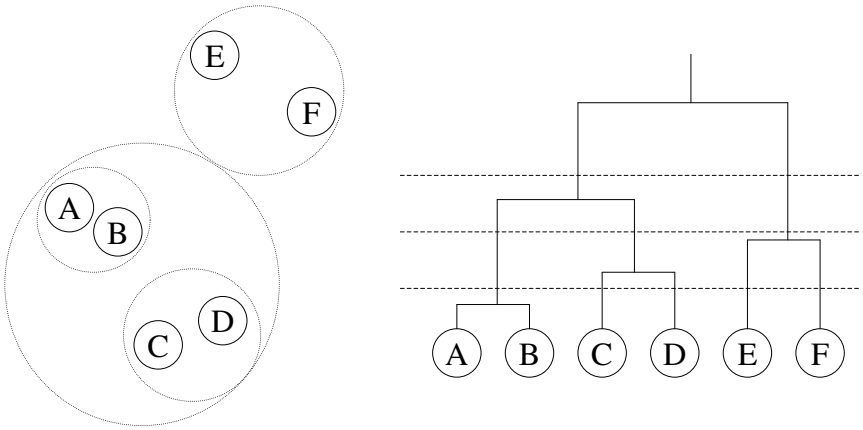


Figure 4.1: Example of hierarchical clustering. Left: data points, right: dendrogram and possible vertical cuts for partitions $\{\{A,B,C,D\},\{E,F\}\}$, $\{\{A,B\},\{C,D\},\{E,F\}\}$ and $\{\{A,B\},\{C\},\{D\},\{E\},\{F\}\}$. Dotted circles display the nested clusters in the data; for similarity, the inverse distance between centres of these circles was used

ordinates are achieved by averaging over the coordinates of all data points in the cluster. The benefit of hierarchical clustering methods, namely the arrangement of objects into nested clusters that are related to each other, is also its largest drawback: in applications that need a sensible number of non-nested clusters (such as the handwritten digits, which require 10 parts of the data), the question of where to place the vertical cut in the cluster hierarchy is non-trivial. Figure 4.1 shows the visualisation of a hierarchical clustering of points with similarity based on their distance in their 2-dimensional space, called dendrogram. The dotted lines are possible vertical cuts to turn the hierarchy into a flat partition.

The other option is to directly partition the objects into a set of disjoint sets of objects such that each object is member of exactly one cluster. This makes the vertical cut in the cluster hierarchy obsolete. The most prominent representative of partitioning algorithms is *k*-means (MacQueen, 1967). The *k*-means algorithm retains k cluster centroids which are initialised randomly in the vector space spanned by the set of features. Each data point is assigned to the centroid to which it is most similar. Then, the cluster centroids are recomputed by averaging the feature vectors of the data points assigned to them. This is done iteratively until no changes can be observed in subsequent steps. *K*-means is very efficient, yet has two disadvantages:

the number of clusters k has to be set beforehand and outcomes of different runs might differ due to the random initialisation of centroids: the algorithm gets stuck in local optima – a feature that is shared by all randomised approaches.

Most literature on clustering encompasses methods operating on data points in an n-dimensional space, and a metric is used to compute similarity values. This is due to the data, which is naturally represented as feature vectors in many settings. An alternative, which will be used throughout this work, is the graph representation. Here, objects are represented as vertices in an undirected graph, and edge weights between these vertices reflect similarity.

Traditionally, the vector space model (Salton *et al.*, 1975), originating from the field of information retrieval (see Baeza-Yates and Ribeiro-Neto, 1999, for a survey), has been widely used in NLP and examined in great detail in (Sahlgren, 2006, inter al.). In Schütze's Word Space (Schütze, 1993) and related works, an entity (e.g. a document) is represented as a high-dimensional feature vector, whereas features are usually words appearing in the document. Notice that it is straightforward to compute the graph representation from the vector space representation by constructing a graph with one vertex per object and computing pair-wise similarities on the vector space that form the weights of edges between the respective vertices. This conversion is lossy, as it is not possible to restore the original feature vectors from the similarity graph. As clustering algorithms normally operate on similarities rather than on the single feature vectors, however, this loss does not render graph-based methods less powerful.

Variants of this conversion are possible: an edge between two vertices can be drawn only if the weight exceeds a certain threshold. Notice that some entities in the vector representation might not end up in the graph in this case. Another variation could be adding only the N most similar adjacent vertices for each vertex.

There are fundamental differences between vector space and graph representation. First of all, vector spaces have dimensions, which are spanned by the features respectively the vector entries. There is nothing corresponding to a dimension in graphs – they encode similarity directly instead of assigning a location in a high-dimensional space. For this reason, the vector space notion of a centroid as the average vector of a set of vectors does not have an equivalent in graphs. A consequence of thresholding the minimum weight in the conversion is the possible unrelatedness of individual entities in the graph: whereas a suitable similarity function assigns non-zero values to all vector pairs, and thus grades the similarity scores, there is a possibly

large fraction of vertex pairs in the graph with zero similarity to all other vertices, which have to be excluded from the clustering.

The conversion algorithm outlined above has a quadratic time-complexity in the number of entities, as each possible pair of entities has to be considered. This is unwanted, as we shall see that further processing steps can have much lower time-complexities which would turn this conversion into a bottleneck. A heuristic to optimise the conversion is to index the term-document matrix by feature and to compare only pairs of entities that show to be amongst the top N entities which have this feature set to a non-zero value.

When transforming data which is represented in a vector space, the question arises why to do so. Dependent on the task to be solved, this may or may not be advantageous.

In some cases, feature vector-based techniques are not applicable, as the data is already given in a graph representation. This is often the case in the field of natural language processing, and this is why this work concentrates mainly on graph methods. For an easily digestible and yet thorough introduction to graph models in NLP, the reader is referred to (Widdows, 2004).

In the next section, two different paradigms of graph clustering are described, one operating directly on graphs, the other operating on the spectrum of its adjacency matrix.

4.1.2 Spectral vs. Non-spectral Graph Partitioning

Spectral graph clustering is based on spectral graph theory, see (Chung, 1997; Cvetkovič *et al.*, 1995) for thorough overviews of this field. Spectral methods make use of invariants of the adjacency matrix (especially its Laplacian[1]), such as characteristic polynomials, eigenvalues and eigenvectors. This is closely related to dimensionality reduction techniques like singular value decomposition (SVD) and latent semantic analysis (LSA, (Deerwester *et al.*, 1990)). E.g. the algorithm of Shi and Malik (2000) splits the vertices in two parts based on the second-smallest eigenvalue of the Laplacian. A related setup is used by Meilă and Shi (2000), where the eigenvectors for the largest k eigenvalues of a adjacency matrix invariant are used as initial centroids for k-means, which of course requires a vector space representation of the data. For spectral clustering, the eigenvectors corresponding to the largest k eigenvalues define the clusters, which means that spectral clustering always needs the number of clusters k

[1] the Laplacian matrix is given by $L = D_G - A_G$, where D_G is the degree matrix of graph G that contains the degree of vertices in the main diagonal

as an input parameter. In general, spectral methods rely on various matrix operations, most of which cause the entire algorithm to be slower than time-quadratic[2] in the number of vertices, making spectral techniques only applicable to graphs of at most several thousand vertices. As in NLP, often millions of vertices are needed to represent e.g. different words in a corpus, and dimensionality reduction techniques have generally shown to be inadequate for the power-law distributions found in natural language (cf. Levin *et al.*, 2006; Steyvers and Tenenbaum, 2005), the focus of this work will be on non-spectral methods in the remainder. Non-spectral methods operate directly on the graph or its adjacency matrix without computing the invariants as described above. Methods are often inspired by 'classical' graph theory, which is outlined in e.g. (Bollobas, 1998) or (Gibbons, 1985). In the next subsection, a number of non-spectral graph clustering algorithms are summarised.

4.1.3 Graph Clustering Algorithms

Given a graph, the question arises how it should be split into two or more parts. Intuitively, a cluster in a graph is connected with many internal edges, and there are only few edges between clusters. In graph theory, several measures have been proposed to determine how vertices and groups of vertices relate to each other.

Measures of Cluster Quality

Šíma and Schaeffer (2005) summarise several locally computable fitness functions, which are used for measuring the quality of a cluster within a graph.

The *conductance* of a cut S, $\phi(S)$, is defined as the size of the cut, divided by the minimum of the sums of degree in the parts:

$$\phi_G(S) = \frac{c_G(S)}{\min(\sum_{a \in S}(k_S(a), \sum_{b \in V/S}(k_{V/S}(b)))}$$

For clustering, the conductance of a cut should be minimised. The conductance of a weighted graph is also called *normalised cut*, as used by Shi and Malik (2000).

The local density of a subset $\emptyset \neq S \subset V$ in a graph G is the ratio of the number of edges in the subgraph $G(S)$ induced by S over the

[2] iterative improvements in matrix multiplication complexity suggest that its time complexity is quadratic, although the fastest known algorithms are slower, see (Cohn *et al.*, 2005)

number of edges in a clique of $|S|$ vertices:

$$ldensity_G(S) = \frac{2|E(S)|}{(|S| \cdot (|S| - 1))}.$$

The local density measures the ratio between actual and possible edges in a set of vertices. For clustering, the local density of parts should be maximised.

The relative density of a cut S is defined as

$$rdensity_G(S) = \frac{|E(S)|}{(|E(S)| + c_G(S))}$$

and measures the amount edges that are within the parts versus the amount that are in the cut. Cluster quality is higher for larger relative density.

Single cluster editing of a subset $S \subset V$ counts the number of edge operations (additions of edges and deletions of edges) needed to transform S into an isolated clique and can be computed as

$$sincluedit_G(S) = \binom{|S|}{2} - |E(S)| + c_G(S).$$

Single cluster editing should be minimised for a good clustering.

Many graph clustering algorithms search for clusters that directly optimise these measures (see e.g. Brandes *et al.*, 2003; Jain *et al.*, 1999; Mihail *et al.*, 2002; Schaeffer, 2005).

In (Šíma and Schaeffer, 2005), proofs are given that optimising each single of the four measures is NP-complete, i.e. the number of computations needed for the optimisation is exponential in the number of vertices, making the optimisation intractable: there exist no efficient solutions for large numbers of vertices[3]. Therefore, algorithms that optimise these measures are theoretically justified, but are not applicable for large graphs because their run-times are too long to be of practical use. Other NP-complete measures and approximations thereof can be found in (Bern and Eppstein, 1997). In (Kannan *et al.*, 2000), cases where the optimisation of graph measures even leads to undesired results are discussed and exemplified.

Desiderata on Graph Clustering for Natural Language

Graph partitioning has been of practical use mainly in the field of very large scale integration (VLSI), being a part of hardware design

[3] if $P \neq NP$ holds for the classes P and NP in computational complexity, which is unproven, yet widely accepted

for processors. A major question in VLSI is now to distribute digital logic on several chips. Atomic circuits can be modelled as vertices in a graph, where edges represent dependencies between circuits. To arrive at a performant architecture, the number of dependencies between different chips should be minimised, because the transfer between chips is much slower than within chips. An additional constraint in VLSI is that the parts are of roughly equal size to implement the overall circuit on several equisized chips. See (Fjällström, 1998) for a survey on graph partitioning in VLSI and (Hauck and Borriello, 1995) for an evaluation of bipartitioning techniques. Unfortunately, methods developed for VLSI are only applicable with restrictions to language data, as the constraint of a fixed number of equisized parts does not reflect the natural structure of language: as indicated in Chapter 3, the distribution of cluster sizes in language data can be expected to follow a power-law, which implies a highly skewed distribution on natural cluster sizes rather than clusters of similar scale. As an example, the number of word types in syntactic word classes differ considerably: whereas open word classes like nouns, adjectives and verbs contain a large number of members, closed word classes like determiners, prepositions and pronouns only posses a few. A clustering algorithm on word classes that aims at equisized clusters would split open word classes into several clusters while crudely lumping closed word classes together. Another difference between applications like VLSI and natural language is that in contrast to an engineer's task of distributing a current network over a fixed number of chips, in natural language it is often not known beforehand, how many clusters is a sensible choice. Think of the number of topics in a text collection: without reading the collection, a user will not be able to specify their number, but clustering into topics is exactly what a user wants in order to avoid reading the whole collection. Other examples include the number of different languages in a random internet crawl, which will be discussed in depth in Section 5.1, and the number of word meanings in word sense induction, see Section 5.3.

Facing the NP-completeness of the measures above, a constructor of an efficient graph clustering algorithm is left with two choices: either to optimise a measure that is not NP-complete, or to give computationally low-cost heuristics that only approximate one of the measures above. Both strategies can be observed in a selection of graph clustering algorithms that will be presented now.

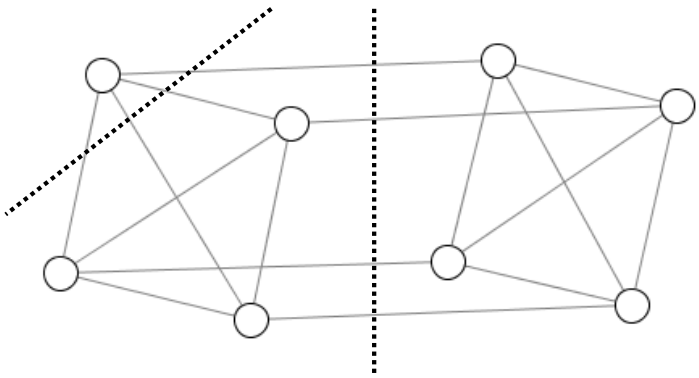


Figure 4.2: Two possible minimal cuts for an unweighted graph of 8 vertices. Cutting only one vertex off is counter-intuitive and does not optimise any of measures given above

Graph Clustering Review

The most widely known polynomial-time graph measure is *mincut* (minimal cut): a graph is (iteratively) split in a top-down manner into two parts in each step, such that the size of the cut is minimised. Finding a minimal cut is equivalent to finding the maximal flow between two vertices in a current network as stated in the max-flow min-cut theorem (Ford and Fulkerson, 1956). The global minimal cut is achieved by minimising over max-flows between all pairs of vertices. The mincut problem can be solved for weighted, undirected graphs in $O(|E| + log^3(|V|)$ as proposed by Karger (1996) and is therefore a measure that has viable run-time properties for graphs with several thousand vertices. For larger graphs, Wu and Leahy (1993) describe a method to transform the graph into a cut-equivalent tree that allows even faster processing times. Another alternative of scaling is the use of parallelisation (see Karger, 1993, 1996). Mincut is used for building trees on graphs by Flake *et al.* (2004) for web community identification. A problem of mincut, however, is that several minimum cuts can exist, of which many can be counter-intuitive. Figure 4.2 illustrates two possible outcomes of mincut on an unweighted graph with cut size of 4.

The counter-intuitiveness of the 'bad' minimal cut in Figure 4.2 originates in the fact that one of the 4-cliques is cut apart. To overcome this flaw by balancing cluster sizes, *ratio cuts* were used by Wei and Cheng (1991) for VLSI and normalised cuts by Shi and Malik (2000) for image segmentation. In (Shi and Malik, 2000), a spectral

approximation for minimising conductance only results in bearable run-times for pictures of some 10,000 pixels when taking properties of image segmentation into account: e.g. that the graphs are only locally connected – again a property that does not hold for graphs in natural language, opposing the shortest average path lengths in word graphs, see Chapter 3. When using mincut to iteratively split a graph in a hierarchical setting, the number of clusters or a maximum cut value has to be defined, making the methods subject to individual tuning for each dataset.

More appropriate for the clustering of Small World graphs seems Small World Clustering (Matsuo, 2002). Here, the weighted sum of average shortest path length and clustering coefficient are to be maximised by removing and adding edges until a local maximum is found. Non-connected components of the resulting graph constitute the clusters. The author tests the approach on a small word co-occurrence network. Since the computation of both measures is computationally expensive, a more efficient heuristic would be needed in order to make the procedure feasible for large graphs.

Much faster and also employing the Small World property is the HyperLex algorithm of Veronis (2004). Here, the following steps are repeated on an undirected graph, edge-weighted by dissimilarity of vertex pairs: the largest hub is identified as being a root vertex, and all of its neighbours are deleted from the graph, until no more hub vertex with a sufficient degree and local clustering coefficient can be found. These root vertices form the first level of a minimum spanning tree (that is, the tree containing all vertices of the graph with the minimum total edge weight), which contains the clusters of vertices in its first level subtrees. The minimum spanning tree can be computed in $O(|E|log|E|)$, which makes the overall procedure very efficient. HyperLex was developed and evaluated for word sense induction, see Section 5.3. The various parameters for determining whether a vertex is counted as hub, however, are subject to application-dependent fine-tuning.

An interesting point is raised by Ertöz *et al.* (2002) in the context of graph-based document clustering: a general problem with clustering algorithms is that regions with a higher density, i.e. many data points in a local neighbourhood in the vector space or highly connected subgraphs in the graph model, tend to be clustered together earlier than regions of low density. It might be the case that in hierarchical settings, two dense clusters that are close to each other are agglomerated while sparse regions are still split into small pieces, resulting in an unbalanced clustering. Figure 4.3 shows an illustrating

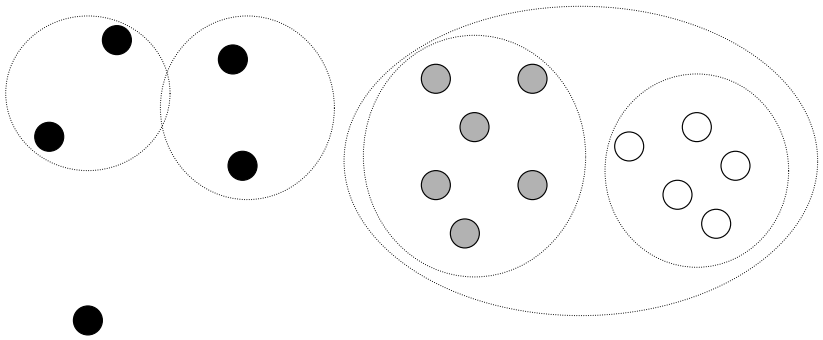


Figure 4.3: Dense and sparse regions: the dense clusters are joined before the sparse region is clustered. In hierarchical settings, this situation leads to no vertical cut that would produce the intuitively expected clusters. In partitioning the same problem arises with either too many small clusters in the sparse region or too large clusters in the dense region. Shades indicate the desired partition; the dotted circles show two levels of hierarchical clustering

example.

To overcome this problem, Ertöz *et al.* (2002) introduce self scaling in the graph model: for each vertex, only the N most similar (highest-weighted) neighbours are considered. A new graph is computed from this top-N graph by counting the number of different paths of length 2 between any pair of vertices and assigning the result as edge weights in the new graph. The more common neighbours two vertices had in the top-N graph, the higher is their resulting weight in the new graph, cf. the higher order transformation described in Section 3.2.2. As the operation of taking the top N neighbours is not attributed with a weight or similarity threshold, this operation adapts to the density differences in the original graph. Further, Ertöz *et al.* (2002) describe the necessity for various similarity thresholds: for clustering, a higher threshold is used to determine the partition. As thresholding might result in a portion of vertices without edges, these are attributed to their nearest cluster in a subsequent step that uses a lower threshold. In this way, vertices with low edge weights do not contribute to the cluster structure but are nevertheless part of the final clustering.

Elaborating on the bipartite structure of many datasets (e.g. terms and documents in information retrieval), Zha *et al.* (2001) apply methods designed for general graphs to the bipartite setting and define bipartite cuts and bipartite normalised cuts. The divisive binary

clustering algorithm described there is time-linear in the number of vertices, which is achieved by an approximation using singular value decomposition (which has to be computed beforehand). Here, again, the number of relevant dimensions determines the number of clusters and has to be provided as a parameter.

Another way of looking at graph clustering is based on the notion of a random walk on graphs. A random walk is the formalisation of the idea of taking successive steps at random along possible paths. In graphs, a random walk is a path through vertices of a graph along the edges; in weighted graphs, the probability of walking along an edge is proportional to its weight. Random walks on graphs can be viewed as Markov chains: the next step is only dependent on the vertex (state) where the random walker is positioned at a certain time step, and time invariance is given by the fact that the probabilities of successor vertices (edge weights) are invariant of specific time steps.

Random walks have been applied to rank vertices according to their prominence in PageRank (Brin and Page, 1998) for internet search engines, further for text summarisation in (Mihalcea and Tarau, 2004) and (Erkan and Radev, 2004) and for word sense induction by Agirre *et al.* (2006b). Prominence ranking using random walks is based on the intuition that the more important a vertex is, the more often it is visited by a long random walk.

Using random walks in clustering is based on another idea: random walks starting at some point in a cluster should be ending up in the same cluster after a couple of steps more likely than ending up in another cluster because of a higher inter-cluster than intra-cluster connectivity that constitutes a desired clustering. This idea is employed in Markov Chain Clustering (MCL, (van Dongen, 2000)). Since MCL has been used in NLP for word sense induction by Widdows and Dorow (2002) and can be viewed as a generalisation of a clustering algorithm that will be introduced in the next section, it is explained in more detail.

MCL is the parallel simulation of all possible random walks up to a finite length on a graph G. MCL simulates flow on a graph by repeatedly updating transition probabilities between all vertices, eventually converging to a transition matrix after k steps that can be interpreted as a clustering of G. This is achieved by alternating an expansion step and an inflation step. The expansion step is a matrix multiplication of the adjacency matrix A_G with the current transition matrix. The inflation step is a column-wise non-linear operator that increases the contrast between small and large transition probabilities and normalises the column-wise sums to 1. The k matrix mul-

tiplications of the expansion step of MCL lead to a time-complexity worse than quadratic. The outcome of the algorithm is deterministic and depends sensitively on four parameters that have to be fine-tuned by the user: a loop weight for allowing the random walk to stay at a vertex, two inflation constants for influencing the cluster granularity, and an initial prefix length for manipulating the input data. The number of clusters is found by the method itself.

It has been observed by van Dongen (2000) that only a few early iterations operate on dense matrices – when using a strong inflation operator, matrices in the later steps tend to be sparse. The author further discusses pruning schemes that only keep a few of the largest entries per column dependent on an additional pruning parameter, leading to drastic optimisation possibilities. But the most aggressive sort of pruning is not considered: only keeping one single largest entry. This and other optimisations will be undertaken in the next section, at the price of ending up with a randomised and non-deterministic, yet more efficient algorithm.

4.2 Chinese Whispers Graph Clustering

In this section, the Chinese Whispers (CW) graph clustering algorithm is described. CW was first published in (Biemann and Teresniak, 2005) and later more formally described in (Biemann, 2006b), both publications form the basis for this section. Additionally, extensions of CW are discussed. The development of this algorithm is one of the major contributions of this thesis, and the algorithm is used throughout Chapter 5 to realise Structure Discovery procedures.

As data sets in NLP are usually large, there is a strong need for efficient methods, i.e. of low computational complexities. In the previous section, several graph clustering algorithms were examined on their utility for these datasets. It turned out that algorithms that exactly optimise graph-theoretical measures are mostly intractable, and that most approximations either aim at producing equisized clusters or take the number of desired clusters as input parameter. Now, a very efficient graph clustering algorithm is introduced that is capable of partitioning very large graphs in comparatively short time. This is realised by merely taking local properties of vertices into account, in contrast to optimising global measures like the minimal cut. The number of parts emerges in the process and does not have to be defined by the user. While this is reached by a non-deterministic and formally not converging algorithm, the method is applicable efficiently to datasets that are prohibitively large for most other meth-

ods.

4.2.1 Chinese Whispers Algorithm

CW is a very basic – yet effective – algorithm to partition the vertices of weighted, undirected graphs. The name is motivated by the eponymous children's game, where children whisper words to each other[4]. While the game's goal is to arrive at some funny derivative of the original message by passing it through several noisy channels, the CW algorithm aims at finding groups of vertices that broadcast the same message to their neighbours. It can be viewed as a simulation of an agent-based social network; for an overview of this field, see (Amblard, 2002). The algorithm is outlined as follows:

Algorithm 5 Standard Chinese Whispers CW(graph $G(V, E)$)

```
for all v_i ∈ V do
  class(v_i) = i
end for
for it=1 to number-of-iterations do
  for all v ∈ V, randomised order do
    class(v)=predominant class in neigh(v)
  end for
end for
return partition P induced by class labels
```

Intuitively, the algorithm works in a bottom-up fashion: first, all vertices get different classes. Then the vertices are processed in random order for a small number of iterations and inherit the predominant class in the local neighbourhood. This is the class with the maximum sum of edge weights to the current vertex. In case of multiple predominant classes, one is chosen randomly. This behaviour on predominant class ties and the random processing order of vertices within one iterations render the algorithm non-deterministic.

Regions of the same class stabilise during the iteration and grow until they reach the border of a stable region of another class. Notice that classes are updated continuously: a vertex can obtain classes from the neighbourhood that were introduced there in the same iteration. The fractions of a class a in the neighbourhood of a vertex v is computed as

$$fraction(a, v) = \frac{\sum_{w \in neigh(v), class(w)=a} ew(v, w)}{\sum_{w \in neigh(v)} ew(v, w)},$$

[4] the name was given by Vincenzo Moscati in a personal conversation

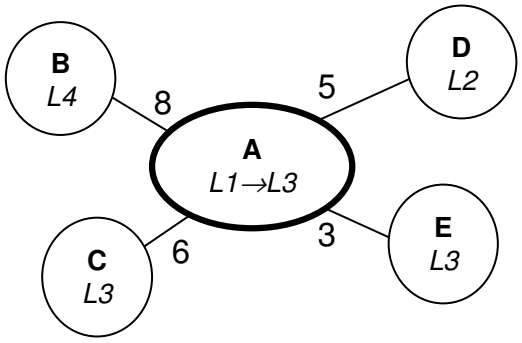


Figure 4.4: The class label of vertex A changes from L1 to L3 due to the following scores in the neighbourhood: L3:9, L4:8 and L2:5

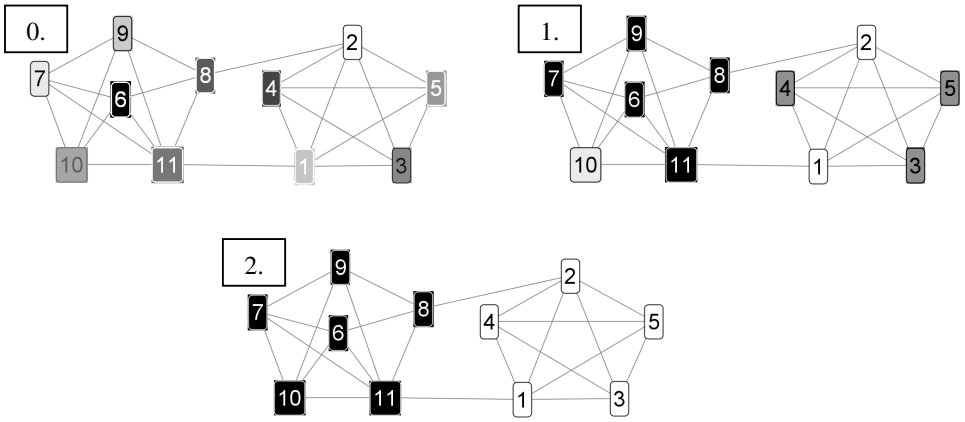


Figure 4.5: Clustering an 11-vertices graph with CW in two iterations

the predominant class a in the neighbourhood of v is given by

$$\arg\max_a fraction(a, v).$$

For each class label in the neighbourhood, the sum of the weights of the edges to the vertex in question is taken as score for ranking. Figure 4.4 illustrates the change of a class label.

With increasing iterations, clusters become self-preserving: if a strongly connected cluster happens to be homogeneous with respect to class labels, it will never be infected by a few connections from other clusters.

Figure 4.5 illustrates how a small unweighted graph is clustered into two regions in two iterations. Different classes are symbolised by different shades of grey.

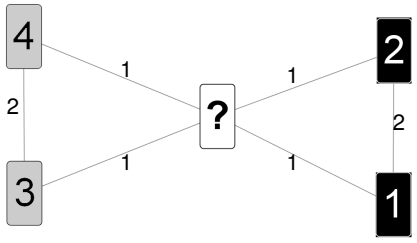


Figure 4.6: The middle vertex gets assigned the grey or the black class. Small numbers denote edge weights

The CW algorithm cannot cross component boundaries, because there are no edges between vertices belonging to different components along which class labels could be spread. Further, vertices that are not connected by any edge are discarded from the clustering process, which possibly leaves a portion of vertices unclustered. Formally, CW does not converge, as Figure 4.6 exemplifies: here, the middle vertex's neighbourhood consists of a tie that can be decided in assigning the class of the left or the class of the right vertices in any iteration all over again. Ties, however, do not play a major role in most weighted graphs.

Apart from ties, the classes usually do not change any more after a handful of iterations. The number of iterations depends on the diameter of the graph: the larger the distance between two vertices is, the more iterations it takes to percolate information from one to another.

The result of CW is a hard partition of the given graph into a number of parts that emerges in the process – CW is parameter-free. It is possible to obtain a soft partition by assigning a class distribution to each vertex, based on the weighted distribution of (hard) classes in its neighbourhood in a final step.

In an unweighted setting, isolated n-cliques always get assigned the same class label. This is due to the reason that a vertex typing with two different class labels in a n-clique is not stable under CW iterations, as there are always more edges to the larger one of the parts than to vertices having the same label for sizes up to $\frac{n}{2}$. Recursively applying this argument leads to homogeneity of classes in n-cliques.

Another algorithm that uses only local contexts for time-linear clustering is DBSCAN as described in (Ester *et al.*, 1996), which depends on two input parameters (although the authors propose an interactive approach to determine them). DBSCAN is especially suited for graphs with a geometrical interpretation, i.e. the objects

have coordinates in a multidimensional space. A quite similar algorithm to CW is MAJORCLUST (Stein and Niggemann, 1999), which is based on a comparable idea but converges slower, because vertex type changes do not take effect until the next iteration.

As CW can be viewed as a special and altered case of MCL (van Dongen (2000), see Section 4.1.3), similarities and differences are examined. The main contribution in the complexity of MCL originates from the multiplication of non-sparse matrices, which is alleviated through pruning most row entries to zero. In CW, the same pruning is done but with the difference of keeping only one single non-zero entry per row, i.e. one class label per vertex.

One could try writing CW as a matrix process in the style of MCL (see van Dongen, 2000). Let `maxrow(.)` be an operator that operates row-wise on a matrix and sets all entries of a row to zero except the largest entry, which is set to 1. Then, the algorithm is denoted as simple as this:

Algorithm 6 matrix-CW(adjaceny matrix A_G)

$D_0 = I^n$
for t=1 to number-of-iterations **do**
  D_{t-1} = maxrow(D_{t-1})
  $D_t = D_{t-1} A_G$
end for

In Algorithm 6, t is time step, I^n is the identity matrix of size $n \times n$, A_G is the adjacency matrix of graph G and D_t is the matrix retaining the clustering at time step t.

By applying `maxrow(.)`, D_{t-1} has exactly n non-zero entries. This causes the time-complexity to be dependent on the number of edges, namely $O(|E|)$. In the worst case of a fully connected graph, this equals the time-complexity of MCL. However, as shall be clear from Section 2.2, Small World graphs are rather sparsely connected. A problem with the matrix CW process is that it does not necessarily converge to an iteration-invariant class matrix D_∞, but rather to a pair of oscillating class matrices. Figure 4.7 shows an example.

This is caused by the stepwise update of the class matrix: entries changed in the previous iteration do not take effect until the next iteration. As opposed to this, the standard CW as outlined in Algorithm 5 immediately updates D after the processing of each vertex. Apart from pruning with the `maxrow(.)` operator, this is the main difference between CW and MCL. Due to the immediate update mechanism, standard CW can not be expressed as a process involving matrix multiplications. To avoid these oscillations, alternative measures

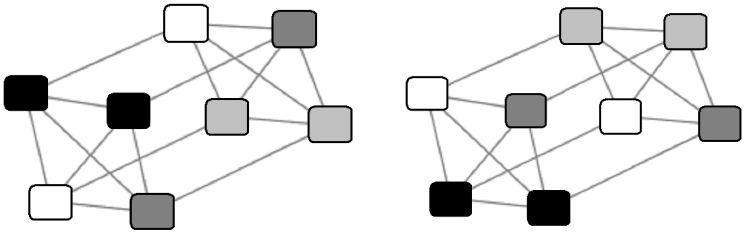


Figure 4.7: Oscillating states in matrix CW for an unweighted graph

could be taken:

- *Random mutation*: with some probability, the maxrow operator places the 1 for an otherwise unused class.
- *Keep class*: with some probability, the row is copied from D_{t-1} to D_t.

While converging to the same limits, the continuous update strategy converges the fastest because prominent classes are spread much faster in early iterations. Random mutation showed to have negative effects when partitioning small graphs, as mutation weakens clusters in their formation phase, which then possibly gets overtaken by the labels of a strong neighbouring cluster.

4.2.2 Empirical Analysis

The analysis of the CW process is difficult due to its nonlinear and randomised nature. Its run-time complexity indicates that it cannot directly optimise most global graph cluster measures mentioned in Section 4.1.3. Therefore, experiments on synthetic graphs are performed to empirically arrive at an impression of the algorithm's abilities. All experiments were conducted with an implementation following Algorithm 5. Experiments with synthetic graphs are restricted to unweighted graphs, if not stated otherwise.

A cluster algorithm should keep dense regions together while cutting apart regions that are sparsely connected. The highest density is reached in fully connected sub-graphs of n vertices, a.k.a. n-cliques. An *n-bipartite-clique* is defined as a graph of two n-cliques, which are connected such that each vertex has exactly one edge going to the clique it does not belong to. Figures 4.7 and 4.8 are n-bipartite cliques for $n = 4, 10$.

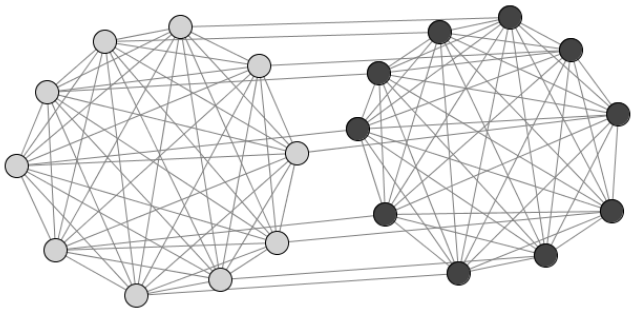


Figure 4.8: The 10-bipartite clique

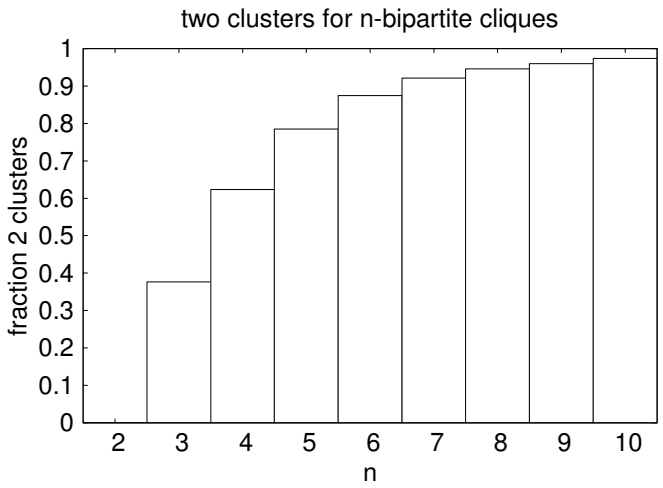


Figure 4.9: Percentage of obtaining two clusters when applying CW on n-bipartite cliques. Data obtained from 10,000 runs/n

From a graph clustering algorithm, it can be clearly expected to cut the two cliques apart. In the unweighted case, CW is left with two choices: producing two clusters of n-cliques or grouping all vertices into one cluster (recall that n-cliques always receive a homogeneous class labelling). This is largely dependent on the random choices in very early iterations – if the same class is assigned to several vertices in both cliques, it will finally cover the whole graph. Figure 4.9 illustrates on what rate this happens on n-bipartite-cliques for varying n.

It is clearly a drawback that the outcome of CW is non-deterministic. Only two third of the experiments with 4-bipartite cliques resulted in separation. But the problem is only dramatic on

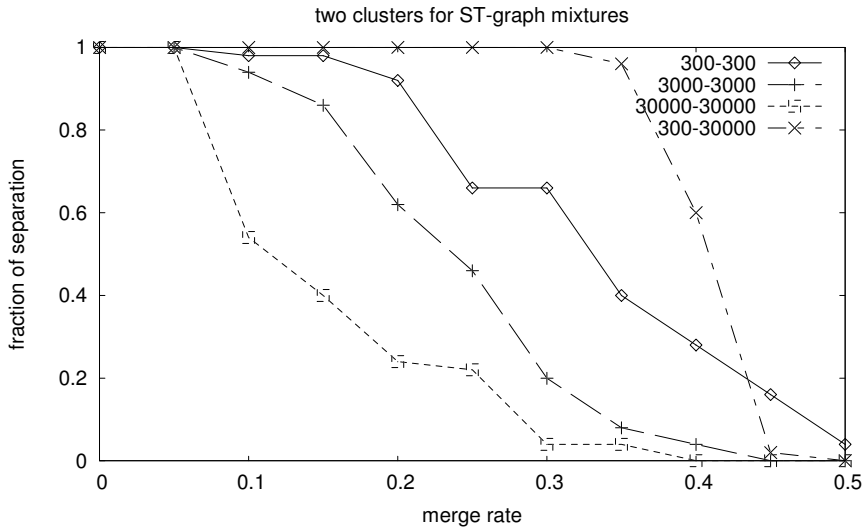


Figure 4.10: Rate of obtaining two clusters for mixtures of SW-graphs dependent on merge rate r. Data obtained for 5 mixtures and 10 runs per mixture for each data point

small graphs and ceases to exist for larger graphs as demonstrated in Figure 4.9. These small clustering problems can also be solved exactly with other clustering algorithms – interesting in the context of NLP is the performance on larger graphs and graphs with the Small World property.

For the next experiment, it is assumed that natural systems can be characterised by Small World graphs. If two or more of those systems interfere, their graphs are joined by merging a part of their vertices, retaining their edges. A graph clustering algorithm should split the resulting graph in its original parts, at least if not too many vertices were merged. To measure CW's performance on SW-graph mixtures, graphs of various sizes were generated and merged in various extents in a pair-wise fashion. Now it is possible to measure the amount of cases where clustering with CW leads to the reconstruction of the original parts. Graphs were generated using the Steyvers-Tenenbaum model (see Section 2.2): the mean connectivity $2M$ was fixed to 20, the number of vertices n and the merge rate r, which is the fraction of vertices of the smaller graph that is merged with vertices of the larger graph, was varied.

Figure 4.10 summarises the results for equisized mixtures of 300, 3,000 and 30,000 vertices and mixtures of 300 with 30,000 vertices.

It is not surprising that separating the two parts is more difficult

Iter	1	2	3	5	10	20	30	40	49
1K	1	8	13	20	37	58	90	90	91
10K	6	27	46	64	79	90	93	95	96
7ling	29	66	90	97	99.5	99.5	99.5	99.5	99.5

Table 4.1: Normalised Mutual Information values for three graphs and different iterations in %

for higher r. Results are not very sensitive to size and size ratio, indicating that CW is able to identify clusters especially if they differ considerably in size. The separation quality depends on the merge rate, but also on the total number of common vertices.

As the algorithm formally does not converge, it is important to define a stopping criterion or to set the number of iterations. To show that only a few iterations are needed until almost-convergence, the normalised Mutual Information (nMI) between the clustering in the 50th iteration and the clusterings of earlier iterations was measured. The normalised Mutual Information between two partitions X and Y is defined as

$$nMI(X,Y) = \frac{(H(X) + H(Y) - H(X,Y))}{\max(H(X), H(Y))}$$

with entropy

$$H(X) = -\sum_{i=1..n} p(X_i) log(p(X_i))$$

for $p(X_i)$ probability of choosing from part X_i when selecting a vertex from X randomly, $X_1..X_n$ forming a partition of X, and

$$H(X,Y) = -\sum_{x,y} p_{x,y} log(p_{x,y})$$

joint entropy of X and Y. A value of 0 denotes independence, 1 corresponds to congruence of partitions X and Y.

This was conducted for two unweighted ST-graphs with 1,000 (1K) and 10,000 (10K) vertices, $M = 5$ and a weighted 7-lingual co-occurrence graph (cf. Section 5.1, as used in (Biemann and Teresniak, 2005)) with 22,805 vertices and 232,875 edges. Table 4.1 indicates that for unweighted graphs, only small changes happen after 20-30 iterations. In iterations 40-50, the normalised MI-values do not improve any more. The weighted graph converges much faster due to fewer ties and reaches a stable plateau after only 6 iterations.

To summarise the empirical analysis, experiments with synthetic graphs showed that for small graphs, results can be inconclusive due

to the randomised nature of CW. But while there exist a huge variety of clustering approaches that can deal well with small graphs, its power lies in its capability of handling very large graphs in edge-linear time. The applicability of CW rather reaches out in size regions, where other approaches' solutions are intractable. In Chapter 5, CW will be evaluated in the context of unsupervised NLP applications. But before, I will discuss extensions, some of which will be used in Chapter 5 as well.

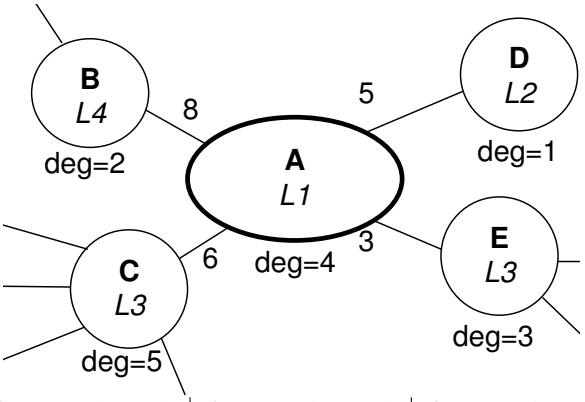
4.2.3 Weighting of Vertices

Until now, the update step of CW treats all neighbouring vertices equally. That means, the strength of neighbouring class labels only depends on the edge weights to the vertex to be updated, and not on other properties. If a graph possesses scale-free Small World structure, then it contains vertices with very high degrees (hubs), which try to propagate their class label to a large number of vertices. As the nature of those hubs implies that they have connections to many regions of the graph, this effect might be unwanted, as class labels can tunnel through them, probably resulting in several intuitive parts to obtain the same class – this especially happens if several hubs obtain the same class in early iterations. A possibility would be the deletion of hubs (see (Albert *et al.*, 2000) for attacking the connectivity of graphs) – hence it is not clear how many of the vertices with very high degrees should be deleted. Another possibility is to downgrade the influence of hubs by assigning lower vertex weights vw to them. For the vertex weighted version, the fraction of a class a in the neighbourhood of a vertex v is rewritten as:

$$fraction(a,v) = \frac{\sum_{w \in neigh(v), class(w)=a} vw(w) \cdot ew(v,w)}{\sum_{w \in neigh(v)} vw(w) \cdot ew(v,w)}.$$

Note that the incorporation of vertex weights is a real extension, as the function $vw(.)$ is arbitrary and can lead to different effective weights of edges, dependent on which vertex is to be updated. For downgrading hubs, $vw(x)$ can be defined to be dependent on the degree of vertex x. Here, two variants are examined: $vw_{LIN}(x) = 1/k(x)$ and $vw_{LOG}(x) = 1/\log(1+k(x))$. Dependent on vertex weighting, different outcomes of update steps can be achieved, as Figure 4.11 illustrates.

Weighting down hubs should result in more and smaller clusters. To test this hypothesis, a ST-network (see Section 2.2) with 100,000 vertices and an average degree of $2M = 5$ was generated



v. weight.	fraction(L2,A)	fraction(L3, A)	fraction(L4, A)	winner class
constant	5/22	9/22	8/22	L3
vw_{LIN}	0.379	0.318	0.303	L2
vw_{LOG}	0.360	0.276	0.364	L4

Figure 4.11: Effects of vertex weighting in the neighbourhood of vertex A. The table summarises the fractions of the different classes in the neighbourhood for different ways of vertex weighting. Standard CW is equivalent to constant vertex weighting

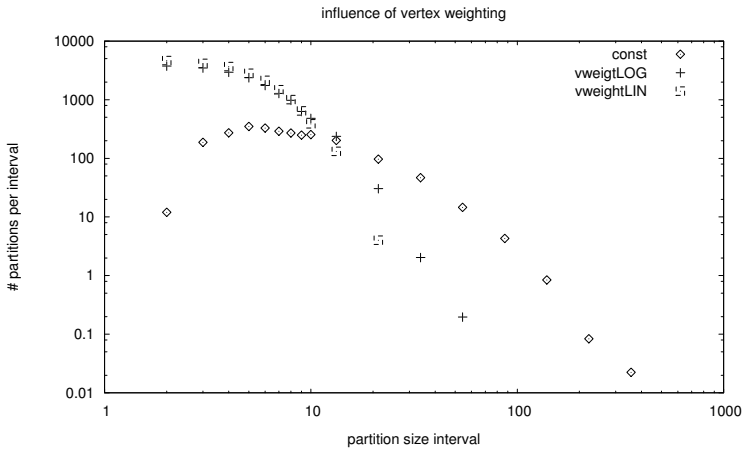


Figure 4.12: Cluster size distribution for different vertex weighting schemes

and clustered using the different vertex weighting schemes. Figure 4.12 shows the distribution of cluster sizes.

The data presented in Figure 4.12 supports the claim that the more rigid hubs get weighted down according to their degree, the more and the smaller clusters arise. Whereas in this experiment the largest cluster for the standard setting was 358 vertices, it was 46 vertices

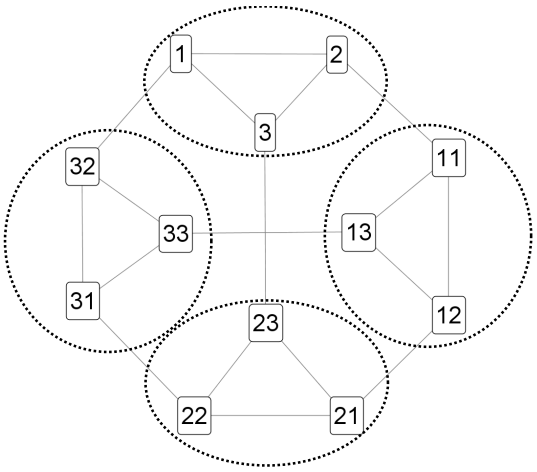


Figure 4.13: Topped tetrahedron graph with plausible partition

using vw_{LOG} and only 21 vertices in the case of vw_{LIN}. In summary, vertex weighting provides a means for influencing the granularity of the partition.

4.2.4 Approximating Deterministic Outcome

It might seem an undesired property of CW that different runs on the same graph can produce different outcomes, which is caused by the randomised nature of CW. This behaviour does not result in severe performance drops on real world data (see Chapter 5), because there are several ways to split a graph in equally motivated clusters, which result in similar performance in applications. Still, a possibility of alleviating this problem is discussed here shortly. Consider the topped tetrahedron graph (cf. van Dongen, 2000) in Figure 4.13.

A plausible partition would split this graph into four parts, consisting of one triangle each. As in the experiments with n-bipartite graphs, however, CW is able to find this partition, but can assign the same class to arbitrary combinations of these four parts, yielding 1, 2, 3 or 4 parts for this graph. *Deterministic Chinese Whispers* combines several runs of standard CW in the following way: amongst all runs, the run having produced the most parts is chosen. This serves as a reference run all other partitions are aligned to. For each other run, a confusion matrix of class labels is computed: looking up the classes for every vertex in both partitions, the confusion matrix contains the number of times the two class labels of the reference run and the other run are found together. For the classes of the other run, a fuzzy

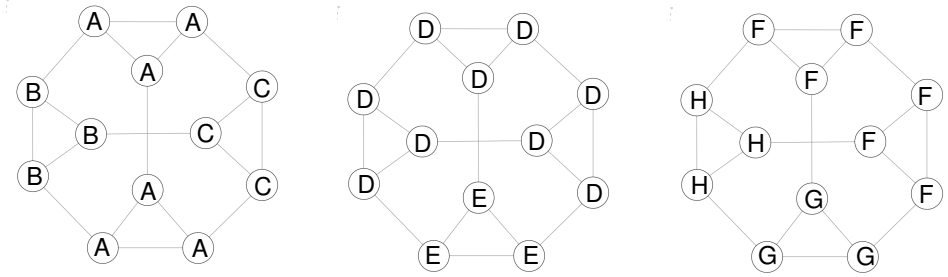


Figure 4.14: Three partitions of the topped tetrahedron. The first run is selected for reference

mapping to the reference classes is then given by the vector of reference classes per other class. The final result is obtained by replacing the classes in all runs by their reference class vectors and summing up the vectors position-wise. Vertices with the same reference vectors belong to the same class. Figure 4.14 shows an example for three different partitions of the topped tetrahedron graph.

From the example it should be clear that the finest-grained partition possible is found by merging possibly overlapping partitions. The result partition does not have to be contained in any of the runs, and can be sometimes finer-grained than any partition of a single run: the graph in Figure 4.6 is split into three clusters in the result, while in both possible outcomes of single CW runs only two clusters are found. Picking the run with the maximum number of clusters as reference avoids picking a trivial partition, and aligning all other runs to it keeps the procedure linear in the number of vertices. Using enough runs, this procedure finds all pairs of vertices that can possibly end up in two different parts when applying CW. In this way, the outcome of CW can be made almost deterministic. The number of standard CW runs does not have to be fixed beforehand, but can be iteratively increased until no changes are observed in the result. The convergence test pins down to counting the number of different classes in the combined result, which increases monotonically until the maximum possible number is reached. As this operation involves n (number of vertices) operations, it does not increase the computational complexity of the overall process, yet larger graphs require a larger number of runs. Using this stopping criterion, however, does not give a guarantee that the maximal possible number of clusters is found, as the currently last run of standard CW does not increase the number of clusters if equivalent in result to a previous run. When taking several recent runs without changes into account,

class	A	B	C	D	E	F	G	H
A	6	0	0	3	3	3	3	0
B	0	3	0	3	0	0	0	3
C	0	0	3	3	0	3	0	3

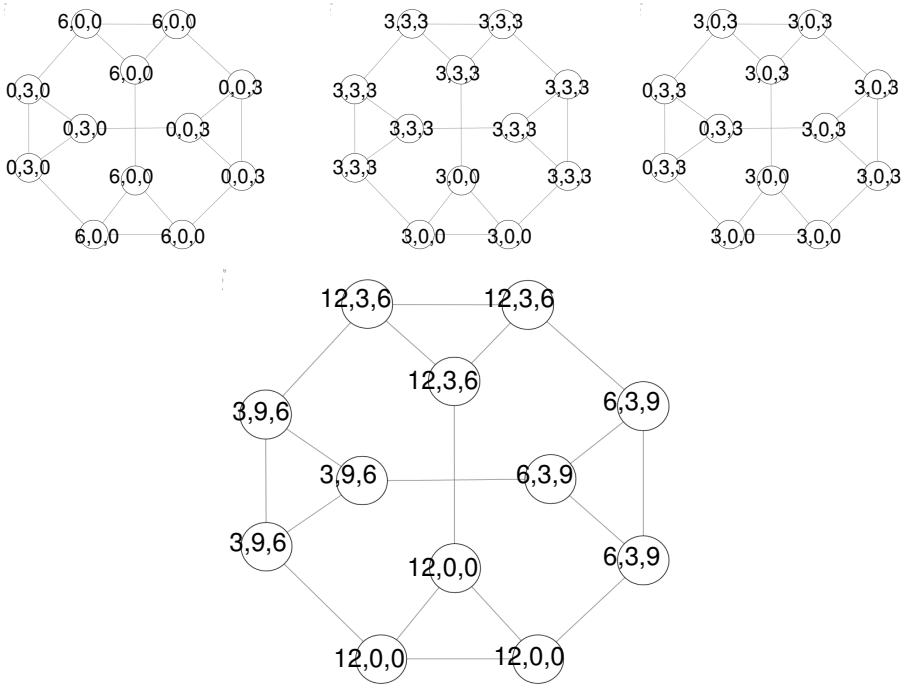


Figure 4.15: Reference class vectors obtained from comparing run 1 with run 2 and run 1 with run 3, then result, which is equivalent to the plausible partition in Figure 4.13. Notice that no class occurs in more than one partition

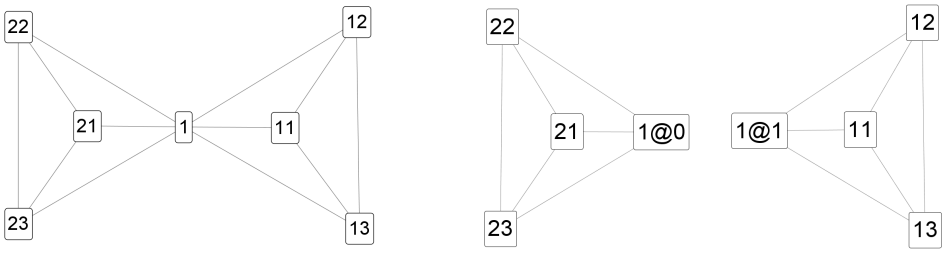


Figure 4.16: Left: a graph. Right: its disambiguated graph

however, the possibility of premature convergence can be lowered.

4.2.5 Disambiguation of Vertices

Now, a graph transformation is described that performs the conversion of the original graph into a graph with disambiguated vertices. The transformation is motivated by the fact that in natural language, ambiguity, i.e. the possibility to use the same symbol for several denotations, is omnipresent. The most widely used example for lexical ambiguity is "bank", which can be used in the sense of a financial institution or refer to a river bank. But also other levels of ambiguity, e.g. in compound noun splitting, part-of-speech, phoneme-to-text conversion and phrase attachment need to be resolved. In the graph framework, a vertex is said to be ambiguous, if it belongs to more than one group, therefore acting in different roles and having edges to vertices of all groups it is related to. This property of vertices is purely local and constitutes itself in the neighbourhood of the vertex, not being affected by other regions of the graph. Following the idea of Widdows and Dorow (2002), the different roles or usages of a vertex v can be obtained by partitioning the subgraph induced by the neighbourhood of vertex v, $G(neigh(v))$. As $v \notin neigh(v)$, the subgraph may be formed of several components already. Partitioning $neigh(v)$ with CW results in $m \geq 1$ parts of $G(neigh(v))$. For the transformation, vertex v is replaced by m vertices $v@0, v@1, ...v@(m-1)$, and $v@i$ is connected to all vertices in cluster i, retaining edge weights. This is done iteratively for all vertices, and the results are combined in the disambiguated graph. Figure 4.16 shows an unweighted graph and its disambiguated graph. Note that all neighbourhoods of vertices apart from vertex 1 yield one cluster, whereas $neigh(1)$ is split into two clusters.

Whereas a CW-partition of the original graph can result in one cluster, the disambiguated version is always split into two clus-

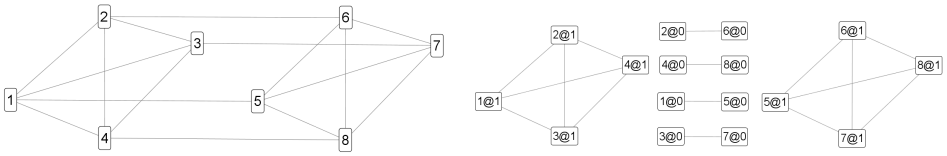


Figure 4.17: Left: the 4-bipartite clique. Right: the disambiguated 4-bipartite clique

ters. Applying the disambiguation transformation to the n-bipartite cliques above results in splitting the graph in $n + 2$ components, of which n reflect the connecting edges between the cliques, and the remaining two are formed by the cliques, see Figure 4.17. The small components reflecting the connecting edges can be pruned by not allowing singletons in the neighbourhood graphs.

4.2.6 Hierarchical Divisive Chinese Whispers

In Section 4.1.1, the differences between a hierarchical clustering and a partition were laid out. CW in its presently discussed form is a partitioning algorithm that returns un-nested groups of vertices. To turn CW into a hierarchical cluster algorithm, one can split a cluster iteratively into smaller parts in a top-down way. Since a CW cluster is a set of vertices that receives a uniform class labelling and CW operates locally, a CW clustering of the subgraph induced by the cluster will again result in a uniform labelling. To be able to still split this cluster, the subgraph has to be modified in a way that CW will be able to find more subclusters. A way of doing this for weighted graphs is to apply a threshold on edge weights and to delete edges that have a weight below that threshold. This can result in singleton vertices, which will not be contained in any of the subclusters, but remain assigned to the upper level cluster.

Especially for scale-free Small World graphs, hierarchical divisive CW provides a means to avoid one very large cluster. To illustrate this, the following experiment was carried out on the sentence-based significant word-cooccurrence graph of LCC's 1 million sentence German corpus (as used in Section 3.2.2): clustering this graph of 245,990 vertices with standard CW, the largest cluster consisted of 219,655 vertices. Only the largest cluster was selected for splitting: the edge weight threshold was set in a way that half of the vertices remained in the subgraph, the other half was left aside. In the first division step, clustering the subgraph of 109,827 vertices yielded a largest cluster of 44,814 vertices, of which again half of the vertices

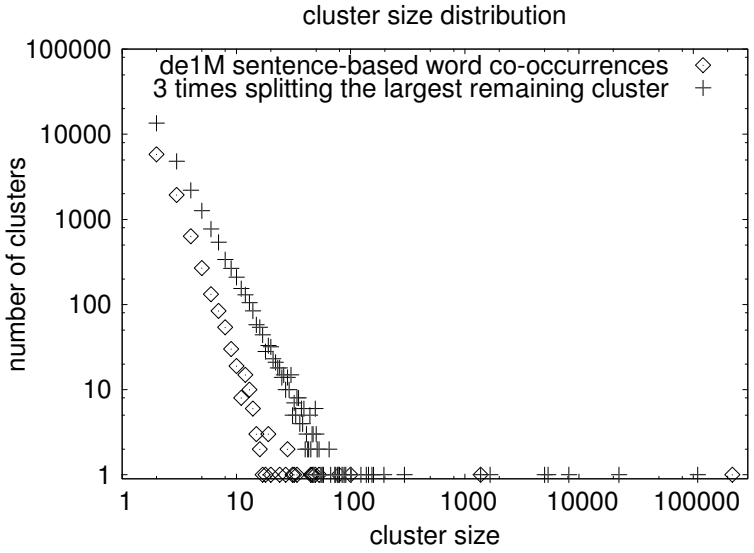


Figure 4.18: Cluster size distribution for standard CW and three times applying a hierarchical divisive clustering step on the largest cluster

were selected for the second step. The largest cluster of this subgraph's partition (22,407 vertices) contained 16,312 vertices. In a similar third step, a subgraph of 8,157 vertices produced a largest cluster of 5,367 vertices.

Figure 4.2.6 shows the cluster size distribution of the standard clustering and after the third divisive step. For the size distribution, the hierarchy of clusters is ignored. The quantitative analysis shows that the vertices of the very large initial cluster are separated in many small clusters after three times splitting the largest remaining cluster. The number of clusters increased from 9,071 in the initial clustering to 24,880 after the third step.

The decisions about which clusters to split and how many vertices to involve in the subcluster clustering render hierarchical divisive CW difficult to apply in a conclusive and motivated way. Easier to handle is the bottom-up variant of hierarchical clustering as presented in the next section.

4.2.7 Hierarchical Agglomerative Chinese Whispers

Section 4.2.4 described how the combination of several runs of CW runs results in a maximally fine-grained partition with respect to the iterative CW process. In applications, this partition might be too

fine-grained, as having more clusters means less vertices per cluster, which might cause data sparseness problems in the application. As an example, consider lexical ambiguity: whereas CW, as other clustering algorithms, might find so-called micro-senses or facets (cf. Cruse, 2002), these fine-grained distinctions between e.g. *school* as a building or as an institution might not be useful in a word sense disambiguation task, or even harmful as there might not be enough clues for either micro-sense, yet the combination of their clues could suffice to distinguish them from unrelated usages.

Now a possibility is described how clusters can be arranged hierarchically in an agglomerative fashion. A partition of a graph can be viewed as dividing the graph's edges into inter-cluster and intra-cluster edges, the former connecting vertices in the same cluster whereas the latter being edges between clusters. Intuitively, two clusters have more in common, or are more similar to each other, if there is a large amount of inter-cluster edges connecting their vertices. In hierarchical agglomerative CW, hypervertices are formed by subsuming all vertices of one cluster under one hypervertex. the hyperedges connecting the hypervertices are weighted, their weight reflects the connection strength between the respective clusters. For example, a hyperedge weighting function can be given as

$$hyperweight(P_i, P_j) = \frac{|\{\{u, w\}|u \in P_i, w \in P_j\}|}{min(|P_i|, |P_j|)}$$

where P_i is the set of vertices constituting cluster i. This hyperweight function measures the amount of vertices in the smaller cluster that are endpoints of an edge between the two clusters. The resulting hypergraph – hypervertices and hyperedges – is again partitioned with CW, resulting in a set of hyperclusters. In this way, nested clusters are obtained by subordinating the clusters of the graph under the hyperclusters in the hypergraph. Iterative application results in a hierarchical clustering, with as many top-clusters as there are components in the graph. Figure 4.19 shows the hierarchical clustering of a graph in three steps.

As compared to other hierarchical clustering methods that mostly produce binary tree hierarchies, hierarchical agglomerative CW results in flat hierarchies with arbitrary branching factors. Apart from the instantiation of the *hyperweight* function, the hierarchical agglomerative version is parameter-free, and all modifications of CW can be employed for it.

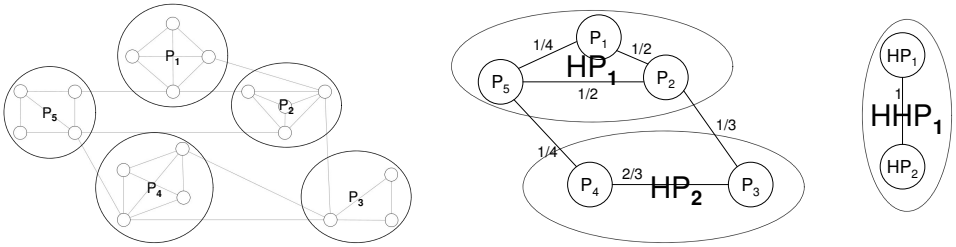


Figure 4.19: Hierarchical agglomerative CW in three steps on an unweighted graph with 19 vertices. First step: five clusters P_1, P_2, P_3, P_4, P_5 by deterministic CW. Second step: two hyperclusters HP_1, HP_2. Third step: one large hyper-hypercluster HHP_1

4.2.8 Summary on Chinese Whispers

With its run-time linear in the number of edges, Chinese Whispers belongs to the class of graph partitioning algorithms at the lower bound of computational complexity: at least, the graph itself has to be taken into account when attempting to partition it, and the list of edges is the most compact form of its representation. The randomised nature of CW make it possible to run the algorithm in a decentralised and distributed way: since there are no constraints on the processing order of vertices and the algorithm uses only local information when updating the class of a vertex, a parallelisation is straightforward. This enables the partitioning of even larger graphs than the ones discussed here.

The property of standard CW to be parameter-free is highly desired for applications, as brittle parameter tuning can be avoided. Nevertheless, for datasets that demand a different treatment than standard CW provides, various extensions that try to leverage the non-deterministic nature and the absence of hierarchically nested clusters were laid out.

Since theoretical results are hard to obtain for randomised and non-deterministic algorithms, CW was tested on a variety of artificial graphs, showing great capabilities of producing meaningful clusters. Exploring the practical utility of CW will be undertaken in the next chapter.

An implementation of Chinese Whispers is available for download[5].

[5] `http://wortschatz.uni-leipzig.de/~cbiemann/software/CW.html` [June 1st, 2006]

5 Structure Discovery Procedures

This chapter discusses the application of the graph clustering algorithm as introduced in Chapter 4 to unsupervised natural language learning, following the Structure Discovery paradigm. With the implementation of every step that recognises and uses structural properties of natural language, the vision of natural language understanding by machines as outlined in Section 1.4 becomes more feasible. Nevertheless, the various methods and algorithms described here merely exemplify possibilities of unsupervised and knowledge-free natural language learning – the compilation of a more comprehensive system goes beyond the scope of a single dissertation and is left for future work.

Referring back to Section 1.3, methods that find homogeneous word sets on various levels of homogeneity are laid out in theory and practice. Section 5.1 is devoted to language separation: monolingual chunks of a mixed-language corpus shall be identified, resulting in a language identifier. It contains a modified and extended version of (Biemann and Teresniak, 2005). Section 5.2 constitutes the main practical contribution of this work: the implementation and evaluation of an unsupervised part-of-speech tagger that assigns syntactic-semantic categories to all words in a monolingual corpus. The system was previously described in (Biemann, 2006c). This tagging method is also evaluated in an application-based way as a component in various systems, published as (Biemann *et al.*, 2007).

Another procedure tackling the problem of word senses, which was only set up prototypically and rather indicates future directions than a components in their final form, is discussed in Section 5.3.

5.1 Unsupervised Language Separation

This section presents an unsupervised solution to language identification. The method sorts multilingual text corpora sentence-wise into different languages. Here, no assumptions about number or size of the monolingual fractions are made.

With a growing need for text corpora of various languages in mind, the question of how to build monolingual corpora from multilingual

sources is addressed. Admittedly, language identification is a solved problem, as reflected in the title of (McNamee, 2005), where it is rendered as a suitable task for undergraduate exercises. In this attempt, the main difference to previous methods is that no training data for the different languages is provided and the number of languages does not have to be known beforehand. This application illustrates the benefits of a parameter-free graph clustering algorithm like Chinese Whispers, as the data – words and their statistical dependencies – are represented naturally in a graph, and the number of clusters (here: languages) as well as their size distribution is unknown.

5.1.1 Related Work

According to Banko and Brill (2001), Pantel *et al.* (2004) and others, shallow methods of text processing can yield comparable results to deep methods when allowing them to operate on large corpora. The larger the corpus, however, the more difficult it is to ensure sufficient corpus quality. Most approaches in computational linguistics work on monolingual resources or on multilingual resources with monolingual parts, and will be disturbed or even fail if a considerable amount of 'dirt' (sublanguages or different languages) are contained. Viewing the World Wide Web as the world's largest text corpus, it is difficult to extract monolingual parts of it, even when restricting crawling to country domains or selected servers.

While some languages can be identified easily due to their unique encoding ranges in ASCII or UNICODE (like Greek, Thai, Korean, Japanese and Chinese), the main difficulty arises in the discrimination of languages that use the same encoding and some common words, as most of the European languages do. In the past, a variety of tools have been developed to classify text with respect to its language. The most popular free system, the TextCat Language Guesser[1] by Cavnar and Trenkle (1994), makes use of the language-specific letter N-gram distribution and can determine 69 different natural languages. According to Dunning (1994), letter trigrams can identify the language almost perfectly from a text-length of 500 bytes on. Other language identification approaches use short words and high-frequency words as features, e.g. (Johnson, 1993), or combine both approaches (cf. Schulze, 2000). For a comparison, see (Grefenstette, 1995).

All of these approaches work in a supervised way: given a sample of each language, the model parameters are estimated and texts are

[1] `http://odur.let.rug.nl/~vannoord/TextCat/Demo/` [December 1st, 2006]

classified according to their similarity to the training texts. But supervised training has a major drawback: the language identifier will fail on languages that are not contained in the training and, even worse, it will mostly have no clue about that and assign some arbitrary language: TextCat assigns 'Nepali' to texts like "xx xxx x xxx ..." and 'Persian' to "öö ö öö öööö ...".

The method described here operates on words as features and finds the number of languages in a fully unsupervised way. Further, it is able to decide for every single sentence, to which language it belongs. Of course, it is not able to label text with the names of the involved languages, but rather groups sentences of the same language together.

5.1.2 Method

Employing word co-occurrence statistics (cf. Section 3.2.1), weighted graphs built from words as vertices and their associations as edges are constructed. Assuming monolinguality of sentences, there are more word pairs of the same language exhibiting significant co-occurrence than word pairs of different languages, Chinese Whispers will find one cluster for each language. The words in the clusters serve as features to identify the languages of the text collection by using a word-based language identifier.

As input, a multilingual, sentence-separated plain text corpus is assumed. The accuracy of the sentence separation, however, does not play an important role here and can be realised by splitting sentences on sentence delimiters like ".!?" or merely using paragraph boundaries. What matters is that the corpus is split into parts roughly corresponding to sentences in order to define the significant sentence-based co-occurrence graph as described in Section 3.2.1. Alternatively, window-based word co-occurrences could be employed.

A multilingual word co-occurrence graph is constructed by adding all word pairs that occur together at least twice with a significance of at least 1.64 (20% error; as preliminary experiments showed, however, the significance level can be set to any value in reasonable ranges without influencing results much), adding the words as vertices and weighting the edges by the significance value.

This multilingual word co-occurrence graph is clustered with standard Chinese Whispers, using 20 iterations. All clusters that exceed 1% of the size of the whole graph are used to define languages: the assumption is that words of the same language are found in the same cluster. Words that are contained in two or more languages will be

assigned to only one cluster, languages are labelled by their cluster ID. The size threshold of 1% is arbitrary but showed to be a suitable setting to exclude noise in preliminary experiments.

These language clusters are used in the simple word-based language identifier described in (Teresniak, 2005) to assign cluster IDs to sentences: counting, how many words of the sentence are found in which cluster, the sentence is labelled with the ID of the cluster that contained most of the words. For labelling, at least two words and at least 10% of the words in the sentence have to be found in a cluster, otherwise the sentence is marked as 'unknown' – this was found to be a sensible setup to exclude questionable cases.

The algorithm for language separation that assigns a language ID $l(s)$ to every sentence s of corpus C is given in Algorithm 7:

Algorithm 7 langSepP(corpus C):

G = significant sentence-based word co-occurrence graph of C
partition P = CW(G)
for all sentences $s = \{w_1, w_2, ...w_n\}$ in C **do**
  language L = arg max$_{P_i}$ $|s \cap P_i|$
  if $(|L \cap s| < 2)$ OR $(|L \cap s| \cdot 10 < |s|)$ **then**
    $l(s)$=unknown
  else
    $l(s) = L$
  end if
end for

5.1.3 Evaluation

To test performance of the language separator, monolingual corpora of different languages are combined into a multilingual corpus. The success of the language separator is measured by the extent to which the original monolingual parts can be restored. For this, standard measures as used in information retrieval are employed:

- *Precision* (P) is the number of true positives divided by the sum of true positives and false positives,
- *Recall* (R) is the number of true positives divided by the total number of target items.

As it cannot be assumed beforehand that the number and extension of monolingual parts equals the number of language clusters, the following mapping between target languages and language clusters

	N	1	2	3	4	5	map	P	R
A	100	75	0	0	0	0	1	1	0.75
B	200	5	110	40	1	20	2	0.625	0.55
C	300	5	120	60	30	10	4	0.133	0.1
D	400	15	0	280	25	0	3	0.875	0.7

Table 5.1: Exemplary mapping of target languages and cluster IDs. N = number of sentences in target language, the matrix $\{A, B, C, D\} \times \{1, 2, 3, 4, 5\}$ contains the number of sentences of language A..D labelled with cluster ID 1..5. Column 'map' indicates which cluster ID is mapped to which target language, columns P and R provide precision and recall for the single target languages, overall precision: 0.6219, overall recall: 0.495

is carried out: a target language is mapped to the cluster ID that is found predominantly in the monolingual part of the target language. In case a cluster ID is found predominantly in several target languages, it is only mapped to the one target language where it constitutes the largest fraction, the other target languages get the next predominant cluster ID until a free cluster ID is found.

Table 5.1 shows an example of the mapping for a hypothetical dataset and exemplifies overall precision and recall as well as for single target languages. Here, language C gets assigned ID 4 from the clustering although the overlap between language C and cluster ID 2 is higher. The reason is that ID 2 is already mapped to language B, which has a higher percentual overlap to cluster ID 2 than language C. Notice that cluster ID 5 is not mapped to any language, which effects overall recall negatively.

5.1.4 Experiments with Equisized Parts for 10 Languages

In (Biemann and Teresniak, 2005), experiments on mixing equisized parts to 7-lingual corpora are described. Here, the difficulty of the task is increased slightly in two ways: for experiments with equisized parts, 10 languages are used, and the sentences are not drawn subsequently from documents, but randomly from a large corpus. By randomly selecting sentences, the effect of word burstiness (cf. Church, 2000) is ruled out, which leads to fewer significant co-occurrences for small monolingual chunks: sentences from the same document tend to be centred around the document's topic which is reflected in their wording, producing significant co-occurrences of these topic-specific words. These should not be observed to that

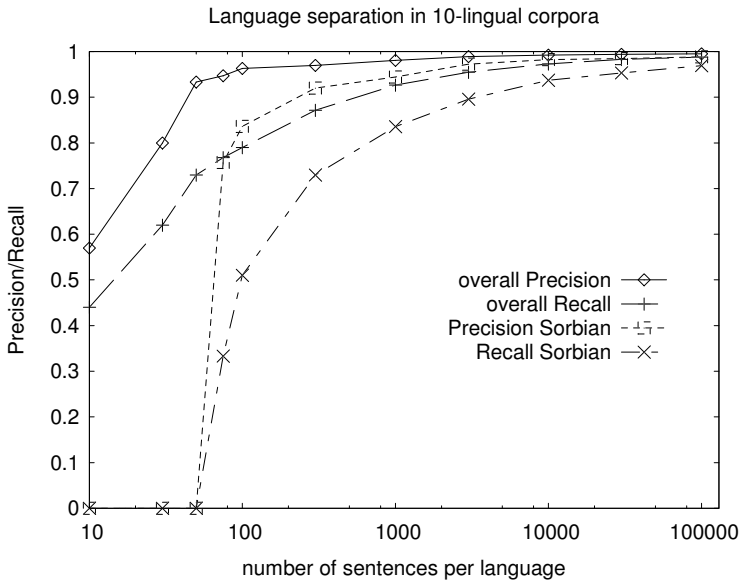


Figure 5.1: Language separation in 10-lingual corpora with equisized monolingual parts for varying numbers of sentences

extent when selecting sentences randomly from documents encompassing a wide range of topics, making the task more difficult.

LCC was used as a data source to compile multilingual corpora consisting of Dutch, English, Finnish, French, German, Icelandic, Italian, Japanese, Norwegian and Sorbian up to 100,000 sentences per language.

Sorbian is a special case, as it is the only language that is not an official language of a country, but spoken by a Slavonic minority in Eastern Germany, why Sorbian texts contain words and even sentences in German. Considering this, it is not surprising that performance on Sorbian was lowest among all languages involved. Figure 5.1 shows overall results and results for Sorbian.

From 100 sentences per language on, the method is able to find exactly 10 clusters, and map them to the 10 languages involved. For fewer sentences, the number of clusters was smaller (7 for 10 sentences per language, 9 for 30 sentences per language). The easiest language to recognise in this experiment is Japanese, which shows a precision of 1 in all experiments and recall values from 0.9987 to 1. This can be attributed to its distinct character set – the second easiest language is English, scoring more than 0.9997 on precision and more than 0.997 on recall for all experiments from 1,000 sentences

cluster ID	1	2	3	4	5	6	7	8	9	10
French	99709	13			3	3			1	
English	19	99950			3	1	2			
Icelandic	26	118	99060		17	13	30	17	2	
Japanese				99993						
Italian	29	34	1	1	99451	2			1	3
Norwegian	56	158	668	4	8	97165	27	4	36	4
German	21	117	1	1	6		99235		7	2
Finnish	14	21	5	1	4	60	3	99616	2	2
Dutch	611	999	2	1	30	6	273	4	96793	5
Sorbian	36	138	3	2	13	7	920	14	4	96909

Table 5.2: Confusion matrix of clusters and languages for the experiment with 100,000 sentences per language

per language on. In the Sorbian case, a manual inspection revealed that about 1% of sentences in this data are either proper German or German dialect, sometimes mixed with Sorbian.

Most misclassifications are caused by embedded foreign-language elements, such as the sentence *"INA besitzt Anteile an der Banca Nazionale di Lavoro und am Banco di Napoli."* in the German part that is misclassified as Italian because it contains names of two Italian banks. Languages of the same language family are more difficult to separate, e.g. Norwegian-Icelandic or Dutch-German. Most confusion is observable for English, which is the largest contributor of foreign-language material in all other languages. Table 5.2 shows the confusion matrix for the experiment with 100,000 sentences per language.

Recall is dependent on sentence length: in short sentences, it is less likely to find enough words from the clusters. This is depicted in Figure 5.2: sentences longer than 10 words almost surely get classified.

Most long sentences marked as 'unknown' contained names, dates or address parts, such as e.g.

- *Ruth Klink, Niddatal 1, 100,- ; Margot Klöck, 50,- ; Frank Klöß, Bad Vilbel, 20,-.* (German)
- *BC-Net Secretariat , European Commission , DG23 , Rue de la Loi 200 , B-1049 Brussels .* (English)
- *Aspelinin veli) 22 - Aspelin Emil (JRA:n serkku) 10 - Aspelin Henrik Emanuel (Manne , JRA:n veli) 1 Hr19 J.R.* (Finnish)

These kind of sentences are not informative regarding language structure and it might be a good idea to remove them during corpus compilation anyway.

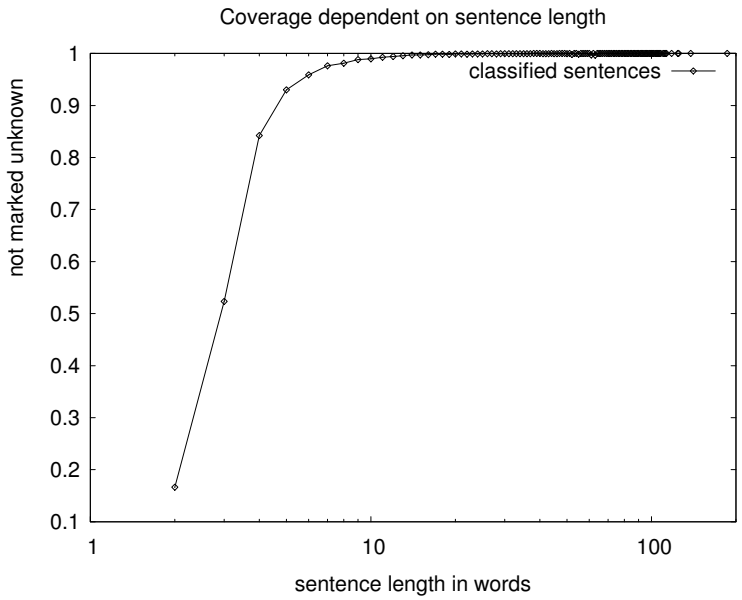


Figure 5.2: Coverage dependent on sentence length

Unsupervised language separation proves capable of separating equisized monolingual chunks from each other. Performance is almost perfect starting from a few hundred sentences per language. The next section is devoted to testing the stability of the method when applying it to bilingual mixtures with monolingual parts of different sizes.

5.1.5 Experiments with Bilingual Corpora

While the experiments in the previous section focussed on equal fractions of many languages, the method is now evaluated on bilingual corpora with monolingual fractions of differing sizes. This setting is somewhat more realistic when identifying languages in web data, for a top-level domain usually provides most of its content in one language, and the task is to remove substantially smaller parts of foreign language material. The experimental setup is as follows: into a monolingual corpus of 100,000 sentences, chunks of 10, 30, 100, 300, 1,000, 3,000, 10,000 and 30,000 sentences of another language are injected. The major language is always identified; therefore performance is only reported on the injected language.

In the previous section it was observed that languages of the same language family are harder to separate than very different languages

or even languages using different alphabets. Therefore, two extreme cases are examined here: mixtures of French as major and Japanese as minor language, and mixtures of Swedish as major and Norwegian as a minor language. Separation of other language pairs can be expected to perform in between these borderline cases.

French and Japanese do not have any words in common apart from names; therefore the multilingual word graph is easily separable. The only crucial parameter is the cluster size threshold: setting it to 1% as in the previous section, the Japanese part is recognised perfectly from 300 sentences on. Lowering the threshold to 0.03%, it is even possible to find every single of the 10 sentences of Japanese in a 100,000 sentence corpus of French without a drop on precision and recall in the French part.

In (Biemann and Teresniak, 2005), experiments on injecting English in an Estonian corpus and Dutch in a German corpus succeeded in almost perfect separation from 500 minor language sentences on. It was only slightly more difficult to find the French injection into an Italian corpus due to the relatedness of these languages: the more words languages have in common, the more difficult it is to separate them. In (Quasthoff and Biemann, 2006), the overlap between the most frequent 1,000 words of several European languages is given. Danish and Norwegian are the most similar languages with respect to this measure, as they share almost half of their top 1,000 words. Experiments aiming at separating Danish and Norwegian mixtures with the method described here did not succeed. The next most similar language pair is Swedish and Norwegian with sharing over 1/4th of these words. As the results in Figure 5.3 show, this results in lower performance on this pair as compared to the multilingual experiments.

While performance on 100,000 sentences of each Norwegian and Swedish is comparable to what has been reported in the equisized experiments, smaller fractions of Norwegian are difficult to identify and get often labelled with the same ID as the larger Swedish part.

5.1.6 Summary on Language Separation

Apart from the experiments with Scandinavian languages, which aimed at testing the method for very extreme cases, it is possible to conclude that unsupervised language separation arrives at performance levels that are comparable to its supervised counterparts. The problem of language separation and identification can be regarded as solved not only from a supervised, but also from a Structure Dis-

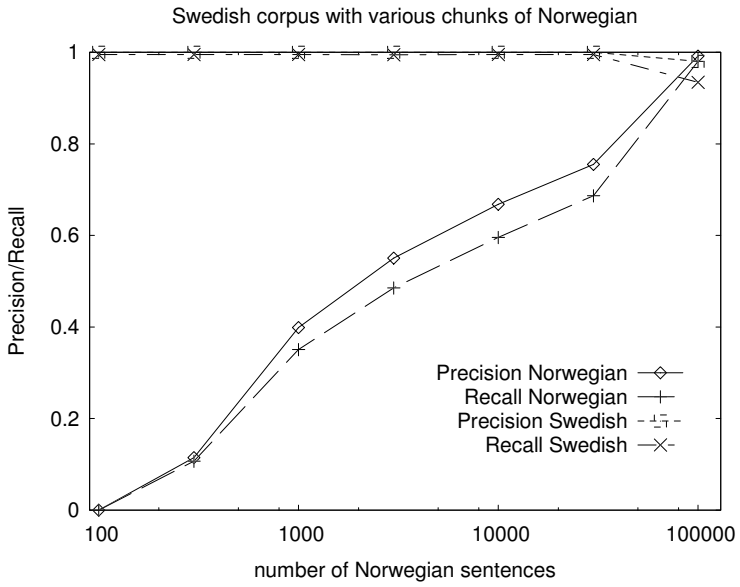


Figure 5.3: Language separation performance for Norwegian injected in a Swedish corpus of 100,000 sentences

covery perspective.

To collect the few unclassified sentences, the approach could be extended to the document level: unknown sentences are assigned the language that surrounds them. Using the extensions of CW as discussed in Sections 4.2.4 and 4.2.5 does not seem necessary in the light of the obtained performance levels.

An implementation of this method called "langSepP" (language separation program) is available for download[2].

5.2 Unsupervised Part-of-Speech Tagging

The previous section presented a method that produced homogeneity in word sets with respect to different languages. Now, homogeneity with respect to parts-of-speech (POS) is aimed at. The method presented in this section is called unsupervised POS-tagging, as its application results in corpus annotation in a comparable way to what POS-taggers provide. Nevertheless, its application results in slightly different categories as opposed to what is assumed by a linguistically motivated POS-tagger. These differences hamper evaluation

[2] `http://wortschatz.uni-leipzig.de/~cbiemann/software/langSepP.html` [July 7th, 2007]

procedures that compare the output of the unsupervised POS-tagger to a tagging with a supervised tagger, and will be illustrated below. To measure the extent to which unsupervised POS tagging can contribute in application-based settings, the system is evaluated in supervised POS tagging, word sense disambiguation, named entity recognition and chunking.

Unsupervised POS-tagging has been explored since the beginning of the 1990s. Unlike in previous approaches, the kind and number of different tags is here generated by the method itself. Another difference to other methods is that not all words above a certain frequency rank get assigned a tag, but the method is allowed to exclude words from the clustering, if their distribution does not match closely enough with other words. The lexicon size is considerably larger than in previous approaches.

The system description is an extended version of (Biemann, 2006c), the application-based evaluation was previously described in (Biemann *et al.*, 2007).

5.2.1 Introduction to Unsupervised POS Tagging

Assigning syntactic categories to words is an important preprocessing step for most NLP applications. POS tags are used for parsing, chunking, anaphora resolution, named entity recognition and information extraction, just to name a few.

Essentially, two things are needed to construct a tagger: a lexicon that contains tags for words and a mechanism to assign tags to tokens in a text. For some words, the tags depend on their use, e.g. in "I saw the man with a saw". It is also necessary to handle previously unseen words. Lexical resources have to offer the possible tags, and a mechanism has to choose the appropriate tag based on the context, in order to produce annotation like this: "I/PNP saw/VVD the/AT0 man/NN1 with/PRP a/AT0 saw/NN1 ./PUN"[3].

Given a sufficient amount of manually tagged text, two approaches have demonstrated the ability to learn the instance of a tagging mechanism from labelled data and apply it successfully to unseen data. The first is the rule-based approach (Brill, 1992), where transformation rules are constructed that operate on a tag sequence delivered by the lexicon. The second approach is statistical, for example HMM-taggers (Charniak *et al.*, 1993, inter al.) or taggers based

[3] in this tagset (Garside *et al.*, 1987), PNP stands for personal pronoun, VVD is full verb, AT0 is determiner is singular or plural, NN1 is singular noun, PRP is Preposition, PUN is punctuation

on conditional random fields (see Lafferty *et al.*, 2001). Both approaches employ supervised learning and therefore need manually tagged training data. Those high-quality resources are typically unavailable for many languages and their creation is labour-intensive. Even for languages with rich resources like English, tagger performance breaks down on noisy input. Texts of a different genre than the training material may also create problems, e.g. e-mails as opposed to newswire or literature. It is, in general, not viable to annotate texts for all these cases.

Here, an alternative needing much less human intervention is described. Steps are undertaken to derive a lexicon of syntactic categories from unstructured text following the Structure Discovery paradigm. Hence, it is not possible to aim at exact correspondence with linguistically motivated tagsets, but for obvious reasons: even for the same language, linguistically motivated tagsets differ considerably, as it was measured for various tagsets for English by Clark (2003).

Two different techniques are employed here, one for high-and medium frequency words, another for medium- and low frequency words. The categories will be used for the tagging of the same text the categories were derived from. In this way, domain- or language-specific categories are automatically discovered. Extracting syntactic categories for text processing from the texts to be processed fits the obtained structures neatly and directly to them, which is not possible using general-purpose resources.

5.2.2 Related Work

There are a number of approaches to derive syntactic categories. All of them employ a syntactic version of Harris' distributional hypothesis (Harris, 1968): words of similar parts of speech can be observed in the same syntactic contexts. Measuring to what extent two words appear in similar contexts measures their similarity (cf. Miller and Charles, 1991). As function words form the syntactic skeleton of a language and almost exclusively contribute to the most frequent words in a corpus, contexts in that sense are often restricted to the most frequent words. The words used to describe syntactic contexts are further called *feature words*. *Target words*, as opposed to this, are the words that are to be grouped into syntactic clusters. Note that usually, the feature words form a subset of the target words.

The general methodology (Finch and Chater, 1992; Schütze, 1993, 1995; Gauch and Futrelle, 1994; Clark, 2000; Rapp, 2005) for inducing

```
... COMMA sagte der Sprecher bei der Sitzung FULLSTOP
... COMMA rief der Vorsitzende in der Sitzung FULLSTOP
... COMMA warf in die Tasche aus der Ecke FULLSTOP
```

Features: der(1), die(2), bei(3), in(4), FULLSTOP (5), COMMA (6)

position	-2				-1						+1						+2	
target/feature	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2
sagte										1	1							
rief										1	1							
warf										1				1				1
Sprecher					1								1				1	
Vorsitzende					1									1			1	
Tasche		1				1											1	
Sitzung	1	1			2										2			
Ecke					1										1			

Figure 5.4: Corpus and context vectors for 6 feature words and a context window of size 4. The feature vectors of different positions are concatenated

word class information can be outlined as follows:

1. Collect global context vectors of target words by counting how often feature words appear in neighbouring positions.
2. Apply a clustering algorithm on these vectors to obtain word classes

Throughout, feature words are the 150-250 words with the highest frequency. Some authors employ a much larger number of features and reduce the dimensions of the resulting matrix using Singular Value Decomposition (Schütze, 1993; Rapp, 2005). The choice of high frequency words as features is motivated by Zipf's law: these few stop words constitute the bulk of tokens in a corpus. Pruning context features to these allows efficient implementations without considerably losing on coverage. Contexts are the feature words appearing in the immediate neighbourhood of a word. The word's global context is the sum of all its contexts. Figure 5.4 illustrates the collection of contexts for a German toy example.

As outlined in Chapter 4, clustering consists of a similarity measure and a clustering algorithm. Finch and Chater (1992) use the Spearman Rank Correlation Coefficient and a hierarchical clustering, Schütze (1993, 1995) uses the cosine between vector angles and Buckshot clustering, Gauch and Futrelle (1994) use cosine on Mutual Information vectors for hierarchical agglomerative clustering

and Clark (2000) applies Kullback-Leibler divergence in his CDC algorithm.

An extension to this generic scheme is presented in (Clark, 2003), where morphological information is used for determining the word class of rare words. Freitag (2004a) does not sum up the contexts of each word in a context vector, but uses the most frequent instances of four-word windows in a co-clustering algorithm (Dhillon *et al.*, 2003): rows and columns (here words and contexts) are clustered simultaneously. Two-step clustering is undertaken by Schütze (1993)): clusters from the first step are used as features in the second step.

The number of target words in the clustering differ from 1,000 target words in a 200,000 token corpus (Gauch and Futrelle, 1994) over 5,000 target words (Finch and Chater, 1992; Freitag, 2004a) to all 47,025 words in the Brown Corpus in (Schütze, 1995). Clark (2000) uses 12 million tokens as input; Finch and Chater (1992) operate on 40 million tokens.

Evaluation methodologies differ considerably amongst the papers discussed here. Finch and Chater (1992) inspect their clusters manually, Rapp (2005) performs flawlessly in sorting 50 medium frequency words into nouns, verbs and adjectives. Schütze (1995) presents precision and recall figures for a reduced tagset, excluding rare and non-English-word tags from the evaluation. More recent approaches (Clark, 2000, 2003; Freitag, 2004a) employ information-theoretic measures, see Section 5.2.8. Regarding syntactic ambiguity, most approaches do not deal with this issue while clustering, but try to resolve ambiguities at the later tagging stage.

As the virtue of unsupervised POS tagging lies in its possible application to all natural languages or domain-specific subsets, it is surprising that in most previous works, only experiments with English are reported. An exception is (Clark, 2003), who additionally uses languages of the Slavonic, Finno-Ugric and Romance families.

A severe problem with most clustering algorithms is that they are parameterised by the number of clusters. As there are as many different word class schemes as tagsets, and the exact amount of word classes is not agreed upon intra- and interlingually, having to specify the number of desired clusters a-priori is clearly a drawback. In that way, the clustering algorithm is forced to split coherent clusters or to join incompatible sub-clusters. In contrast, unsupervised part-of-speech induction means the induction of the tagset, which implies finding the number of classes in an unguided way.

Another alternative which operates on a predefined tagset is presented by Haghighi and Klein (2006): in this semi-supervised frame-

work, only three words per tag have to be provided to induce a POS-tagger for English with 80% accuracy. The amount of data the authors use in their experiments is rather small (8,000 sentences), but their computationally expensive methods – gradient-based search to optimise Markov Random Field parameters – does not allow for substantially more input data.

5.2.3 Method

What follows is the description of the construction of the unsupervised POS-tagger from scratch. Input to the system is a considerable amount of unlabelled, tokenised monolingual text without any POS information. In a first stage, Chinese Whispers is applied to distributional similarity data, which groups a subset of the most frequent 10,000 words of a corpus into several hundred clusters (tagset 1). Second, similarity scores on neighbouring co-occurrence profiles are used to obtain again several hundred clusters of medium- and low frequency words (tagset 2). The combination of both partitions yields sets of word forms belonging to the same induced syntactic category. To gain on text coverage, ambiguous high-frequency words that were discarded for tagset 1 are added to the lexicon. Finally, a Viterbi trigram tagger is trained with this lexicon and augmented with an affix classifier for unknown words.

Figure 5.5 depicts the process of unsupervised POS-tagging from unlabelled to fully labelled text. The details of the method will be outlined in the following sections.

The method employed here follows the coarse methodology as described in the previous subsection, but differs from other works in several respects. Although four-word context windows and the top frequency words as features are used (as in Schütze, 1995), the cosine similarity values between the vectors of target words are transformed into a graph representation in order to be able to cluster them with CW. Additionally, a method to identify and incorporate POS-ambiguous words as well as low-frequency words into the lexicon is provided.

5.2.4 Tagset 1: High and Medium Frequency Words

Four steps are executed in order to obtain tagset 1 for high- and medium frequency words from a text corpus.

1. Determine 10,000 target and 200 feature words from frequency counts

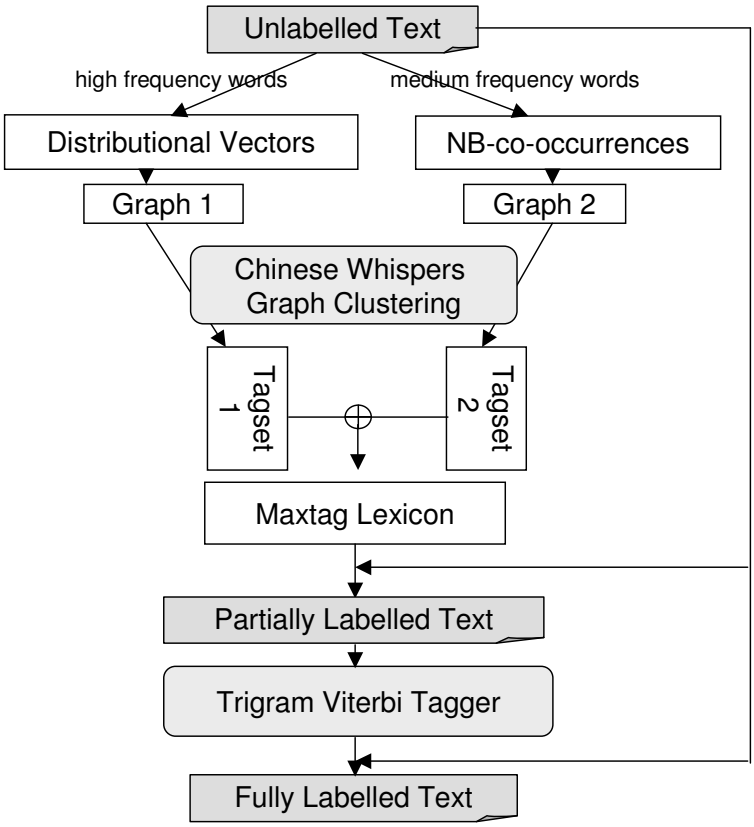


Figure 5.5: Diagram of the process of unsupervised POS tagging, from unlabelled over partially labelled to fully labelled text

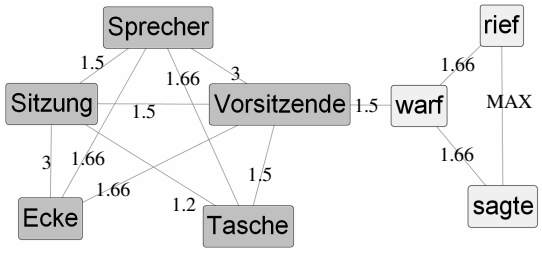


Figure 5.6: Graph for the data given in Figure 5.4 and its partition into nouns and verbs

2. Collect context statistics and construct graph
3. Apply Chinese Whispers on graph.
4. Add the feature words not present in the partition as one-member clusters.

The graph construction in step 2 is conducted by adding an edge between two words a and b with weight[4] $w = 1/(1 - cos(\overrightarrow{a}, \overrightarrow{b}))$, computed using the feature vectors $\overrightarrow{a}$ and $\overrightarrow{b}$ (cf. Figure 5.4) of words a and b. The edge is only drawn if w exceeds a similarity threshold s. The latter influences the number of words that actually end up in the graph and get clustered. It might be desired to cluster fewer words with higher confidence as opposed to running the risk of joining two unrelated clusters because of too many ambiguous words that connect them. After step 3, there is already a partition of a subset of target words that can be perceived as tagset. Figure 5.6 shows the weighted graph and its CW-partition for the example given in Figure 5.4. The number of target words is limited by computational considerations: since the feature vectors have to be compared in a pair-wise fashion, a considerably higher number of target words results in long run times. The number of feature words was examined in preliminary experiments, showing only minor differences with respect to cluster quality in the range of 100–300.

As noted e.g. in (Schütze, 1995)), the clusters are motivated syntactically as well as semantically and several clusters per open word class can be observed. The distinctions are normally finer-grained than existing tagsets, as Figure 5.7 illustrates.

Since the feature words form the bulk of tokens in the corpus, it is clearly desired to make sure that they appear in the tagset, although

[4] cosine similarity is a standard measure for POS induction, however, other measures would be possible

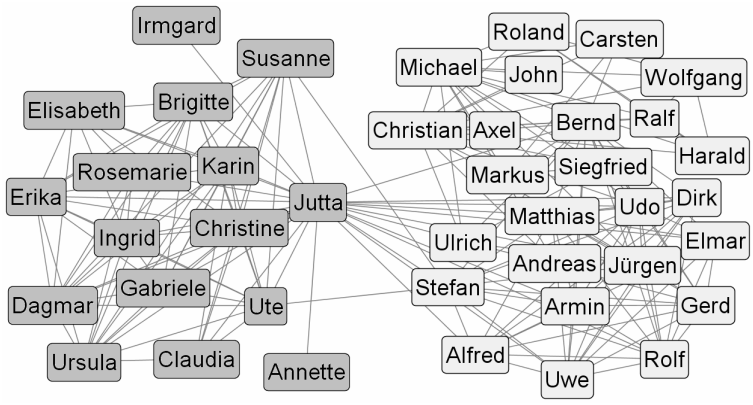


Figure 5.7: Fine-grained distinctions: female and male first names from German corpus. The figure shows only a local neighbourhood of the graph for tagset 1

they might end up in clusters with only one element. This might even be desired, e.g. for English *'not'*, which usually has its own POS-tag in linguistic tagsets. This is done in step 4, where assigning separate word classes for high frequency words is considered to be a more robust choice than trying to disambiguate them while tagging. Starting from this, it is possible to map all words contained in a cluster onto one feature and iterate this process, replacing feature words by the clusters obtained (cf. Schütze, 1993). In that way, higher counts in the feature vectors are obtained, which could provide a better basis for similarity statistics. Preliminary experiments showed, however, that only marginal improvements could be reached, as text coverage is not substantially increased.

Table 5.3 shows as illustration a selection of clusters for the BNC. Several clusters for nouns can be observed. Evaluating lexical clusters against a gold standard may lead to inconclusive results, because the granularities of the gold standard and the clusters usually differ, e.g. English singular and plural nouns end up in one cluster, but first and last names are distinguished. The evaluation scores are largely depending on the tagset used for gold standard. Here, an information-theoretic measure is employed that allows an intuitive interpretation: Entropy precision (EP) measures the extent to which the gold standard classification is reconstructable from the clustering result. EP directly relates to the precision measure in information retrieval. Its counterpart, recall as the number of retrieved vs. the total number of instances relates to the coverage on target words as reached by the clustering algorithm. For the gold

standard, each word gets assigned its most frequent tag, ignoring POS-ambiguities. Despite all these disadvantages, EP provides a means to relatively compare the quality of partitions for varying thresholds s.

Definition Entropy Precision (EP): Let $G = G_1, ...G_m$ be the gold standard classification and $C = C_1, ...C_p$ be the clustering result. Then, EP is computed as follows:

$$EP(C,G) = \frac{M_{CG}}{I_G}$$

with mutual information M_{XY} between X and Y

$$M_{XY} = \sum_{xy} P(x,y) ln \frac{P(x,y)}{P(x)P(y)}$$

and I_X entropy of X.

$$I_X = -\sum_{x} P(x) ln P(x)$$

A maximal EP of 1 is reached by a trivial clustering of singleton clusters. This does not impose a severe problem, considering the typical cluster size distribution as depicted in Figure 5.8. Nevertheless, optimising EP promotes a large number of small clusters, which is why the number of clusters has to be provided along with the EP figures to give an impression of the result's quality. A minimal EP of 0 indicates statistical independence of C and G.

For evaluation of tagset 1, three corpora of different languages were chosen: 10 million sentences of German tagged with 52 tags using TreeTagger (Schmid, 1994), the 6 million sentences of BNC for English, pretagged semi-automatically with the CLAWS tagset of 84 tags (Garside *et al.*, 1987) and 1 million sentences from a Norwegian web corpus tagged with the Oslo-Bergen tagger (Hagen *et al.*, 2000), using a simplified tagset of 66 tags. Figure 5.9 gives the EP results for varying numbers of target words included in the partition and the number of clusters.

From Figure 5.9 it is possible to observe that EP remains stable for a wide range of target word coverage between about 2,000-9,000 words. The number of parts is maximal for the medium range of coverage: at higher coverage, POS-ambiguous words that are related to several clusters serve as bridges. If too many links are established between two clusters, CW will collapse both into one cluster, possibly at cost of EP. At lower coverage, many classes are left out. This

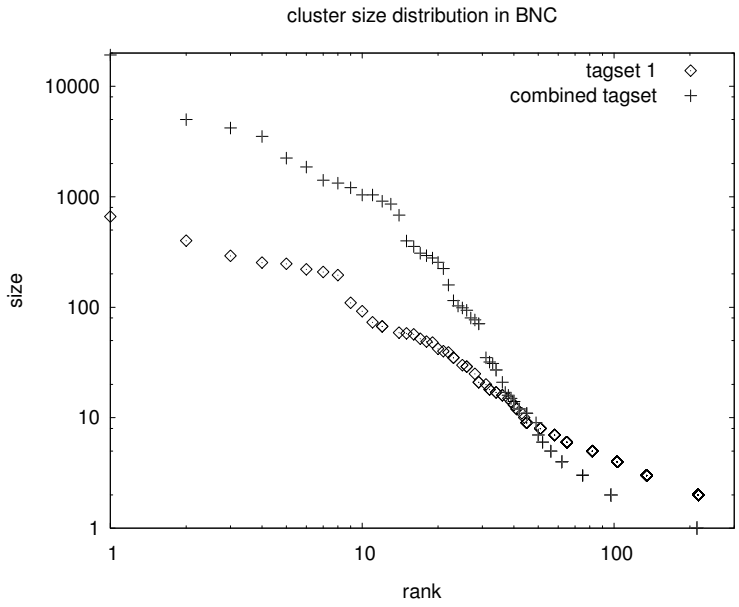


Figure 5.8: Cluster size distribution for tagset 1 and combined tagset in the BNC

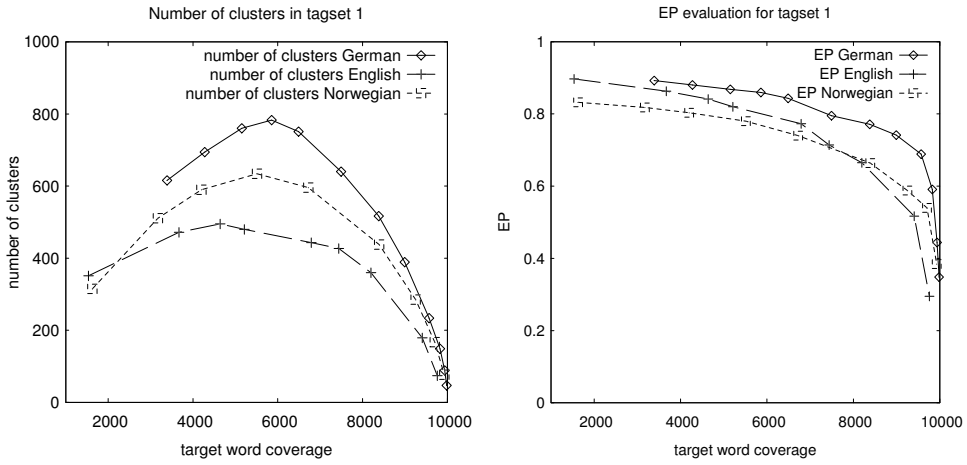


Figure 5.9: Tagset size and Entropy precision dependent on number of included target words for tagset 1

rank	size	gold standard tags (count)	description	sample words
1	662	NN1(588), NN2(44)	Singular Nouns	day, government, world, system, company, house, family
2	401	NN1(311), NN2(86)	Singular Nouns	part, end, state, development, members, question, policy, ...
3	292	NN2(284), NN1(7)	Plural Nouns	men, services, groups, companies, systems, schools, ...
4	254	NP0(252), NN1(2)	First Names	John, David, Peter, Paul, George, James, Michael, ...
5	247	AJ0(233), NN1(9)	Adjectives	social, political, real, economic, national, human, private, ...
6	220	NN1(148), NN2(63)	Singular and Plural Nouns	business, water, service, staff, land, training, management, ...
7	209	VVI(209)	Verbs	get, make, take, give, keep, provide, play, move, leave, ...
8	195	AJ0(118), NN1(25)	Adjectives (country)	British, police, New, European, individual, National, ...
9	110	NP0(109), NN1(1)	Last names	Smith, Jones, Brown, Wilson, Lewis, Taylor, Williams, ...
10	92	AJ0(89), CRD(1)	Adjectives (size/quality)	new, good, little, few, small, great, large, major, big, special
11	73	AJ0(73)	Adjectives (animate)	heavy, beautiful, quiet, soft, bright, charming, cruel, ...
12	67	NN2(67)	Plural Nouns	problems, conditions, costs, issues, activities, lines, ...
12	67	NP0(66), NN1(1)	Countries	England, Scotland, France, Germany, America, Ireland, ...
16	57	NP0(57)	Cities	Oxford, Edinburgh, Liverpool, Manchester, Leeds, Glasgow, ...
22	39	AV0(39)	Sentence Beginning	Well, However, Thus, Indeed, Also, Finally, Nevertheless, ...
25	30	NN2(30)	Plural Professions	teachers, managers, farmers, governments, employers, ...
34	17	CRD(17)	Numbers	three, four, five, six, ten, eight, seven, nine, twelve, fifteen, ...
65	6	NP0(6)	Titles	Mr, Mrs, Dr, Miss, Aunt, Ms
217	2	AT0(2)	Indefinite determiner	a, an
217	2	NP0(2)	location 1st	Saudi, Sri
217	2	VVZ, VVD	to wear	wore, wears
217	2	VVZ, VVD	to insist	insisted, insists

Table 5.3: Selected clusters from the BNC clustering for setting s such that the partition contains 5,000 words. In total, 464 clusters are obtained. EP for this partition is 0.8276. Gold standard tags have been gathered from the BNC

evaluation indicates the language-independence of the method, as results are qualitatively similar for all languages tested.

As indicated above, lexicon size for tagset 1 is limited by the computational complexity of step 2, which is time-quadratic in the number of target words. Due to the non-sparseness of context vectors of high-frequency words there is not much room for optimisation. In order to add words with lower frequencies, another strategy is pursued.

5.2.5 Tagset 2: Medium and Low Frequency Words

As noted in (Dunning, 1993), log likelihood statistics capture word bigram regularities. Given a word, its neighbouring co-occurrences as ranked by their log likelihood ratio are the typical immediate contexts of the word. Regarding the highest ranked neighbours as the profile of the word, it is possible to assign similarity scores between two words A and B according to how many neighbours they share, i.e. to what extent the profiles of A and B overlap. The hypothesis here is that words sharing many neighbours should usually be observed with the same part-of-speech. For the acquisition of word classes in tagset 2, the second-order graph on neighbouring co-occurrences is used, cf Section 3.2.2. To set up the graph, a co-occurrence calculation is performed which yields word pairs based on their significant co-occurrence as immediate neighbours. Here, all word pairs exceeding a log likelihood threshold of 1.00 (corresponding to a positive correlation, yet the outcome is robust in a wide threshold range) enter this bipartite graph. Note that if similar words occur in both parts, they form two distinct vertices. Only words with a frequency rank higher than 2,000 are taken into account: as preliminary experiments revealed, high-frequency words of closed word classes spread over the clusters, resulting in deteriorated tagging performance later, so they are excluded in this step.

This graph is transformed into a second-order graph by comparing the number of common right and left neighbours for two words. The similarity (edge weight) between two words is the sum the number of common neighbours on both sides. Figure 5.10 depicts the significant neighbouring graph, the second-order graph derived from it, and its CW-partition. The word-class-ambiguous word 'drink' (to drink the drink) is responsible for all inter-cluster edges. In the example provided in Figure 5.10, three clusters are obtained that correspond to different parts-of-speech. For computing the similarities based on the significant neighbour-based word co-occurrence

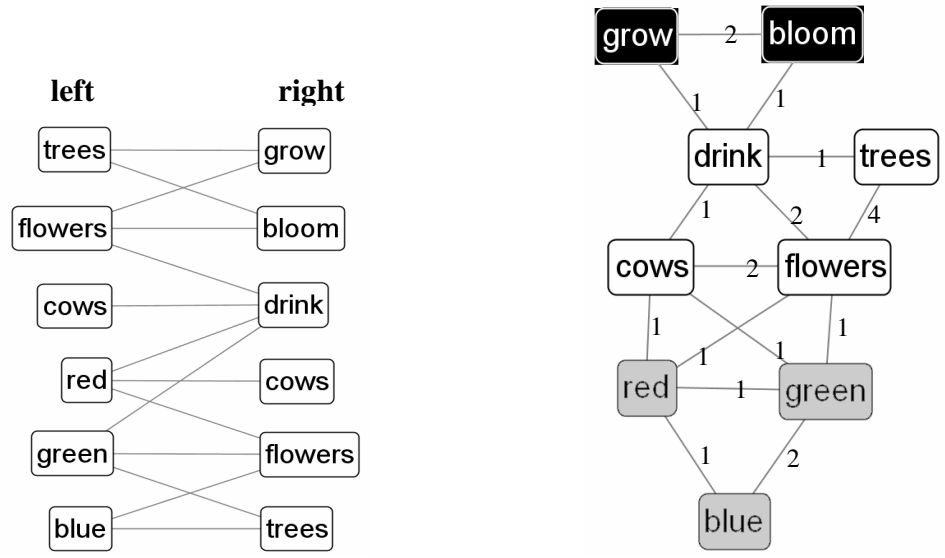


Figure 5.10: Left: Bi-partite neighbouring co-occurrence graph. Right: second-order graph on neighbouring co-occurrences clustered with CW

graphs for both directions, at maximum the 150 most significant co-occurrences per word are considered, which regulates the density of ther graph and leads to improvements in run-time.

To test this on a large scale, the second-order similarity graph for the BNC was computed, excluding the most frequent 2,000 words and drawing edges between words only if they shared at least four left and four right common neighbours. The clusters are checked against a lexicon that contains the most frequent tag for each word in the BNC. The largest clusters are presented in Table 5.4, together with the predominant tags in the BNC.

In total, CW produced 282 clusters, of which 26 exceed a size of 100. The weighted average of cluster purity (i.e. the number of predominant tags divided by cluster size) was measured at 88.8%, which exceeds significantly the precision of 53% on word type as reported by Schütze (1995).

Again, several hundred clusters, mostly of open word classes are obtained. For computing tagset 2, an efficient algorithm like CW is crucial: the graphs as used for the experiments consist typically of 10,000 to 100,000 vertices and about 100,000 to 1 million edges.

size	tags(count)	sample words
18432	NN(17120), AJ(631)	secret, officials, transport, unemployment, farm, county, wood, procedure, grounds, ...
4916	AJ(4208), V(343)	busy, grey, tiny, thin, sufficient, attractive, vital, ...
4192	V(3784), AJ(286)	filled, revealed, experienced, learned, pushed, occurred, ...
3515	NP(3198), NN(255)	White, Green, Jones, Hill, Brown, Lee, Lewis, Young, ...
2211	NP(1980), NN(174)	Ian, Alan, Martin, Tony, Prince, Chris, Brian, Harry, Andrew, Christ, Steve, ...
1855	NP(1670), NN(148)	Central, Leeds, Manchester, Australia, Yorkshire, Belfast, Glasgow, Middlesbrough, ...

Table 5.4: The largest clusters of tagset 2 for the BNC

5.2.6 Combination of Tagsets 1 and 2

Now, there are two tagsets of two different, yet overlapping frequency bands. A large portion of these 8,000 words in the overlapping region is present in both tagsets. Again, a graph is constructed, containing the clusters of both tagsets as vertices; weights of edges represent the number of common elements, if at least two elements are shared. Notice that the graph is bipartite.

And again, CW is used to cluster this graph of clusters. This results in fewer clusters than before for the following reason: while the granularities of tagsets 1 and 2 are both high, they capture different aspects as they are obtained from different sources. Vertices of large clusters (which usually consist of open word classes) have many edges to the other partition's vertices, which in turn connect to yet other clusters of the same word class. Eventually, these clusters can be grouped into one.

Clusters that are not included in the graph of clusters are treated differently, depending on their origin: clusters of tagset 1 are added to the result, as they are believed to contain important closed word class groups. Dropouts from tagset 2 are simply left out, as they mostly consist of small, yet semantically motivated word sets. The total loss of words by disregarding these many, but small clusters did never exceed 10% in any experiment. Figure 5.11 illustrates this combination process.

Conducting this combination yields about 300–600 clusters that

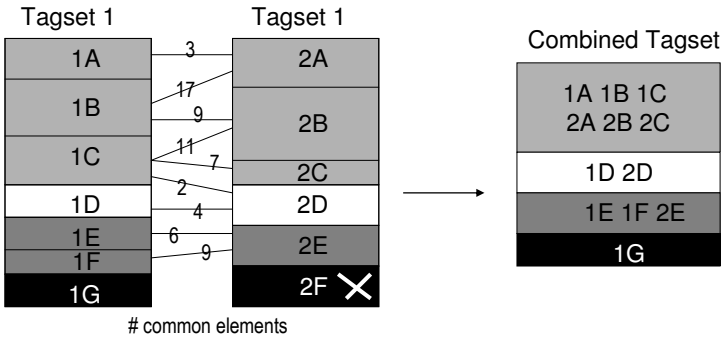


Figure 5.11: Combination process: tagsets 1 and 2 are related via the number of common elements in their respective clusters. Shades symbolise the outcome of Chinese Whispers on this graph of clusters. Clusters marked with x are not included in the resulting graph of clusters

will be further used as a lexicon for tagging. As opposed to the observations made in (Schütze, 1995), only a handful of clusters are found per open word class, of which most distinctions are syntactically motivated, e.g. adjectives with different case markers. For unsupervised POS tagging, the aim is to arrive at a low number of clusters to mimic the supervised counterparts. A more rigid method to arrive at yet less clusters would be to leave out classes of low corpus frequency.

5.2.7 Setting up the Tagger

Lexicon Construction

From the merged tagsets, a lexicon is constructed that contains one possible tag (the cluster ID) per word. To increase text coverage, it is possible to include those words that dropped out in the distributional step for tagset 1 into the lexicon. It is assumed that some of these words could not be assigned to any cluster because of ambiguity. From a graph with a lower similarity threshold s (here: such that the graph contains 9,500 target words), neighbourhoods of these words are obtained one at a time. This is comparable to the methodology in (Ertöz *et al.*, 2002), where only some vertices are used for clustering and the rest is assimilated. Here, the added target words are not assigned to only one cluster: the tags of their neighbours – if known – provide a distribution of possible tags for these words. Figure 5.12 gives an example: the name 'Miles' (frequency rank 8,297 in the BNC) is rated 65% as belonging to a first name cluster and 35%

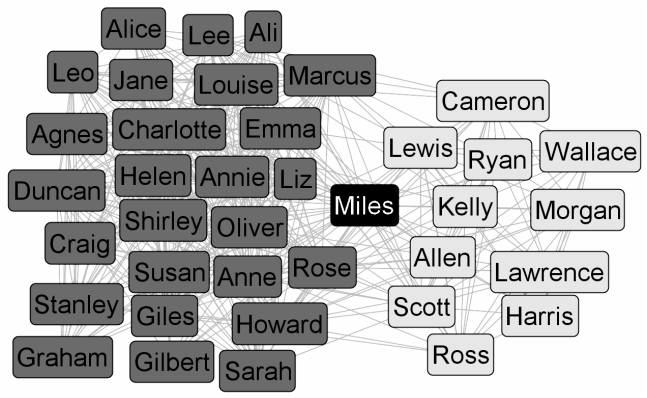


Figure 5.12: POS-disambiguation in the BNC for 'Miles' as first and last name. Note that most of the last names are ambiguous themselves, causing 'Miles' to be similar to them

as last name.

Constructing the Tagger

Unlike in supervised scenarios, the task is not to train a tagger model from a small corpus of hand-tagged data, but from the clusters of derived syntactic categories and a large, yet unlabelled corpus. This realises a class-based N-gram model (Brown *et al.*, 1992).

Here, a simple trigram Viterbi model without re-estimation techniques (such as Baum-Welch) is employed in order not to blur the quality of lexicon construction and to speed up processing. Adapting a previous standard POS-tagging framework (cf. Charniak *et al.*, 1993), the probability of the joint occurrence of tokens t_i and categories c_i for a sequence of length n is maximised:

$$P(T, C) = \prod_{i=1}^{n} P(c_i|c_{i-1}, c_{i-2})P(c_i|t_i)$$

The transition probability $P(c_i|c_{i-1}, c_{i-2})$ is estimated from word trigrams in the corpus whose elements are all present in the lexicon. The last term of the product, namely $P(c_i|t_i)$, is dependent on the lexicon. If the lexicon does not contain t_i, then c_i only depends on neighbouring categories, i.e. $P(c_i|t_i) = 1$. Words like these are called out-of-vocabulary (OOV) words.

Although Charniak *et al.* (1993) report that using $P(t_i|c_i)$ for the last term leads to superior results in the supervised setting, this 'di-

rect' lexicon probability is used here. The experiments in (Charniak *et al.*, 1993) were carried out for small, labelled corpora in a supervised setting. The main advantage of the current standard model, better smoothing capability, is not an issue when using much larger corpora, as conducted here. For an efficient implementation, beam search (cf. (Brants, 2000) in a tagging context) is employed to keep only the 5 most probable states per token position. The beam width of 5 is a safe choice, as preliminary experiments showed that already a beam width of 3 produces practically equal tagging results compared to using all states.

Morphological Extension

The main performance flaw of supervised POS taggers originates from OOV words. Morphologically motivated add-ons are used e.g. in (Clark, 2003; Freitag, 2004a) to guess a more appropriate category distribution based on a word's suffix or its capitalisation. Here, Compact Patricia Trie classifiers (CPT, (see Knuth, 1998)) trained on prefixes and suffixes are employed. For OOV words, the category-wise product of both classifier's distributions serve as probabilities $P(c_i|t_i)$: Let $w = ab = cd$ be a word, a be the longest common prefix of w and any lexicon word, and d be the longest common suffix of w and any lexicon words. Then

$$P(c_i|w) = \frac{|\{u|u = ax \wedge class(u) = c_i\}|}{|\{u|u = ax\}|} \cdot \frac{|\{v|v = yd \wedge class(v) = c_i\}|}{|\{v|v = yd\}|}.$$

CPTs do not only serve as a substitute lexicon component, they also handle capitalisation, camelCase and suffix endings without having to define features explicitly or setting length or maximal number thresholds (as in (Freitag, 2004a) for suffixes). A similar technique is employed by Cucerzan and Yarowsky (1999) in the context of named entity recognition. My implementation of CPTs is further used in supervised settings by Witschel and Biemann (2005) for compound splitting and in (Eiken *et al.*, 2006) for base form reduction, where it is described in more detail.

5.2.8 Direct Evaluation of Tagging

Adopting the methodology of Freitag (2004a), the cluster-conditional tag perplexity PP as the average amount of uncertainty to predict the tags of a POS-tagged corpus, given the tagging with classes from the unsupervised method is measured: for the same corpus tagged with

language	sent.	tokens	tagger	nr.tags	200 cov.	10K cov.
English	6M	100M	BNC	84	55%	90%
Finnish	3M	43M	Connexor[5]	31	30%	60%
German	10M	177M	(Schmid, 1994)	54	49%	78%

Table 5.5: Characteristics of corpora for POS induction evaluation: number of sentences, number of tokens, tagger and tagset size, corpus coverage of top 200 and 10,000 words

two methods, the measure indicates how well one tagging can be reproduced from the other. Let

$$I_x = -\sum_x P(x) ln P(x)$$

be the entropy of a random variable X and

$$M_{XY} = \sum_{xy} P(x,y) ln \frac{P(x,y)}{P(x)P(y)}$$

be the mutual information between two random variables X and Y. Then the cluster-conditional tag perplexity for a gold-standard tagging T and a tagging resulting from clusters C is computed as

$$PP = exp(I_{T|C}) = exp(I_T - M_{TC}).$$

Minimum PP is 1.0, connoting a perfect congruence with gold standard tags. Below, PP on lexicon words and OOV words is reported separately. The objective is to minimise the total PP.

Unsupervised POS-tagging is meant for yet untagged text, so a system should be robustly performing on a variety of typologically different languages. For evaluating tagging performance, three corpora are chosen: the BNC for English, a 10 million sentences newspaper corpus from LCC for German, and 3 million sentences from LCC's Finnish web corpus. Table 5.5 summarises some characteristics.

Since a high text coverage is reached with only a few words in English, a strategy that assigns only the most frequent words to sensible clusters already ensures satisfactory performance. In the Finnish case, a high OOV rate can be expected, hampering performance of strategies that cannot cope well with low frequency or unseen words.

To put the results in perspective, the following baselines on random samples of the same 1,000 randomly chosen sentences used for evaluation were computed:

- 1: the trivial top clustering: all words are in the same cluster

language	English			Finnish			German		
baseline	1	200	400	1	200	400	1	200	400
PP	29	3.6	3.1	20	6.1	5.3	19	3.4	2.9

Table 5.6: Baselines for various tagset sizes

- 200: the most frequent 199 words form clusters of their own; all the rest is put into one cluster.
- 400: same as 200, but with 399 most frequent words

Table 5.6 summarises the baselines in terms of PP.

Influence of System Components

The quality of the resulting taggers for combinations of several substeps is measured using:

- O: tagset 1
- M: the CPT morphology extension
- T: merged tagsets 1 and 2
- A: adding ambiguous words to the lexicon

Figure 5.13 illustrates the influence of the similarity threshold s for O, O+M and O+M+A for German – for other languages, results look qualitatively similar. Varying s influences coverage on the 10,000 target words. When clustering on very few words, tagging performance on these words reaches a PP as low as 1.25 but the high OOV rate impairs the total performance. Clustering too many words results in deterioration of results – most words end up in one big part. In the medium ranges, higher coverage and lower known PP compensate each other, optimal total *PP*s were observed at target word coverages of 4,000-8,000. The system's performance is stable with respect to changing thresholds, as long as it is set in reasonable ranges. Adding ambiguous words results in a worse performance on lexicon words, yet improves overall performance, especially for high thresholds.

For all further experiments, the threshold s was fixed in a way that tagset 1 consisted of 5,000 words, so only half of the top 10,000 words are considered unambiguous. At this value, the best performance throughout all corpora tested was achieved.

Overall results are presented in Table 5.7. The combined strategy T+M+A reaches the lowest PP for all languages. The morphology

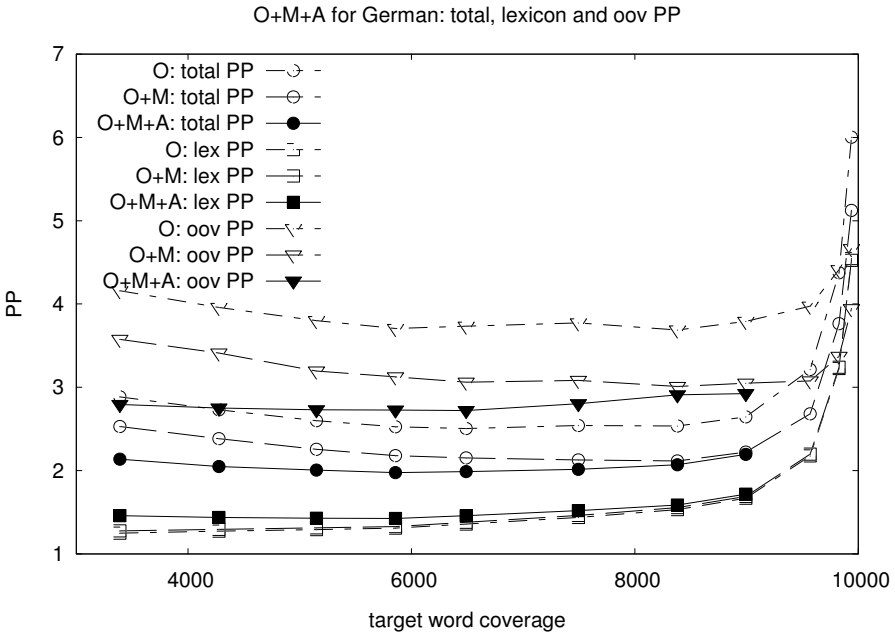


Figure 5.13: Influence of threshold s on tagger performance: cluster-conditional tag perplexity PP as a function of target word coverage for tagset 1

lang	words	O	O+M	O+M+A	T+M	T+M+A
EN	total	2.66	2.43	2.08	2.27	2.05
	lex	1.25		1.51	1.58	1.83
	OOV	6.74	6.70	5.82	9.89	7.64
	OOV%	28.07		14.25	14.98	4.62
	tags	619			345	
FI	total	4.91	3.96	3.79	3.36	3.22
	lex	1.60		2.04	1.99	2.29
	OOV	8.58	7.90	7.05	7.54	6.94
	OOV%	47.52		36.31	32.01	23.80
	tags	625			466	
GER	total	2.53	2.18	1.98	1.84	1.79
	lex	1.32		1.43	1.51	1.57
	OOV	3.71	3.12	2.73	2.97	2.57
	OOV%	31.34		23.60	19.12	13.80
	tags	781			440	

Table 5.7: Results in PP for English, Finnish, German. OOV% is the fraction of non-lexicon words in terms of tokens

Word	cluster ID	cluster members (size)
I	166	I (1)
saw	2	*past tense verbs* (3818)
the	73	a, an, the (3)
man	1	*nouns* (17418)
with	13	*prepositions* (143)
a	73	a, an, the (3)
saw	1	*nouns* (17418)
.	116	. ! ? (3)

Table 5.8: Tagging example

extension (M) always improves the OOV scores. Adding ambiguous words (A) hurts the lexicon performance, but largely reduces the OOV rate, which in turn leads to better overall performance. Combining both partitions (T) does not always decrease the total PP a lot, but lowers the number of tags significantly.

Finnish figures are generally worse than for the other languages, consistent with higher baselines. Differences between languages are most obvious when comparing O+M+A and T+M: whereas for English it pays off much more to add ambiguous words than to merge the two partitions, it is the other way around in the German and Finnish experiments.

To sum up the discussion of results: all introduced steps improve the performance, yet their influence's strength varies. As a sample of the system's output, consider the example in Table 5.8 that has been tagged by the English T+M+A model: as in the example above, 'saw' is disambiguated correctly. Further, the determiner cluster is complete; unfortunately, the pronoun 'I' constitutes a singleton cluster.

The results can be compared to (Freitag, 2004a); most other work uses different evaluation techniques that are only indirectly measuring what is tried to optimise here. Unfortunately, Freitag (2004a) does not provide a total PP score for his 200 tags. He experiments with a hand-tagged, clean English corpus that is not free (the Penn Treebank) and is therefore not an option here. Freitag reports a PP for known words of 1.57 for the top 5,000 words (91% corpus coverage, baseline 1 at 23.6), a PP for unknown words without morphological extension of 4.8. Using morphological features the unknown PP score is lowered to 4.0. When augmenting the lexicon with low frequency words via their distributional characteristics, a PP as low as 2.9 is obtained for the remaining 9% of tokens. His methodology, however, does not allow for class ambiguity in the lexicon, the

low number of OOV words is handled by a Hidden Markov Model trained with Baum-Welch-Reestimation. It is hard to relate results to (Clark, 2003): there, much smaller corpora and smaller tagsets are used, resulting in worse performance. Notice that Clark reports conditional entropy and not perplexity. His method is computationally much more expensive than the approach discussed here. For that reason, it does not scale to much larger corpora.

Influence of Parameters

A number of parameters for the process of unsupervised POS tagging were introduced at the points where they arised. Now, all parameters are listed for recapitulating the possibilities to fine-tune the method. Table 5.9 gives the parameters, a short explanation, and the default setting used in all experiments.

Their influence and interplay is outlined as follows. *FEAT* did not show to have a large influence in ranges 100–300. It might be adviseable to use higher values for languages with low Zipfian exponents (such as Finnish) to gain higher text coverage for building tagset 1. When processing small corpora, *TARG* should not be too high, because a low corpus frequency for target words results in unreliable context statistics. The parameter *CWTARG* must be set smaller than *TARG*, Figure 5.13 indicates that 40%–80% of *TARG* is a sensible range. Higher settings result in more words that can overlap for combining the two tagsets.

NB_SIG and *NB_THRESH* go hand in hand to regulate the density of the graph for tagset 2: Lower significance thresholds lead to more edges in the neighbouring co-occurrence graph, higher values for *NB_THRESH* prune edges in the graph for tagset 2. The maximum number of neighbouring co-occurrences per word *NB_MAX* influences the density of the graph for tagset 2, lower settings result in less edges per word. All experiments were carried out with the default value, however, higher values lead to more coarse-grained tagsets that e.g. join common and proper nouns. Different settings could prove advantageous for different applications, but no experiments were conducted to measure to what extent.

BEHEAD should be set in a way that stop words are excluded from tagset 2, but considerably lower than *TARG*, to enable sufficient overlap between the two tagsets. Less than a value of 2 for *CONF_OVERLAP* can result in spurious combinations in the graph of clusters, higher values reduce the lexicon size since clusters from tagset 2 are more likely to be excluded.

Parameter	Default	Explanation
FEAT	200	Number of feature words for tagset 1 similarities
TARG	10,000	Number of target words for tagset 1 similarities
CWTARG	5000	Number of words that are clustered for tagset 1 amongst *TARG* words, by applying an appropriate similarity threshold *s* on the graph
TOPADD	9500	Number of words that are considered for adding ambiguous words amonst *TARG* words, by applying an appropriate similarity threshold *s* on the graph
NB_SIG	1.00	Significance threshold for neighbour-based co-occurrences
NB_THRESH	2	Minimum number of common neighbour-based co-occurrences per side for constituting an edge in the graph for tagset 2
NB_MAX	150	Maximum neighbouring co-occurrences per word to consider for the second-order graph of tagset 2
CONF_OVERLAP	2	Minimum number of common words for connecting partitions in the graph of clusters to merge tagset 1 and 2
BEHEAD	2000	Minimum rank of words to enter the graph for tagset 2
SING_ADD	200	Maximum frequency rank of words to add as singletons, if not already contained in the combined tagset.

Table 5.9: Parameters, default settings and explanation

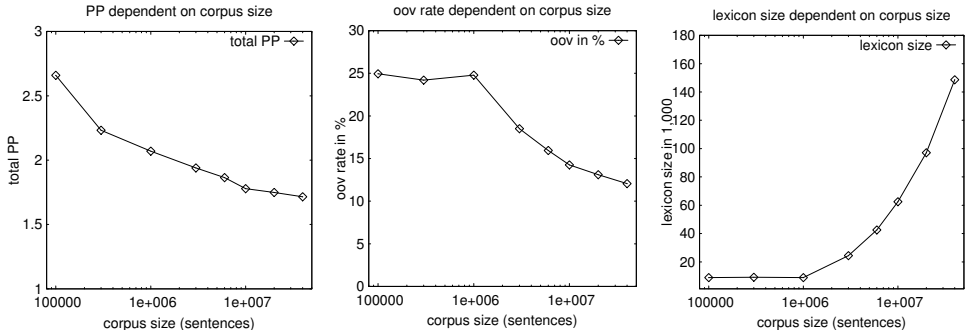


Figure 5.14: PP, OOV rate and lexicon size vs. corpus size for German

Adding more singletons amongst the $SING\_ADD$ most frequent words increases the number of tags, but also the number of trigrams available for training the Viterbi tagger.

A sensible extension would be to limit the total number of tags by excluding thse clusters from the combined tagset that have the lowest corpus frequency, i.e. the sum of frequencies of the lexicon words constituting this tag.

Influence of Corpus Size

Having determined a generic setting for the interplay of the different system components (M+T+A), the influence of corpus size on the cluster-conditional tag perplexity PP shall be examined now. For this purpose, taggers were induced from German corpora of varying size from 100,000 sentences up to 40 million sentences, taken from LCC. Evaluation was carried out by measuring PP between the resulting taggers and the hand-tagged German NEGRA corpus (Brants *et al.*, 1997), testing on all 20,000 sentences of NEGRA. The evaluation corpus was not part of the corpora used for tagger induction. Figure 5.14 provides total PP, the OOV rate and the lexicon size, dependent on corpus size.

Not surprisingly, the more corpus is provided for tagger induction, the better performance levels are reached in terms of PP. The more data provided, the more reliable the statistics for tagset 1, which is reflected in tremendous PP improvement from using 100,000 to 1,000,000 sentences. In this range, tagset 2 is almost empty and does not contribute to the lexicon size, which is mirrored in a constant OOV rate for this range. Above 1 million sentences, the size of tagset 2 increases, resulting in lower PP and OOV rates. The lexicon size explodes to some 100,000 entries for a corpus size of 40 million sen-

tences. Summarising the results obtained by training the unsupervised POS tagger on corpora of various sizes, there always seems to be room for improvements by simply adding more data. However, improvements beyond 10 million sentences are small in terms of PP.

The interpretation of the PP measure is difficult, as it largely depends on the gold standard. While it is possible to relatively compare the performance of different components of a system or different systems along these lines, it only gives a poor impression on the utility of the unsupervised tagger's output. Therefore, several application-based evaluations are undertaken in the following section.

5.2.9 Application-based Evaluation

POS taggers are a standard component in any applied NLP system, and their utility is hardly questioned at all. In this section, a number of NLP tasks are viewed as machine learning problems: the POS tagger component provides some of the features that are used to learn a function that assigns a label to unseen examples, characterised by the same set of features as the examples used for training. In this setting, it is straightforward to evaluate the contribution of POS taggers – be they supervised or unsupervised – by providing the different POS tagging annotations to the learning algorithm or not.

Having advanced machine learning algorithms at hand that automatically perform feature weighting and selection, the standard approach to NLP systems is to provide all possible features and to leave the choice to the learning algorithm.

The task-performing systems for application-based evaluation were chosen to cover a wide range of machine learning paradigms: Markov chains in a POS tagging system, kernel methods in a word sense disambiguation (WSD) system and Conditional Random Fields (CRFs, (see Lafferty *et al.*, 2001)) for named entity recognition (NER) and chunking. The WSD and CRF experiments were conducted by Claudio Giuliano and Alfio Gliozzo, who are both co-authors of (Biemann *et al.*, 2007), where the results of this section have been published.

All evaluation results are compared in a pair-wise fashion using the approximate randomisation procedure of Noreen (1989) as significance test, for which p-values as error probabilities are given, i.e. a significant difference with $p<0.01$ means that the test is more than 99% sure that the difference has not been caused by chance.

Unsupervised POS for Supervised POS

The Viterbi tagger described in Section 5.2.7 is a very simple trigram tagger that does not use parameter re-estimation or smoothing techniques. It is designed for a large amount of training data as present in the unsupervised setting, rather than for small training sets in the supervised case. For training, the frequency of tag trigrams and the number of times each word occurs with each tag are counted and directly transformed into (transition) probabilities by normalisation.

In the unsupervised case, the transition probabilities $P(c_i|c_{i-1}, c_{i-2})$ are only estimated from trigrams where all three tags are present, whereas in the supervised case, tags are provided for all tokens in the training corpus. The probability $P(c_i|t_i)$ is obtained from the tagger's lexicon and equals 1 if t_i is not contained.

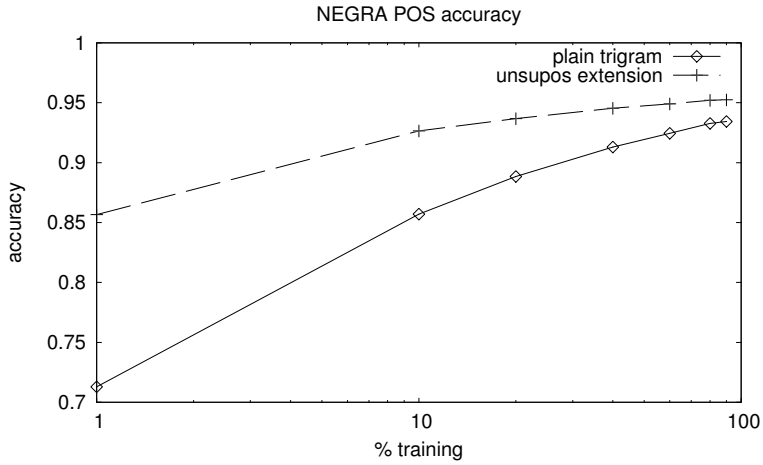
For the inclusion of unsupervised tags, another factor $P(c_i|u_i)$ is introduced that accounts for the fraction of times the supervised tag c_i was found together with the unsupervised tag u_i in the training text:

$$P_{unsupos}(T, C) = \prod_{i=1}^{n} P(c_i|c_{i-1}, c_{i-2})P(c_i|t_i)P(c_i|u_i)$$

Notice that only the unsupervised tag at the same position influences the goal category in this simple extension. Using surrounding unsupervised tags would be possible, but has not been carried out here. More elaborate strategies, like morphological components as in the unsupervised setting (also cf. Brants, 2000)) or the utilisation of a more up-to-date tagger model, are not considered for the supervised tagger. The objective is to examine the influence of unsupervised tags, not to construct a state of the art POS tagger. Ushioda (1996) describes a somewhat related strategy, where a hierarchical clustering of words is used for improvement of a decision-tree-based tagger.

Training sets of varying sizes are selected randomly from the 20,000 sentences of NEGRA corpus, the respective remainders are used for evaluation. The performance of the plain Viterbi tagger is compared with the performance of the tagger using unsupervised tags. These are obtained by tagging the NEGRA corpus with the tagger model induced on 40 million sentences from the Wortschatz project as evaluated in Figure 5.14, Section 5.2.8. Results are reported in tagging accuracy (number of correctly assigned tags divided by total number of tokens), averaged over three different splits per training size each. Figure 5.15 shows the learning curve.

Results indicate that supervised tagging can clearly benefit from



% training	1	10	20	40	60	80	90
plain	0.713	0.857	0.888	0.913	0.925	0.933	0.934
unsupos	0.857	0.926	0.937	0.946	0.949	0.952	0.953

Figure 5.15: Learning curve for supervised POS tagging with and without using unsupervised POS tags (accuracy). Differences between the two models are significant with $p<0.01$ for all percentages of training

unsupervised tags: already at 20% training with unsupervised tags, the performance on 90% training without the unsupervised extension is surpassed. At 90% training, error rate reduction is 27.8%, indicating that the unsupervised tagger grasps very well the linguistically motivated syntactic categories and provides a valuable feature to either reduce the size of the required annotated training corpus or to improve overall accuracy. Despite its simplicity, the unsupervised extension comes close to the performance of (Brants, 2000), who reports an accuracy of 0.967 at 90% training on the same corpus.

Unsupervised POS for Word Sense Disambiguation

The task in word sense disambiguation (WSD) is to assign the correct word sense to ambiguous words in a text based on the context. The senses are provided by a sense inventory (usually WordNet, (Miller *et al.*, 1990)). Supervised WSD is trained on examples where the correct sense is provided manually, and tested by comparing the system's outcome on held-out examples.

In the WSD literature, many algorithms have been proposed, characterised by different feature sets and classification algorithms. The

state of the art supervised WSD methodology, reporting the best results in most of the Senseval-3 lexical sample tasks (Mihalcea *et al.*, 2004b) in different languages, is based on a combination of syntagmatic and domain kernels (Gliozzo, 2005) in a Support Vector Machine classification framework.

A great advantage of this methodology is that all its pre-processing steps are also unsupervised and knowledge-free and therefore comply to the SD paradigm. It has been shown here that the only language-dependent component in the system of Gliozzo *et al.* (2005) – a supervised POS tagger – can safely be replaced by the unsupervised POS tagger.

Kernel WSD basically encompasses two different aspects of similarity: domain aspects, mainly related to the topic (i.e. the global context) of the texts in which the word occurs, and syntagmatic aspects, concerning the lexical-syntactic pattern in the local contexts. Domain aspects are captured by the domain kernel, while syntagmatic aspects are taken into account by the syntagmatic kernel.

The domain kernel handles domain aspects of similarity among two texts based on the Domain Model as introduced in (Gliozzo, 2005), which is a soft clustering of terms reflecting semantic domains. On the other hand, syntagmatic aspects are probably the most important evidence while recognising sense similarity. In general, the strategy adapted to model syntagmatic relations in WSD is to provide bigrams and trigrams of collocated words as features to describe local contexts (Yarowsky, 1994). The main drawback of this approach is that non-contiguous or shifted collocations cannot be identified, decreasing the generalisation power of the learning algorithm.

The syntagmatic kernel allows estimating the number of common non-continuous subsequences of lemmas (i.e. collocations) between two examples, in order to capture syntagmatic similarity. Analogously, the POS Kernel is defined to operate on sequences of parts-of-speech. The syntagmatic kernel is given by a linear combination of the collocation kernel and the POS kernel.

The modularity of the kernel approach makes it possible to easily compare systems with different configurations by testing various kernel combinations. To examine the influence of POS tags, two comparative experiments were undertaken. The first experiment uses only the POS kernel, i.e. the POS labels are the only feature visible to the learning and classification algorithm. In a second experiment, the full system as in (Gliozzo *et al.*, 2005) is tested against replacing the original POS kernel with the unsupervised POS kernel and omitting the POS kernel completely. Table 5.10 summarises the results in

System	only POS	full
no POS	N/A	0.717
supervised POS	0.629	0.733
unsupervised POS	0.633	0.735

Table 5.10: Comparative evaluation on Senseval scores for WSD. All differences are not significant at $p<0.1$

terms of Senseval scores for WSD, tested on the lexical sample task for English. The unsupervised POS annotation was created using the BNC tagger model, see Section 5.2.8.

Results show that POS information is generally contributing to a very small extent to WSD accuracy in the full WSD system. Using the unsupervised POS tagger results in a slight performance increase, improving over the state of the art results in this task, that have been previously achieved with the same system using supervised POS tags. In conclusion, supervised tagging can safely be exchanged in kernel WSD with the unsupervised variant. Replacing the only pre-processing step that is dependent on manual resources in the system of Gliozzo *et al.* (2005), state of the art supervised WSD is proven to not being dependent on any linguistic pre-processing at all.

Unsupervised POS for NER and Chunking

Named entity recognition (NER) is the task of finding and classifying named entities, such as persons, organisations and locations. Chunking is concerned with shallow syntactic annotation; here, words in a text are labelled as being syntactically correlated, e.g. in noun phrases, verb phrases and prepositional phrases. For performing NER and chunking, these applications are perceived as a tagging task: in each case, labels from a training set are learned and applied to unseen examples. In the NER task, these labels mark named entities and non-named entities, in the chunking task, the respective phrases or chunks are labelled.

For both tasks, the MALLET tagger (McCallum, 2002) is trained. It is based on first-order Conditional Random Fields (CRFs), which define a conditional probability distribution over label sequences given a particular observation sequence. The flexibility of CRFs to include arbitrary, non-independent features makes it easy to supply either standard POS tags, unsupervised POS tags or no POS tags to the system without changing its overall architecture.

Category	PER	ORG	LOC	MISC	ALL
no POS	0.8084	0.7445	0.8151	0.7462	0.7781
supervised POS	0.8154	0.7418	0.8156	0.7660	0.7857
unsupervised POS	0.8083	0.7357	0.8326	0.7527	0.7817

Table 5.11: Comparative evaluation of NER on the Dutch CoNLL-2002 dataset in terms of F1 for PERson, ORGanisation, LOCation, MISCellaneous, ALL. No differences are significant with p<0.1

The tagger operates on a different set of features for the two tasks. In the NER system, the following features are accessible, time-shifted by -2, -1, 0, 1, 2:

- the word itself
- its POS-tag
- Orthographic predicates
- Character bigram and trigram predicates

In the case of chunking, features are only time-shifted by -1, 0, 1 and consist only of:

- Word itself
- POS-tag

This simple feature set for chunking was chosen to obtain a means of almost direct comparison of the different POS schemes without blurring results by other features or system components. Per system, three experiments were carried out, using standard POS features, unsupervised POS features and no POS features.

To evaluate the performance on NER, the methodology as proposed by the providers of the CoNLL-2002 (Roth and van den Bosch, 2002) dataset is adopted: for all settings, the difference in performance in terms of the F1[6] measure is reported. Here, the Dutch dataset is employed, the unsupervised POS tagger is induced on the 70 million token Dutch CLEF corpus (see Peters, 2006). Table 5.11 summarises the results of this experiment for selected categories using the full training set for training and evaluating on the test data.

The scores in Table 5.11 indicate that POS information is hardly contributing anything to the system's performance, be it supervised

[6] F1 is the harmonic mean of precision P (number of correct divided by number of assigned labels) and recall R (number of correct divided by number of all labels), $F1 = \frac{2PR}{P+R}$ (cf. Van Rijsbergen, 1979)

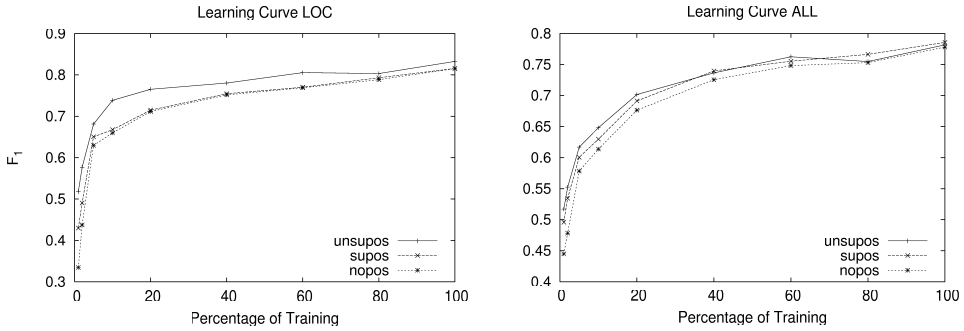


Figure 5.16: Learning curves in NER task for category LOC and combined category

or unsupervised. This indicates that the training set is large enough to compensate for the lack of generalisation when using no POS tags, in line with e.g. (Banko and Brill, 2001). The situation changes when taking a closer look on the learning curve, produced by using train set fractions of differing size. Figure 5.16 shows the learning curves for the categories LOCATION and the (micro average) F1 evaluated over all the categories (ALL).

On the LOCATION category, unsupervised POS tags provide a high generalisation power for a small number of training samples. This is due to the fact that the induced tagset treats locations as a different tag; the tagger's lexicon plays the role of a gazetteer in this case, comprising 765 lexicon entries for the location tag. On the combination of ALL categories, this effect is smaller, yet the incorporation of POS information outperforms the system without POS for small percentages of training.

This disagrees with the findings of Freitag (2004b), where features produced by distributional clustering were used in a boosting algorithm. Freitag reports improved performance on PERSON and ORGANISATION, but not on LOCATION, as compared to not using a tagger at all.

Experiments on NER reveal that POS information is not making a difference, as long as the training set is large enough. For small training sets, usage of unsupervised POS features results in higher performance than supervised or no POS, which can be attributed to its finer-grained tagset that directly indicates types of named entities.

For testing performance of the simple chunking system, different portions of the training set as provided in the English CoNLL-2000 data (see Tjong Kim Sang and Buchholz, 2000) and evaluated on the provided test set. Performance is reported in Figure 5.17.

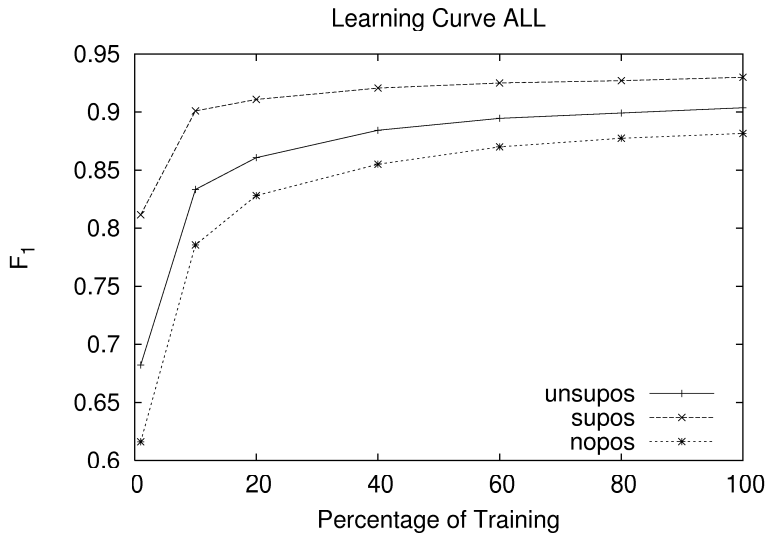


Figure 5.17: Learning curve for the chunking task in terms of F1. Performance at 100% training is 0.882 (no POS), 0.904 (unsupervised POS) and 0.930 (supervised POS), respectively

As POS is the only feature that is used here apart from the word tokens themselves, and chunking reflects syntactic structure, it is not surprising that providing this feature to the system results in increased performance: both kinds of POS significantly outperform not using POS ($p<0.01$). In contrast to the previous systems tested, using the supervised POS labels resulted in significantly better chunking ($p<0.01$) than using the unsupervised labels. This can be attributed to a smaller tagset for supervised POS, providing more reliable statistics because of less sparseness. Further, both supervised tagging and chunking aim at reproducing the same perception of syntax, which does not necessarily fit the distributionally acquired classes of an unsupervised system. Despite the low number of features, the chunking system using supervised tags compares well with the best system in the CoNLL-2000 evaluation (F1=0.9348).

5.2.10 Summary on Unsupervised POS Tagging

An unsupervised POS tagging system was described in detail and evaluated directly and indirectly on various languages and tasks. In difference to previous approaches to unsupervised POS tagging, this method allows for a larger lexicon, where also POS ambiguities are handled. Further, the discovery of the number of POS categories is part of the method, rather than chosen beforehand.

Evaluation on typologically different languages demonstrated the language-independence and robustness of the method. In indirect evaluation it was shown that for many tasks that use POS as a preprocessing step, there is no significant difference in results between using a trained POS tagger or the unsupervised tagger presented here.

As far as performance in applications is concerned, the manual efforts necessary to construct a POS tagger should rather be invested in collecting a large basis of text of the target domain or language, which can be also used for other purposes besides training the unsupervised POS tagger.

A re-implementation of the unsupervised POS tagger system by Andreas Klaus is available for download[7]. This implementation uses the parameter names as given in Table 5.9.

5.3 Word Sense Induction and Disambiguation

To complete this chapter, I will sketch another Structure Discovery process now. Since time and space for a dissertation is limited, this procedure will merely be described and exemplified rather than evaluated. Nevertheless, its proposed architecture might serve as a starting point for further work.

The problem of word sense ambiguity has occurred already several times in this work, see Sections 1.1.4, 4.2.5 and 5.2.9. Now, it will be discussed in more detail.

Major difficulties in automatic language processing are caused by the fact that many words are ambiguous, i.e. they have different meanings in different contexts, but are written (or pronounced) in the same way. While syntactic ambiguities have already been addressed in the previous section, we now turn to the semantic dimension of this problem.

When approaching ambiguity in the Structure Discovery framework, two steps towards the automatic labelling of homonymous words with their senses can be distinguished and correspond to the two directions in Figure 1.1:

- *Word Sense Induction (WSI)* (also word sense discrimination) is the step of identifying the different word senses or usages from corpus data. This is a clustering task.

[7] `http://wortschatz.uni-leipzig.de/~cbiemann/software/unsupos.html` [July 7th, 2007]

- *Word Sense Disambiguation (WSD)* assigns the correct sense from a given set of senses to occurrences of ambiguous words in the text. This is a tagging task.

Both steps have been previously examined extensively. For WSI the general methodology (see e.g. Pedersen and Bruce, 1997; Schütze, 1998; Pantel and Lin, 2002; Widdows and Dorow, 2002; Purandare and Pedersen, 2004; Rapp, 2004; Bordag, 2006) is to cluster the word co-occurrences of a target word to arrive at sets of co-occurrences that represent the different senses, with some variety regarding the context window size, dimensionality reduction techniques and the order of co-occurrences. WSD has been predominantly performed using a pre-defined sense inventory like WordNet, see (Agirre and Edmonds, 2006) and the Senseval competitions (e.g. Mihalcea and Edmonds, 2004) for an overview. Here, the WordNet sense is either assigned using lexical overlap between the dictionary definition and the actual context, or by training a classifier on the hand-labelled SemCor corpus[8].

A problem with pre-defined word sense inventories is that their senses often do not match the application domain, and there is no consensus on the granularity level of senses, even within single sense inventories. Kilgarriff (1997) states "that word senses are only ever defined relative to a set of interests. The set of senses defined by a dictionary may or may not match the set that is relevant for an NLP application". Taking the dissent among lexicographers and the fuzziness of the term 'word sense' into account, Cohn (2003) even considers the task of WSD as being ill-defined. Therefore, a domain- and corpus-specific word sense inventory is, besides being the only possibility when working in SD, also highly desired independent of the processing paradigm. This is also reflected in the study of Schütze and Pedersen (1995), one of the rare studies where WSD showed to have a positive effect on information retrieval when using senses induced from a corpus: most other studies (Voorhees, 1993, inter al.) reported negative effects when applying WSD with a predefined sense inventory in a search context.

The study of Agirre *et al.* (2006a) proceeds just in the spirit of SD to evaluate the contribution of HyperLex graph clustering (Veronis (2004), see Section 4.1.3) in the following way: from an unannotated corpus, a sense inventory is induced that is used to assign sense labels for words in a sense-tagged corpus. These are employed to create a mapping between the annotated and the induced senses. A held-out sense-tagged corpus serves as the basis for evaluating the

[8] available at `http://multisemcor.itc.it/semcor.php` [June 1st, 2007]

quality of the induced annotation via the mapping and was used to optimise the parameters of HyperLex. Interestingly, the best results were obtained with a high granularity of microsenses, rather than with coarse-grained distinctions that are supported by a broader data basis per sense. This suggests two things: first, the manually assigned senses and the automatically induced senses do not match well; this is why a finer-grained sense induction produces a more pure mapping. Second, the data basis for microsenses is sufficently large to assign them with an accuracy that almost reaches state of the art performance for unsupervised WSD systems despite the lossy mapping. When comparing HyperLex to PageRank (Brin and Page, 1998) for selecting root vertices in (Agirre *et al.*, 2006b), very similar results suggest that the key issue in graph-based WSI is not so much the clustering algorithm used, but rather the construction of the underlying graphs.

This is also supported by comparing the performance of Chinese Whispers to the triplet-based WSI graph clustering algorithm of Bordag (2006) as undertaken in (Biemann, 2006b)[9]: here, the sentence-based word co-occurrence graphs of 45 test words of different frequency and parts-of-speech are conflated pair-wise in a pseudoword evaluation fashion (cf. Schütze, 1998). It was measured, to which extent the clustering method could restore the original word clusters. Since these words were selected to be not ambiguous, a perfect clustering method would cluster the conflated word co-occurrence graphs in two clusters corresponding to the original graphs of the single words.

To quantify the extent to which this succeeds, four measures are proposed in (Bordag, 2006):

- *retrieval Precision* (rP) - the percentage of overlap of the found sense co-occurrences with the original sense co-occurrences.
- *retrieval Recall* (rR) - the amount of words that have been correctly retrieved into the correct sense.
- *Precision* (P) is defined as the number of times the original co-occurrence sets are properly restored divided by the number of different sets found. Properly restored here means that rP for the correct sense is above 60%.
- *Recall* (R) is defined as the number of senses found divided by the number of words merged to create the pseudoword. For this, rR for the correct sense must be above 25%.

[9] Thanks goes to Stefan Bordag for providing his evaluation framework

Method	mix	standard CW				(Bordag, 2006)			
Measure		*P*	*R*	*rP*	*rR*	*P*	*R*	*rP*	*rR*
samePOS	N	**90.0**	79.5	**94.8**	**71.3**	87.0	**86.7**	90.9	64.2
	V	77.6	67.1	**87.3**	57.9	78.3	64.3	80.2	55.2
	A	**92.2**	61.9	89.3	**71.9**	88.6	**71.0**	88.0	65.4
diffPOS	N/V	81.8	77.3	**94.0**	62.0	**86.6**	77.1	90.5	61.9
	N/A	89.3	75.1	**94.3**	**70.1**	90.9	78.0	90.4	66.8
	V/A	83.4	**68.9**	**90.0**	61.5	80.9	63.6	82.0	60.9
sameFreq	high	93.7	72.9	**95.0**	73.8	93.7	**78.1**	90.3	**80.7**
	med.	80.7	83.8	91.0	55.7	**84.5**	85.2	89.9	54.6
	low	74.1	51.4	72.9	**56.2**	74.8	49.5	71.0	41.7
diffFreq	h/m	**90.0**	70.4	**96.7**	**75.4**	86.4	**79.6**	92.7	72.1
	h/l	93.1	69.6	**97.1**	**78.3**	91.2	67.8	90.9	74.5
	m/l	76.2	72.2	87.7	52.4	**82.3**	74.0	85.3	49.9

Table 5.12: Comparative evaluation of WSI for standard Chinese Whispers and (Bordag, 2006) on the same dataset in average %. Combinations were conducted for all testwords with the same syntactic class (samePOS) for nouns (N), verbs (V) and adjectives (A), of different syntactic class (diffPOS), of same frequency range (sameFreq) and of different frequency ranges (diffFreq). Per category, the better figure is marked in bold, if the difference exceeds 3%

In the experiment, the open neighbourhoods from the significant word co-occurrence graph ($t = 2$, $s = 15$, cf. Section 3.2.1) of the raw BNC for each target word were merged by twos, and standard Chinese Whispers was used to cluster these merged graphs. Table 5.12 compares the performance of the CW clustering with the results of Bordag (2006).

Results in Table 5.12 suggest that both algorithms arrive at about equal overall performance (P and R). Chinese Whispers clustering is able to capture the same information as a specialised graph-clustering algorithm for WSI, given the same input. The slightly superior performance on rR and rP indicates that CW leaves fewer words unclustered, which can be advantageous when using the clusters as clues in word sense disambiguation. However, pseudoword evaluation and the restriction to only a few sample words make it difficult to draw conclusions about WSD performance from these experiments.

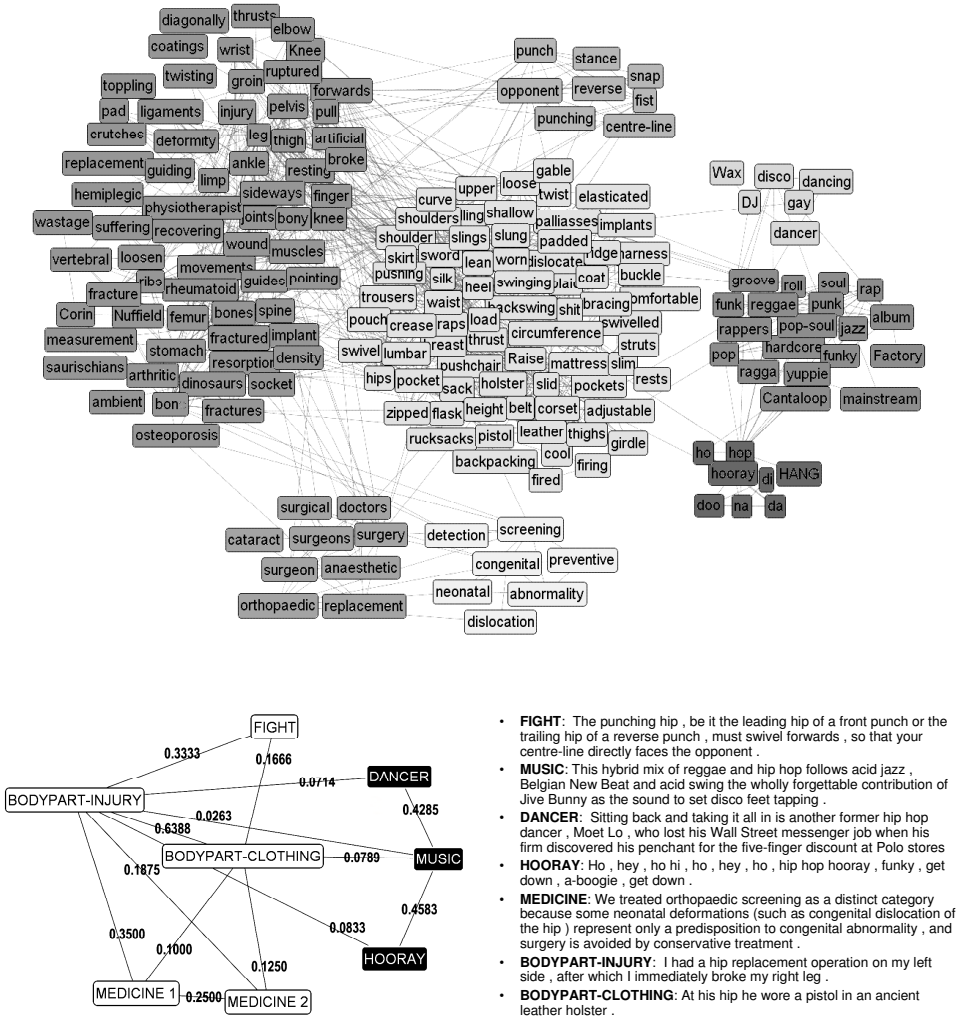
The following suggestions might be worth trying for improving the performance of unsupervised WSI+WSD:

- *Select and Re-train*: When assigning the induced word senses to tokens in the text, it might be advantageous to only carry out the assignment on those tokens that can be resolved with high confidence. These can further be used in a supervised WSD scenario (e.g. the kernel method described in Section 5.2.9) for training a model that is able to assign senses to the remaining tokens.

- *Flat Hierarchy of Senses*: To arrive at a generic method that combines the advantages of fine- and coarse-grained granularities of word senses, it is possible to set up a flat hierarchy when inducing the word senses and to disambiguate for each hierarchy level separately. In an application scenario, machine learning algorithms with appropriate feature selection mechanisms can pick the appropriate granularity level for the given task.

To illustrate the latter point, consider Figure 5.18, where the neighbourhood of 'hip' in the significant sentence-based word co-occurrence graph of the BNC (excluding high frequency words) was clustered with hierarchical agglomerative Chinese Whispers as outlined in Section 4.2.7. On the first level, senses are very fine-grained, whereas on the second level, the two usages of 'hip' as *body part* and in *music* can be identified.

In the WSD step, attention has to be paid to the fact that the context words used for overlap computation can themselves be ambiguous. For this, it might be advisable to disambiguate the graph as outlined in Section 4.2.5 and to apply a ranking mechanism as in Mihalcea *et al.* (2004a) to assign the most probable combination of senses.



- **FIGHT**: The punching hip , be it the leading hip of a front punch or the trailing hip of a reverse punch , must swivel forwards , so that your centre-line directly faces the opponent .
- **MUSIC**: This hybrid mix of reggae and hip hop follows acid jazz , Belgian New Beat and acid swing the wholly forgettable contribution of Jive Bunny as the sound to set disco feet tapping .
- **DANCER**: Sitting back and taking it all in is another former hip hop dancer , Moet Lo , who lost his Wall Street messenger job when his firm discovered his penchant for the five-finger discount at Polo stores
- **HOORAY**: Ho , hey , ho hi , ho , hey , ho , hip hop hooray , funky , get down , a-boogie , get down .
- **MEDICINE**: We treated orthopaedic screening as a distinct category because some neonatal deformations (such as congenital dislocation of the hip) represent only a predisposition to congenital abnormality , and surgery is avoided by conservative treatment .
- **BODYPART-INJURY**: I had a hip replacement operation on my left side , after which I immediately broke my right leg .
- **BODYPART-CLOTHING**: At his hip he wore a pistol in an ancient leather holster .

Figure 5.18: WSI Example of 'hip' with hierarchical agglomerative CW and usages. Top: fist level clustering. Bottom left: second-level clustering. Bottom right: Sample usages from the BNC

6 Conclusion

In this dissertation I introduced the Structure Discovery paradigm, which finds and annotates structure in natural language data using Structure Discovery processes. Structure is to be understood here as any kind of annotation that can be automatically introduced into language data by applying algorithms that employ regularities and make these explicit. The algorithms operate without any linguistic information – neither explicitly stated facts about language, nor implicitly encoded knowledge in the form of manual annotations.

The Structure Discovery paradigm was contrasted to the predominant schools of language processing and its adequacy for linguistic phenomena was motivated. A major contribution of this work is the definition of a framework for processing language in an unsupervised and knowledge-free way.

Although there has been previous work that can be attributed to this paradigm, there has not been an attempt to completely describe it before. The main advantages of telling the machine how to discover structure in language is the language- and domain-independence of this approach, which strives at a uniform treatment by employing language universals and reduces the manual effort of encoding language-specific knowledge to zero.

When working in Structure Discovery, the first step is to characterise the data that is subject to exploration. At this point, the characteristics are also depending on the representation of the data. Here, graph representations allow to model statistically gathered information about language units in an intuitive way: entities of language are represented by vertices; edge weights denote the degree of association between entities.

It has been known before that many graphs built on language as well as on other data exhibit the scale-free Small World property. After reviewing random graph models and finding that none of the existing models are able to reproduce the characteristics found in word co-occurrence networks, a random text model was presented that comes closer to what is observed in natural language texts. This is reached by simulating the generation of words and sentences in two different modules. Further, the rank-frequency distribution of words and language unit length distributions of natural language could be

reproduced. This random text generation model is a another step towards explaining the mechanisms that point to the origin of language by defining simple procedures that produce an output stream that quantitatively resembles language.

Clustering methods are the only choice to perform the abstractions and generalisations needed for assigning structure to previously unstructured text in absence of pre-classified items. A review of clustering methods in general and graph clustering methods in particular showed that most previous approaches are not suitable for natural language data, as they can either not cope with highly skewed distributions or fail to present tractable solutions for large datasets.

Thus, the Chinese Whispers graph clustering algorithm was developed and tested on artificial and real datasets. By working in a decentralised fashion and only with local information, this randomised algorithm is as efficient as any method that takes the full graph into account. This allows the processing of very large graphs, which are common in language processing when e.g. encoding words as vertices and their similarities as edges. Several possible extensions and modifications allow adapting the method to special requirements of datasets. A central advantage of this method is that the number of clusters is found automatically rather than provided by the user, which is especially important for language data, where e.g. the number of word senses or the number of languages in a random web sample are not known a priori.

In the practical part of this work, two Structure Discovery processes were set up and thoroughly evaluated.

A language separation algorithm performs almost perfectly in sorting a multilingual text corpus into monolingual chunks. The number and the size distribution of the involved languages are found by the method itself, which renders prior knowledge obsolete for this task.

An unsupervised part-of-speech tagger, which induces and provides annotation with syntactic-semantic word classes, was tested extensively for a variety of languages against manually annotated resources and in applications, where the performance gains are similar to using a traditional tagger. This makes the manual annotation of parts-of-speech information superfluous for application-based settings.

A further process for word sense induction and disambiguation was outlined, which illustrates possible future work in the Structure Discovery paradigm.

The feasibility of the broad-scale program of this work – the fully

unsupervised and knowledge-free processing of natural language text – has been exemplified on various language processing tasks. It could be demonstrated that in fact Structural Discovery processes can produce structural information that is useful for natural language processing, and are a veritable alternative whenever high manual efforts are not affordable. The presented methods and algorithms serve only as a starting point – further Structure Discovery processes might be the topic of future work.

The gain for automatic natural language processing is twofold: While Structure Discovery provides a cheap and robust way to rapidly improve the processing for resource-scarce languages and domains, it also allows insights in the mechanisms of natural language per se.

Bibliography

Abney, S. (1996). Statistical methods and linguistics. In J. Klavans and P. Resnik, editors, *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pages 1–26. The MIT Press, Cambridge, Massachusetts.

Adamic, L. A. (2000). Zipf, power-law, pareto – a ranking tutorial. Technical report, Information Dynamics Lab, HP Labs, HP Labs, Palo Alto, CA 94304.

Agirre, E. and Edmonds, P., editors (2006). *Word Sense Disambiguation: Algorithms and Applications*. Text, Speech and Language Technology. Springer.

Agirre, E., Martínez, D., López de Lacalle, O., and Soroa, A. (2006a). Evaluating and optimizing the parameters of an unsupervised graph-based wsd algorithm. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, pages 89–96, New York City. Association for Computational Linguistics.

Agirre, E., Martínez, D., López de Lacalle, O., and Soroa, A. (2006b). Two graph-based algorithms for state-of-the-art WSD. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-06)*, pages 585–593, Sydney, Australia. Association for Computational Linguistics.

Aiello, W., Chung, F., and Lu, L. (2000). A random graph model for massive graphs. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 171–180, New York, NY, USA. ACM Press.

Albert, R., Jeong, H., and Barabási, A.-L. (2000). Error and attack tolerance of complex networks. *Nature*, **406**, 378.

Allauzen, C., Mohri, M., and Roark, B. (2003). Generalized algorithms for constructing statistical language models. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL-03)*, pages 40–47, Morristown, NJ, USA. Association for Computational Linguistics.

Amaral, L. A., Scala, A., Barthelemy, M., and Stanley, H. E. (2000). Classes of small-world networks. *Proc. Natl. Acad. Sci. USA*, **97**(21).

Amblard, F. (2002). Which ties to choose? A survey of social networks models for agent-based social simulations. In *Proceedings of the 2002 SCS International Conference On Artificial Intelligence, Simulation and Planning in High Autonomy Systems*, pages 253–258.

Baeza-Yates, R. and Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. Addison Wesley.

Baker, C., Fillmore, C., and Cronin, B. (2003). The structure of the FrameNet database. *International Journal of Lexicography*, **16**(3), 281–296.

Banko, M. and Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of ACL-01*, pages 26–33.

Barabási, A.-L. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, **286**, 509.

Barrat, A., Barthelemy, M., Pastor-Satorras, R., and Vespignani, A. (2004a). The architecture of complex weighted networks. *Proc. Natl. Acad. Sci. USA*, **101**, 37–47.

Barrat, A., Barthelemy, M., and Vespignani, A. (2004b). Weighted evolving networks: coupling topology and weights dynamics. *Physical Review Letters*, **92**.

Berkhin, P. (2002). Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA.

Berland, M. and Charniak, E. (1999). Finding parts in very large corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics (ACL-99)*, pages 57–64, Morristown, NJ, USA. Association for Computational Linguistics.

Bern, M. and Eppstein, D. (1997). Approximation algorithms for geometric problems. In *Approximation algorithms for NP-hard problems*, pages 296–345. PWS Publishing Co., Boston, MA, USA.

Biemann, C. (2006a). Bootstrapping. In G. Heyer, U. Quasthoff, and T. Wittig, editors, *Wissensrohstoff Text*, pages 260–266. W3L, Bochum.

Biemann, C. (2006b). Chinese Whispers – an Efficient Graph Clustering Algorithm and its Application to Natural Language Processing Problems. In *Proceedings of TextGraphs: the Second Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York City. Association for Computational Linguistics.

Biemann, C. (2006c). Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the COLING/ACL-06 Student Research Workshop*, Sydney, Australia.

Biemann, C. (2007). A random text model for the generation of statistical language invariants. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference (HLT-NAACL-07)*, pages 105–112, Rochester, New York. Association for Computational Linguistics.

Biemann, C. and Quasthoff, U. (2005). Dictionary acquisition using parallel text and co-occurrence statistics. In *Proceedings of NODALIDA'05*, Joensuu, Finland.

Biemann, C. and Quasthoff, U. (2007). Similarity of documents and document collections using attributes with low noise. In *Proceedings of the Third International Conference on Web Information Systems and Technologies (WEBIST-07)*, pages 130–135, Barcelona, Spain.

Biemann, C. and Teresniak, S. (2005). Disentangling from babylonian confusion – unsupervised language identification. In *Proceedings of Computational Linguistics and Intelligent Text Processing, 6th International Conference (CICLing-05)*, Springer LNCS, pages 773–784, Mexico D.F., Mexico.

Biemann, C., Bordag, S., and Quasthoff, U. (2004a). Automatic acquisition of paradigmatic relations using iterated co-occurrences. In *Proceedings of the fourth international conference on Language Resources and Evaluation (LREC-04)*, Lisbon, Portugal.

Biemann, C., Böhm, C., Heyer, G., and Melz, R. (2004b). Automatically building concept structures and displaying concept trails for the use in brainstorming sessions and content management systems. In *Proceedings of Innovative Internet Community Systems (IICS-2004)*, Springer LNCS, Guadalajara, Mexico.

Biemann, C., Shin, S.-I., and Choi, K.-S. (2004c). Semiautomatic extension of corenet using a bootstrapping mechanism on corpus-based co-occurrences. In *Proceedings of the 20th international conference on Computational Linguistics (COLING-04)*, Morristown, NJ, USA. Association for Computational Linguistics.

Biemann, C., Giuliano, C., and Gliozzo, A. (2007). Unsupervised part-of-speech tagging supporting supervised methods. In *Proceedings of Recent Advances in Natural Language Processing (RANLP-07)*, Borovets, Bulgaria.

Bod, R. (2006). An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL (COLING/ACL-06)*, pages 865–872, Morristown, NJ, USA. Association for Computational Linguistics.

Bollobas, B. (1985). *Random Graphs*. Academic Press.

Bollobas, B. (1998). *Modern Graph Theory*. Springer-Verlag.

Bollobas, B., Riordan, O., Spencer, J., and Tusnady, G. (2001). The degree sequence of a scale-free random graph process. *Random Structures and Algorithms*, **18**(3).

Bonato, A. (2005). A survey of models of the web graph. *Combinatorial and Algorithmic Aspects of Networking*, pages 159–172.

Bordag, S. (2006). Word sense induction: Triplet-based clustering and automatic evaluation. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*, Trento, Italy.

Bordag, S. (2007). *Elements of Knowledge-free and Unsupervised Lexical Acquisition*. Ph.D. thesis, University of Leipzig.

Brandes, U., Gaertler, M., and Wagner, D. (2003). Experiments on graph clustering algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, pages 568–579. Springer.

Brants, T. (2000). TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on Applied Natural Language Processing (ANLP-00))*, pages 224–231, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Brants, T., Hendriks, R., Kramp, S., Krenn, B., Preis, C., Skut, W., and Uszkoreit, H. (1997). Das NEGRA-Annotationsschema. Negra project report, Universität des Saarlandes, Saarbrücken.

Brill, E. (1992). A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied Natural Language Processing (ANLP-92))*, pages 152–155, Morristown, NJ, USA. Association for Computational Linguistics.

Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, **30**(1-7), 107–117.

Brown, P. F., Pietra, V. J. D., deSouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based n-gram models of natural language. *Computational Linguistics*, **18**(4), 467–479.

Burnard, L. (1995). Users reference guide for the british national corpus. Oxford University Computing Service, Oxford, U.K.

Cardie, C. and Weischedel, R., editors (1997). *A Corpus-Based Approach for Building Semantic Lexicons*, Somerset, New Jersey. Association for Computational Linguistics.

Cavnar, W. B. and Trenkle, J. M. (1994). N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

Chakrabarti, D. and Faloutsos, C. (2006). Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, **38**(1).

Charniak, E., Hendrickson, C., Jacobson, N., and Perkowitz, M. (1993). Equations for part-of-speech tagging. In *National Conference on Artificial Intelligence*, pages 784–789.

Chomsky, N. (1957). *Syntactic Structures*. Mouton, The Hague.

Chomsky, N. (1964). *Current Issues in Linguistic Theory*. Mouton, The Hague.

Chung, F. R. (1997). Spectral graph theory. *Regional Conference Series in Mathematics*, **92**.

Church, K. W. (2000). Empirical estimates of adaptation: the chance of two noriegas is closer to p/2 than p^2. In *Proceedings of the 18th conference on Computational Linguistics (COLING-00)*, pages 180–186, Morristown, NJ, USA. Association for Computational Linguistics.

Clark, A. (2000). Inducing syntactic categories by context distribution clustering. In C. Cardie, W. Daelemans, C. Nédellec, and E. T. K. Sang, editors, *Proceedings of the Fourth Conference on Computational Natural Language Learning and of the Second Learning Language in Logic Workshop, Lisbon, 2000*, pages 91–94. Association for Computational Linguistics, Somerset, New Jersey.

Clark, A. (2003). Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics (EACL-03)*, pages 59–66, Morristown, NJ, USA. Association for Computational Linguistics.

Cohen, P. R., Adams, N., and Heeringa, B. (2007). Voting experts: An unsupervised algorithm for segmenting sequences. *International Journal on Intelligent Data Analysis*, **11**.

Cohn, H., Kleinberg, R., Szegedy, B., and Umans, C. (2005). Group-theoretic algorithms for matrix multiplication. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 379–388, Washington, DC, USA. IEEE Computer Society.

Cohn, T. (2003). Performance metrics for word sense disambiguation. In *Proceedings of the Australasian Language Technology Workshop, 2003*, pages 49–56.

Cruse, D. A. (2002). Aspects of the micro-structure of word meanings. In Y. Ravin and C. Leacock, editors, *Polysemy: Theoretical and Computational Approaches*. Oxford. University Press.

Cucerzan, S. and Yarowsky, D. (1999). Language independent named entity recognition combining morphological and contextual evidence. In *Proceedings of 1999 Joint SIGDAT Conference on EMNLP and VLC .*, pages 132–138, College Park.

Cvetkovič, D., Doob, M., and Sachs, H. (1995). *Spectra of Graphs: Theory and Application, 3rd Edition*. Johann Ambrosius Barth.

Cysouw, M., Biemann, C., and Ongyerth, M. (2007). Using strong's numbers in the bible to test an automatic alignment of parallel texts. *Special issue of Sprachtypologie und Universalienforschung (STUF)*, pages 66–79.

Davidov, D. and Rappoport, A. (2006). Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL-06)*, pages 297–304, Sydney, Australia. Association for Computational Linguistics.

Davidov, D., Rappoport, A., and Koppel, M. (2007). Fully unsupervised discovery of concept-specific relationships by web mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07)*, pages 232–239, Prague, Czech Republic. Association for Computational Linguistics.

de Saussure, F. (1966). *Course in General Linguistics*. New York: McGraw-Hill.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, **41**(6), 391–407.

Deo, N. and Gupta, P. (2001). World wide web: a graph-theoretic perspective. Technical report, Comp. Sci. Tech. Report CS-TR-01-001, University of Central Florida.

Dhillon, I. S., Mallela, S., and Modha, D. S. (2003). Information-theoretic co-clustering. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD-2003)*, pages 89–98.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, **1**, 269–271.

Dorogovtsev, S. N. and Mendes, J. F. F. (2001a). Language as an evolving word web. *Proceedings of The Royal Society of London. Series B, Biological Sciences*, **268**(1485), 2603–2606.

Dorogovtsev, S. N. and Mendes, J. F. F. (2001b). Scaling properties of scale-free evolving networks: Continuous approach. *Physical Review E*, **63**.

Dunning, T. (1994). Statistical identification of language. Techical Report MCCS-94-273, Computing Research Lab (CRL), New Mexico State University.

Dunning, T. E. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, **19**(1), 61–74.

Eiken, U. C., Liseth, A. T., Witschel, H. F., Richter, M., and Biemann, C. (2006). Ord i dag: Mining Norwegian daily newswire. In *Proceedings of the FinTAL*, Turku, Finland.

Erdős, P. and Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci.*, **5**, 17–61.

Erkan, G. and Radev, D. R. (2004). Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research (JAIR)*, **22**, 457–479.

Ertöz, L., Steinbach, M., and Kumar, V. (2002). A new shared nearest neighbor clustering algorithm and its applications. In *Proceedings of Workshop on Clustering High Dimensional Data and its Applications*, pages 105–115.

Ester, M., Kriegel, H. P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, Portland, Oregon. AAAI press.

Everitt, B. S., Landau, S., and Leese, M. (2001). *Cluster analysis (4th edition)*. Arnold.

Evert, S. (2004). *The Statistics of Word Co-occurrences: Word Pairs and Collocations*. Ph.D. thesis, University of Stuttgart.

Ferrer-i-Cancho, R. and Solé, R. V. (2001). The small world of human language. *Proceedings of The Royal Society of London. Series B, Biological Sciences*, **268**(1482), 2261–2265.

Ferrer-i-Cancho, R. and Solé, R. V. (2002). Zipf's law and random texts. *Advances in Complex Systems*, **5**(1), 1–6.

Finch, S. and Chater, N. (1992). Bootstrapping syntactic categories using statistical methods. In *Background and Experiments in Machine Learning of Natural Language: Proceedings of the 1st SHOE Workshop*, pages 229–235. Katholieke Universiteit, Brabant, Holland.

Firth, J. R. (1957). *A Synopsis of Linguistic Theory, 1933-1955*. Blackwell, Oxford.

Fjällström, P. O. (1998). Algorithms for graph partitioning: A survey. *Linkoping Electronic Articles in Computer and Information Science*, **3**.

Flake, G. W., Tarjan, R. E., and Tsioutsiouliklis, K. (2004). Graph clustering and minimum cut trees. *Internet Mathematics*, **1**(4), 385–408.

Ford, L. R. and Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, **8**, 399–404.

Francis, W. N. and Kučera, H. (1982). *Frequency Analysis of English Usage*. Houghton Mifflin, Boston.

Freitag, D. (2004a). Toward unsupervised whole-corpus tagging. In *Proceedings of the 20th international conference on Computational Linguistics (COLING-04)*, page 357, Morristown, NJ, USA. Association for Computational Linguistics.

Freitag, D. (2004b). Trained named entity recognition using distributional clusters. In *Proceedings of EMNLP-04*, pages 262–269.

Garside, R., Leech, G., and Sampson, G. (1987). *The computational analysis of English: a corpus-based approach*. Longman.

Gauch, S. and Futrelle, R. (1994). Experiments in Automatic Word Class and Word Sense Identification for Information Retrieval. In *Proceedings of the 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 425–434, Las Vegas, NV.

Gibbons (1985). *Algorithmic Graph Theory*. Cambridge University Press.

Glassman, S. (1994). A caching relay for the world wide web. *Computer Networks and ISDN Systems*, **27**(2), 165–173.

Gliozzo, A. M. (2005). *Semantic Domains in Computational Linguistics*. Ph.D. thesis, University of Trento, Italy.

Gliozzo, A. M., Giuliano, C., and Strapparava, C. (2005). Domain kernels for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, pages 403–410, Ann Arbor, Michigan, USA.

Grefenstette, G. (1995). Comparing two language identification schemes. In *3rd International Conference on Statistical Analysis of Textual Data*, pages 263–268, Rome, Italy.

Grewendorf, G., Hamm, F., and Sternefeld, W. (1987). *Sprachliches Wissen*. Suhrkamp, Frankfurt/Main.

Ha, L. Q., Sicilia-Garcia, E. I., Ming, J., and Smith, F. J. (2002). Extension of Zipf's law to words and phrases. In *Proceedings of the 19th international conference on Computational Linguistics (COLING-02)*, pages 1–6, Morristown, NJ, USA. Association for Computational Linguistics.

Hagen, K., Johannessen, J. B., and Nøklestad, A. (2000). A constraint-based tagger for Norwegian. *Lindberg, C.-E. og S. Nordahl Lund (red.): 17th Scandinavian Conference of Linguistics, vol. I. Odense: Odense Working Papers in Language and Communication*, **I**(19).

Haghighi, A. and Klein, D. (2006). Prototype-driven learning for sequence models. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL-06)*, New York, NY, USA.

Haghighi, A. and Klein, D. (2007). Unsupervised coreference resolution in a nonparametric bayesian model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07)*, pages 848–855, Prague, Czech Republic. Association for Computational Linguistics.

Harris, Z. S. (1951). *Methods in Structural Linguistics*. University of Chicago Press, Chicago.

Harris, Z. S. (1968). *Mathematical Structures of Language*. Wiley.

Hauck, S. and Borriello, G. (1995). An evaluation of bipartitioning techniques. In *ARVLSI '95: Proceedings of the 16th Conference on Advanced Research in VLSI (ARVLSI'95)*, page 383, Washington, DC, USA. IEEE Computer Society.

Heyer, G., Quasthoff, U., and Wittig, T. (2006). *Wissensrohstoff Text*. W3L, Bochum, Bochum, Germany.

Holz, F., Witschel, H., Heinrich, G., Heyer, G., and Teresniak, S. (2007). An evaluation framework for semantic search in p2p networks. Technical report, University of Leipzig.

Horn, G. M. (1983). *Lexical-Functional Grammar*. Mouton de Gruyter, Berlin.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, **31**(3), 264–323.

Jäger, G. (2003). Evolutionary game theory and linguistic typology: A case study. In P. Dekker, editor, *Proceedings of the 14th Amsterdam Colloquium*, ILLC, University of Amsterdam.

Johnson, S. (1993). Solving the problem of language recognition. Technical report, School of Computer Studies, University of Leeds, UK.

Kannan, R., Vempala, S., and Vetta, A. (2000). On clusterings: Good, bad, and spectral. In *Proceedings of the 41st Annual Symposium on the Foundation of Computer Science*, pages 367–380. IEEE Computer Society.

Kanter, I. and Kessler, D. A. (1995). Markov processes: Linguistics and Zipf's law. *Physical Review Letters*, **74**(22), 4559–4562.

Karger, D. R. (1993). Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *SODA*, pages 21–30.

Karger, D. R. (1996). Minimum cuts in near-linear time. In *STOC '96: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 56–63, New York, NY, USA. ACM Press.

Kay, M. (1997). Machine translation: the disappointing past and present. In *Survey of the state of the art in human language technology*, pages 248–250. Cambridge University Press, New York, NY, USA.

Keller, E. F. (2005). Revisiting "scale-free" networks. *Bioessays*, **27**(10), 1060–1068.

Kilgarriff, A. (1997). I don't belive in word senses. *Computers and the Humanities*, **31**(2), 91–113.

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

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

Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. S. (1999). The Web as a graph: Measurements, models and methods. In *Proceedings of the International Conference on Combinatorics and Computing*, pages 1–17, Singapore.

Knuth, D. E. (1998). *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, **97**(1-2), 273–324.

Kumar, S. R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999). Extracting large-scale knowledge bases from the web. In *The VLDB Journal*, pages 639–650.

Kurimo, M., Creutz, M., Varjokallio, M., Arisoy, E., and Saraclar, M. (2006). Unsupervised segmentation of words into morphemes – challenge 2005. an introduction and evaluation report. In *Proceedings of the PASCAL Challenges Workshop on Unsupervised Segmentation of Words into Morphemes*, Venice, Italy.

Ladefoged, P. and Maddieson, I. (1996). *The Sounds of the World's Languages*. Blackwell Publishers, Oxford, UK.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning (ICML-01)*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Lempel, R. and Moran, S. (2003). Predictive caching and prefetching of query results in search engines. In *Proceedings of the 12th international conference on World Wide Web (WWW-03)*, pages 19–28, New York, NY, USA. ACM Press.

Levin, E., Sharifi, M., and Ball, J. (2006). Evaluation of utility of LSA for word sense discrimination. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 77–80, New York City, USA. Association for Computational Linguistics.

Li, C. and Chen, G. (2003). Network connection strengths: Another power-law? Technical report, cond-mat/0311333, ArXiv, 2003.

Li, W. (1992). Random texts exhibit Zipf's law-like word frequency distribution. *IEEETIT: IEEE Transactions on Information Theory*, **38**.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, volume I*, pages 281–297. Berkeley University of California Press.

Mahn, M. and Biemann, C. (2005). Tuning co-occurrences of higher orders for generating ontology extension candidates. In *Proceedings of the ICML-05 Workshop on Ontology Learning and Extension using Machine Learning Methods*, Bonn, Germany.

Mandelbrot, B. B. (1953). An information theory of the statistical structure of language. In *Proceedings of the Symposium on Applications of Communications Theory*. Butterworths.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts.

Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, **8**(1), 3–30.

Matsuo, Y. (2002). Clustering using small world structure. In *Proceedings of the 6th International Conference on Knowledge-based Intelligent Information Engineering Systems and Applied Technologies (KES-02)*, pages 1257–1261, Crema, Italy. IOS Press/Ohmsha.

McCallum, A. K. (2002). Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

McNamee, P. (2005). Language identification: a solved problem suitable for undergraduate instruction. *Journal of Computing Sciences in Colleges*, **20**(3), 94–101.

Mehler, A. (2007). Evolving lexical networks. a simulation model of terminological alignment. In A. Benz, C. Ebert, and R. van Rooij, editors, *Proceedings of the Workshop on Language, Games, and Evolution at the 9th European Summer School in Logic, Language and Information (ESSLI 2007), Trinity College, Dublin, 6-17 August*.

Meilă, M. (2005). Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd international conference on Machine Learning (ICML-05)*, pages 577–584, New York, NY, USA. ACM Press.

Meilă, M. and Shi, J. (2000). Learning segmentation by random walks. In *Neural Information Processing Systems (NIPS)*, pages 873–879, Breckenridge, CO (USA).

Merris, R. (2001). *Graph Theory*. John Wiley.

Meyer, H. (1967). *Deutsche Sprachstatistik*. Georg Olms Verlagsbuchhandlung.

Mihail, M., Gkantsidis, C., Saberi, A., and Zegura, E. (2002). On the semantics of internet topologies. Technical report, Georgia Institute of Technology.

Mihalcea, R. and Edmonds, P., editors (2004). *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*. Association for Computational Linguistics, Barcelona, Spain.

Mihalcea, R. and Tarau, P. (2004). Textrank: Bringing order into texts. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 404–411, Barcelona, Spain.

Mihalcea, R., Tarau, P., and Figa, E. (2004a). Pagerank on semantic networks, with application to word sense disambiguation. In *Proceedings of the 20th international conference on Computational Linguistics (COLING-04)*, pages 1126–1132, Geneva, Switzerland. COLING.

Mihalcea, R., Chklovsky, T., and Kilgarriff, A. (2004b). The SENSEVAL-3 english lexical sample task. In *Proceedings of SENSEVAL-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 25–28, New Brunswick, NJ, USA.

Milgram (1967). The small world problem. *Psychology Today*, **1**(61).

Miller, G. A. (1957). Some effects of intermittent silence. *American Journal of Psychology*, **70**, 311–313.

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

Miller, G. A., Beckwith, R., Fellbaum, C. D., Gross, D., and Miller, K. (1990). Wordnet: An on-line lexical database. *International Journal of Lexicography 3(4)*, pages 235–244.

Moore, R. C. (2004). On log-likelihood-ratios and the significance of rare events. In D. Lin and D. Wu, editors, *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 333–340, Barcelona, Spain. Association for Computational Linguistics.

Newman, M. E. J., Watts, D. J., and Strogatz, S. H. (2002). Random graph models of social networks. *Proc. Natl. Acad. Sci. USA*, **99**(suppl. 1), 2566–2572.

Nivre, J. (2006). *Inductive Dependency Parsing*, volume 34 of *Text, Speech and Language Technology*. Springer.

Noreen, E. W. (1989). *Computer-Intensive Methods for Testing Hypotheses : An Introduction*. Wiley-Interscience.

Och, F. J. (2005). Statistical machine translation: The fabulous present and future. Invited talk at ACL Workshop on Building and Using Parallel Texts, Ann Arbor, Michigan, USA.

Olney, A. M. (2007). Latent semantic grammar induction: Context, projectivity, and prior distributions. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 45–52, Rochester, NY, USA. Association for Computational Linguistics.

Pantel, P. and Lin, D. (2002). Discovering word senses from text. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619, New York, NY, USA. ACM Press.

Pantel, P., Ravichandran, D., and Hovy, E. (2004). Towards terascale knowledge acquisition. In *Proceedings of the 20th international conference on Computational Linguistics (COLING-04)*, page 771, Morristown, NJ, USA. Association for Computational Linguistics.

Pedersen, T. and Bruce, R. (1997). Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–207, Providence, RI.

Peters, C., editor (2006). *Working notes for the CLEF 2006 Workshop*, Alicante, Spain.

Pollard, C. and Sag, I. A. (1994). *Head-Driven Phrase Structure Grammar*. University of Chicago Press and CSLI Publications, Chicago, Illinois.

Purandare, A. and Pedersen, T. (2004). Word sense discrimination by clustering contexts in vector and similarity spaces. In *Proceedings of CoNLL-2004*, pages 41–48. Boston, MA, USA.

Quasthoff, U. and Biemann, C. (2006). Measuring monolinguality. In *Proceedings of the LREC-06 workshop on Quality Assurance and Quality Measurement for Language and Speech Resources*, Genova, Italy.

Quasthoff, U., Richter, M., and Biemann, C. (2006). Corpus portal for search in monolingual corpora. In *Proceedings of the fifth international conference on Language Resources and Evaluation (LREC-06)*, pages 1799–1802.

Ramakrishnan, G., Chakrabarti, S., Paranjpe, D., and Bhattacharya, P. (2004). Is question answering an acquired skill? In *Proceedings of the 13th international conference on World Wide Web (WWW-04)*, pages 111–120, New York, NY, USA. ACM Press.

Rapp, R. (1996). *Die Berechnung von Assoziationen: ein korpuslinguistischer Ansatz*. Olms.

Rapp, R. (2004). A practical solution to the problem of automatic word sense induction. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, pages 26–29, Morristown, NJ, USA. Association for Computational Linguistics.

Rapp, R. (2005). A practical solution to the problem of automatic part-of-speech induction from text. In *Conference Companion Volume of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL-05)*, Ann Arbor, Michigan, USA.

Roget, P. M. (1852). *Roget's Thesaurus of English Words and Phrases*. Penguin Books.

Roth, D. and van den Bosch, A., editors (2002). *Proceedings of the Sixth Workshop on Computational Language Learning (CoNLL-02)*, Taipei, Taiwan.

Sahlgren, M. (2006). *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Ph.D. thesis, Stockholm University.

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

Sapir, E. (1921). *Language: An introduction to the study of speech*. Harcourt, Brace and company.

Sarkar, A. and Haffari, G. (2006). Inductive semi-supervised learning methods for natural language processing. Tutorial at HLT-NAACL-06, New York, USA.

Schaeffer, S. E. (2005). Stochastic local clustering for massive graphs. In T. B. Ho, D. Cheung, and H. Liu, editors, *Proceedings of the Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-05)*, volume 3518 of *Lecture Notes in Computer Science*, pages 354–360, Berlin/Heidelberg, Germany. Springer-Verlag GmbH.

Schank, T. and Wagner, D. (2004). Approximating clustering-coefficient and transitivity. Technical Report 2004-9, Universität Karlsruhe, Fakultät für Informatik.

Schank, T. and Wagner, D. (2005). Finding, counting and listing all triangles in large graphs, an experimental study. In *Proceedings on the 4th International Workshop on Experimental and Efficient Algorithms (WEA-05)*, volume 3503 of *Lecture Notes in Computer Science*. Springer-Verlag.

Schmid, H. (1994). Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, Manchester, UK.

Schütze, H. (1993). Part-of-speech induction from scratch. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics (ACL-93)*, pages 251–258, Morristown, NJ, USA. Association for Computational Linguistics.

Schütze, H. (1995). Distributional part-of-speech tagging. In *Proceedings of the 7th Conference on European chapter of the Association for Computational Linguistics (EACL-95)*, pages 141–148, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Schütze, H. (1998). Automatic word sense discrimination. *Computational Linguistics*, **24**(1), 97–123.

Schütze, H. and Pedersen, J. O. (1995). Information retrieval based on word senses. In *Fourth Annual Symposium on Document Analysis and Information Retrieval*.

Schulze, B. M. (2000). Automatic language identification using both n-gram and word information. US Patent No. 6,167,369.

Schütze, H. (1993). Word space. In S. Hanson, J. Cowan, and C. Giles, editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann Publishers.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(8), 888–905.

Sigurd, B., Eeg-Olofsson, M., and van de Weijer, J. (2004). Word length, sentence length and frequency – Zipf revisited. *Studia Linguistica*, **58**(1), 37–52.

Šíma, J. and Schaeffer, S. E. (2005). On the NP-completeness of some graph cluster measures. Technical Report cs.CC/0506100, arXiv.org e-Print archive.

Simon, H. A. (1955). On a class of skew distribution functions. *Biometrika*, **42**(3/4), 425–440.

Smith, F. J. and Devine, K. (1985). Storing and retrieving word phrases. *Inf. Process. Manage.*, **21**(3), 215–224.

Steels, L. (2003). Evolving grounded communication for robots. *Trends in Cognitive Sciences*, **7**(7), 308–312.

Stein, B. and Niggemann, O. (1999). On the nature of structure and its identification. In *WG '99: Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 122–134, London, UK. Springer-Verlag.

Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, Boston, MA, USA.

Steyvers, M. and Tenenbaum, J. B. (2005). The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, **29**(1), 41–78.

Strogatz, S. (2001). Exploring complex networks. *Nature*, **410**(6825), 268–276.

Teresniak, S. (2005). Statistikbasierte Sprachenidentifikation auf Satzbasis. Bachelor's thesis, University of Leipzig.

Thelen, M. and Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, pages 214–221, Morristown, NJ, USA. Association for Computational Linguistics.

Tjong Kim Sang, E. and Buchholz, S. (2000). Introduction to the conll-2000 shared task: Chunking. In *Proceedings of CoNLL-2000*, Lisbon, Portugal.

Turney, P. D. (2006). Similarity of semantic relations. *Computational Linguistics*, **32**(3), 379–416.

Ushioda, A. (1996). Hierarchical clustering of words and applications to nlp tasks. In *Proceedings of the Fourth Workshop on Very Large Corpora*, pages 28–41, Somerset, NJ, USA.

van Dongen, S. M. (2000). *Graph Clustering by Flow Simulation*. Ph.D. thesis, University of Utrecht, Netherlands.

Van Rijsbergen, C. J. (1979). *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow.

Veronis, J. (2004). Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language*, **18**(3), 223–252.

Voorhees, E. M. (1993). Using wordnet to disambiguate word senses for text retrieval. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 171–180, New York, NY, USA. ACM Press.

Voss, J. (2005). Measuring Wikipedia. In P. Ingwersen and B. Larsen, editors, *ISSI2005*, volume 1, pages 221–231, Stockholm. International Society for Scientometrics and Informetrics.

Watts, D. J. and Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, **393**(6684), 440–442.

Wei, Y.-C. and Cheng, C.-K. (1991). Ratio cut partitioning for hierarchical designs. *IEEE Trans. on CAD of Integrated Circuits and Systems*, **10**(7), 911–921.

Widdows, D. (2004). *Geometry and Meaning*. Center for the Study of Language and Information/SRI.

Widdows, D. and Dorow, B. (2002). A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on Computational Linguistics (COLING-02)*, pages 1–7, Morristown, NJ, USA. Association for Computational Linguistics.

Witschel, H. and Biemann, C. (2005). Rigorous dimensionality reduction through linguistically motivated feature selection for text categorization. In *Proceedings of NODALIDA'05*, Joensuu, Finland.

Wu, Z. and Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, **15**(11), 1101–1113.

Yarowsky, D. (1994). Decision lists for lexical ambiguity resolution: application to accent restoration in spanish and french. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics (ACL-94)*, pages 88–95, Las Cruces, New Mexico.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pages 189–196, Cambridge, MA.

Zanette, D. H. and Montemurro, M. A. (2005). Dynamics of Text Generation with Realistic Zipf's Distribution. *Journal of Quantitative Linguistics*, **12**(1), 29–40.

Zha, H., He, X., Ding, C., Simon, H., and Gu, M. (2001). Bipartite graph partitioning and data clustering. In *CIKM '01: Proceedings of the tenth International Conference on Information and Knowledge Management*, pages 25–32, New York, NY, USA. ACM Press.

Zhu, X. (2005). Semi-supervised learning literature survey. Technical report, Computer Sciences, University of Wisconsin-Madison.

Zipf, G. K. (1935). *The Psycho-Biology of Language*. Houghton Mifflin, Boston.

Zipf, G. K. (1949). *Human Behavior and the Principle of Least-Effort*. Addison-Wesley, Cambridge, MA.

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, den 7. Juli 2007

Christian Biemann