Word Sense Induction Using Distributional Semantics

Induzieren von Wortbedeutungen mittels Distributioneller Semantik
Master-Thesis von Johannes Simon
Tag der Einreichung:

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Martin Riedl, MA

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Language Technology Group

Word Sense Induction Using Distributional Semantics
Induzieren von Wortbedeutungen mittels Distributioneller Semantik

Vorgelegte Master-Thesis von Johannes Simon

1. Gutachten: Prof. Dr. Chris Biemann
2. Gutachten: Martin Riedl, MA

Tag der Einreichung:

Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 5. Mai 2015

(Johannes Simon)

Contents

Abstract

Word Sense Disambiguation (WSD) has been shown to improve performance in Information Retrieval and Machine Translation systems, among others. While there are many WSD systems that perform well, most of them depend on either pre-defined sense inventories or on hand-labeled training data. Since such data is often only available for few languages such as English, Spanish or German, it is impossible to apply these systems other languages without significant manual efforts in creating the required data. Also, this data often lacks domain-specific word senses.

In this thesis, we investigate possibilities to apply Distributional Semantics to perform WSD in an unsupervised, knowledge-free fashion, also known as Word Sense Induction (WSI). In contrast to most other WSI systems, we put emphasis on creating a fixed, re-usable word-sense inventory that allows for sense labeling of previously unseen instances. To tune parameters involved in the process, we elaborate on a method to automatically assess the quality of induced senses by extracting a sense-labeled, large-scale evaluation dataset from Wikipedia, utilizing its link structure. We compare our results to other state-of-the-art WSI systems and show that our system performs competitively. Additionally, an outlook is given of a possible alignment of our induced sense inventory to other lexical resources to foster re-use of WSI systems.

Zusammenfassung

Word Sense Disambiguation (WSD), d.h. die Disambiguierung von Wortbedeutungen, kann maßgeblich dazu beitragen, bestehende Information-Retrieval-Systeme sowie Systeme für Machine Translation zu verbessern. Die meisten solcher WSD-Systeme benötigen allerdings entweder eine vordefinierte Liste an Wortbedeutungen oder von Hand annotierte Trainingsdaten, aus denen diese gelernt werden können. Da solche Ressourcen und Daten oft nur für wenige, weit-verbreitete Sprachen verfügbar sind, ist es unmöglich, solche Systeme auf andere Sprachen anzuwenden. Zudem fehlen vielen manuell erstellten Ressourcen bzw. Trainingsdaten domänenspezifische Wortbedeutungen.

In dieser Arbeit stellen wir einen Ansatz vor, der mittels Distributioneller Semantik Wortbedeutungen automatisch induziert, basierend auf unstrukturierten Textkorpora. Das Induzieren von Wortbedeutungen, insbesondere ohne Trainingsdaten, ist auch bekannt als Word Sense Induction (WSI). Im Gegensatz zu den meisten anderen WSI-Systemen legen wir unseren Fokus auf das Induzieren eines festen Word Sense Inventories, mit dem auch neue (”unseen”) Instanzen eines Wortes disambiguiert werden können. Um die involvierten Parameter unseres Systems zu optimieren, stellen wir einen Ansatz vor, um automatisch große Datensätze für die Evaluation von WSD-Systemen aus Wikipedia zu extrahieren. Zusätzlich vergleichen wir unser System mit anderen State-of-the-Art-WSI-Systemen im Rahmen einer weiteren Evaluation, und zeigen, dass unser System hier konkurrenzfähig ist. Abschließend geben wir einen Ausblick auf weitere Anwendungsmöglichkeiten unseres Systems, bei denen das induzierte Sense Inventory zu bestehenden Ontologien verlinkt wird.

Acknowledgement

I want to sincerely thank Chris for being a great supervisor, for his many inspirations, and for the extensive feedback I received from him whenever I asked. I really appreciated the freedom he gave me during my thesis, yet reminding me to stay focused on the important things.

Also I want to thank Martin for answering all my many questions, for reviewing my work, and for providing me with many useful practical tips.

My thanks also go to Eugen, for being a great cluster administrator and for fixing it whenever something got broke.

Auch meinen Eltern möchte ich danken, für die viele Unterstützung während meines ganzen Studiums, und dafür, dass sie immer da sind, wenn ich sie brauche.

Last but not least, I want to express my deepest gratitude and love to Verena for her endless support in all situations. She kept me motivated during my entire work.

1 Introduction

A central challenge in Artificial Intelligence (AI) is the ambiguity of human language. The task of identifying the appropriate sense of ambiguous word mentions, called Word Sense Disambiguation (WSD), is non-trivial for computer programs. Some go as far as saying that WSD is AI-complete, meaning that solving it is at least as hard as solving the most difficult problems in AI. [Mallery, 1988]

On the first glance this might come as a surprise: Internet search engines like Google can deal quite well with different meanings of entered key words. If you enter the search query "where to buy a jaguar", it correctly presents results of car sellers, not of nearby pet shops. Here individual key words disambiguate each other, as there are many more webpages about buying sports cars than about buying exotic animals.

The complexity of WSD becomes more apparent when we look at other AI tasks like Machine Translation (MT): Consider the following sentence (in the following called a disambiguation *instance*):

> *He decided to have grilled **bass** for dinner.*

Here Google's translation service[1] will incorrectly produce the German translation

> *Er entschloss sich, **Bass** zum Abendessen vom Grill haben.*

Here *bass* is mistakenly identified as referring to sound or music, not to a fish species. This is either because Google's database does not link *bass* to *Barsch* or because it does not include necessary world knowledge. This would include the fact that *Barsch* is the German translation of the sense referring to fish, and that *grilling* usually refers to this sense and not to others.

But also Information Retrieval (IR) may benefit from the use of WSD: While most disambiguation is already performed by ranking of results, an accurate WSD step may improve performance further. For example, the query "Bank of America" may return documents referring to river banks, but usually below documents referring to financial institutions. However, a WSD component that accurately identifies the sense of *bank* in the query and in all documents would allow to rule out other senses and reduce presented results, and therefore increase precision while pertaining the same recall. That this improvement is difficult to achieve in practice was shown by Sanderson [Sanderson, 1994], who concluded that, for his experiments, WSD does not improve IR performance. However, Schütze and Pedersen later showed that a WSD component with 90% accuracy results in an improvement of IR accuracy from 29.9% to 34.2% in a standard IR test collection [Schütze and Pedersen, 1995].

[1] https://translate.google.com

1.1 Motivation

It is possible to manually build high-quality lexical resources that model this world knowledge. A prominent example is WordNet [Fellbaum, 1998], or its inter-lingual counterpart BabelNet [Navigli and Ponzetto, 2012], which builds upon WordNet, among others. However human-built lexica like these have several drawbacks:

- Hiring linguists to do manual annotations or lexicon-writing is expensive.
- Language changes rapidly, new words are born and other words become outdated: for example, the word *tablet* is nowadays also used to refer to hand-held computers. Similarly, *dialog* nowadays specifically refers to dialogs in a graphical user interface (GUI) when referred to in computer-science domains. This meaning is, however, relatively new and e.g. not listed in WordNet, which was written before this use became popular.
- The number of word senses is often inconsistent and not "reproducible" for other linguists or computers.
- Contextual information for word senses, e.g. example sentences, is sparse. Without *clues* like these it is hard for WSD systems to choose an appropriate sense in context.
- They are domain-dependent and hardly adaptable to other domains, reversely they do not contain certain domain-specific senses.
- They contain rare senses that are irrelevant for most uses and therefore rather confuse WSD systems (such as the word *computer* in WordNet: contains a sense referring to a person performing computations).

As a consequence of the high cost of manually building such resources, there exist only a handful of them for English and other popular languages like Spanish or German. Therefore, languages of minorities ("under-resourced languages") often completely lack these resources. Moreover, due to the hand-written nature of these resources and the complexity of language, they often lack important contextual information.

Another way to perform WSD, without the use of pre-determined lexical resources, is the use of Word Sense Induction (WSI). Here, information necessary for distinction of word senses are automatically *induced*, usually from raw text corpora. This is specifically done without human intervention. Instead, WSI systems extract statistical information about word contexts from such corpora, and based on this information either project words or word instances into a high-dimensional vector space or build a graph in which related words are connected. *Clustering* methods are then used to group related words (or instances of these) into word senses. Obviously, this has the advantage that sense inventories can be computed in a relatively short period of time, at relatively little cost. Also, results are sensitive to domain changes, i.e. using a text corpus from a different domain will be reflected in the induced word senses. This is especially useful for narrow domains, e.g. the medical domain, that make use of a well-defined language. The goal of this thesis is to develop an unsupervised, integrated WSI and WSD system that can automatically construct such word-sense inventories from large amounts of text and use them for disambiguation in context. The challenge in this task is to produce sense inventories that do not only provide a raw sense list, but also *clues* for disambiguation (which we call *context clues*) with both broad coverage, as well as high accuracy.

To foster re-use of the developed methods, we also provide a short outlook on possible integration with existing lexical and ontological resources, such as WordNet, OpenCyc [Lenat, 1995; Matuszek et al., 2006] or FreeBase [Bollacker et al., 2008].

1.2 Hypothesis

The goal of this thesis is to develop an unsupervised, integrated WSI and WSD system (in the following referred to as *WSID* system) that can automatically construct word-sense inventories from large amounts of text and use them for disambiguation in context. Most top-performing WSI systems today notably differ from this approach: they induce senses for a specific, relatively low number of word instances (as opposed to words), by clustering these instances into groups. This is often called sense *discrimination*, as these systems do not identify specific senses from a given or induced sense inventory in context, but only group word instances into clusters with similar senses. Therefore, these systems are incapable of sense-labeling isolated instances. Also, as they do not draw senses from a fixed sense inventory, this makes it hard to link induced senses to existing sense inventories. This, however, could foster re-use of WSI systems and build a bridge between supervised and unsupervised disambiguation methods.

A crucial difference of our goal to many previous systems is hence the use of a fixed, pre-computed word sense inventory. However, when such an inventory is computed beforehand, this requires relevant ambiguous words to be known upfront. Since this is often not the case, we target to support an all-words setting, which comes with the inherent need for scalable, efficient WSI system. We formulate the following question that will guide us through the work of this thesis:

How can we construct a WSID system, which disambiguates unseen instances based on an induced sense inventory with high reliability and coverage?

This question can be further split into two subordinate questions:

1. How can we reliably and effectively induce a sense inventory for a large part of the vocabulary from large corpora?

2. How can we reliably disambiguate word instances in context, using either an induced or a predefined sense inventory?

1.3 Outlook

The structure of this thesis is briefly outlined in the following. In Chapter 2, we summarize current possibilities to approach the general problem of Word Sense Disambiguation. Namely, we provide an overview over several supervised and knowledge-based approaches, as well as their performance in established WSD evaluations.

Before elaborating on unsupervised WSD methods, we in Chapter 3 introduce the concept of *Distributional Similarity*, a way to model and to compute semantic similarity of language elements such as words or multi-word expressions. This concept will later be used as the foundation for inducing word senses.

Chapter 4 discusses current approaches to unsupervised, knowledge-free WSD, which, opposed to supervised and knowledge-based WSD methodologies, do not directly depend on any human-crafted resources or training data. We again briefly summarize the performance, strengths and weaknesses of existing approaches. Following this discussion, we introduce a new method of WSI that utilizes Distributional Semantics. We in detail show how this method produces sense clusters by building a word-similarity graph and clustering this graph using the Markov Chain Clustering (MCL) algorithm. To provide means for disambiguation of the induced senses in context, a probabilistically motivated approach is discussed in detail that allows for extraction of contextualization clues in a fully unsupervised manner.

To support our selection process of an optimal distributional thesaurus setting, the foundation of our sense induction algorithm, Chapter 6 presents the results of two different evaluation scenarios: First,

the impact of choosing different frequency thresholds to cut off features is analyzed, in order to reduce noise. Additionally, to avoid cutting off or penalizing less frequent word senses in the induction process, we analyze the effect of the number of features used to compute word similarities on the resulting similarities between rare and frequent words.

To verify the hypothesis of this work, Chapter 7 contains two evaluations of the performance of the implemented WSI system. First, we introduce a large-scale WSI evaluation dataset, extracted from Wikipedia in a fully unsupervised manner, utilizing its link structure. This evaluation is then used to both assess the performance of our system, as well as to tune parameters without relying on manually sense-labeled data. Secondly, we tested settings of our system that were optimized with regard to the first, unsupervised evaluation, against a part of the SemEval-2013 WSI subtask dataset.

In Chapter 8, we briefly outline a novel approach that attempts to bridge the gap between unstructured (e.g. text corpora) and structured knowledge resources (e.g. complex ontologies) by adding a new, intermediate layer based on induced sense inventories. Specifically, we demonstrate a possible benefit of joining induced word-sense inventories and ontological resources at the example of Entity Linking (EL), a task highly intertwined with WSD.

Implementation details of our system are given in Chapter 9. Finally, Chapter 10 summarizes the results of this work and discusses the validity of the hypothesis and limitations of our method. Also, an outlook is given over possible future work that we believe can further improve the performance of the developed system.

2 Word Sense Disambiguation: A State of the Art

Both instances *He decided to have grilled **bass** for dinner*, and *He turned up the volume to feel the **bass** of his sound system* contain the same lexical entity *bass*. However, both mentions of this term represent two distinct *semantic entities*: a sense meaning a fish and a sense meaning a low frequency sound, resp. A word-sense inventory is a lexical resource that contains information about word-sense distinctions. It additionally may have contextual information that allows the disambiguation of word senses in context.

According to a survey of Navigli, WSD is the *"ability to identify the meaning of words in context in a computational manner"* [Navigli, 2009][1]. While the goal of all such methods is the same, there are substantial differences in their implementation. A lot of the systems that exist today use knowledge-based approaches that work on handcrafted sense inventories, like the aforementioned WordNet, while others use supervised approaches that learn from hand-labeled training data.

However, hand-crafted lexical resources and hand-labeled training data are expensive to create, often inconsistent and domain-dependent. The alternative to these are unsupervised[2], knowledge-free approaches, which are referred to as WSI techniques.

Generally speaking, there exist two scopes for WSD: Some tasks require only disambiguation of one or few words in every piece of text (called *lexical-sample WSD*). Other tasks require the WSD algorithms to disambiguate all word mentions in the specified piece of text, which is called *all-words WSD*. In this thesis we concentrate on the former, specifically on WSD for nouns. The reasons for this are twofold: First, most research on supervised methods has been concentrated on this field. In order to show that unsupervised approaches are competitive in performance, while bringing many advantages, we decided to reduce our efforts on lexical entities that were studied most, i.e. nouns. The other reason is that the focus on nouns opens up the possibility to use Wikipedia as a rich, free resource for evaluation by utilizing its link structure, as we will show in Section 7.3.

The oldest approach to WSD used existing, human-readable dictionaries: In the 80s, Lesk used the *Oxford Advanced Learner's Dictionary of Current English* to look for overlaps between the definition of a target word (i.e. the word to be disambiguated) and definitions of nearby words [Lesk, 1986] in order to find the most appropriate word sense in a computational manner. Since then, many more advanced techniques have been presented, most of them outperforming the original *Lesk algorithm* by far [Navigli, 2009]. While some of them use dictionaries specifically designed for computational use, such as WordNet, others have concentrated more on the sole use of Machine Learning (ML) by training classifiers on hand-annotated samples. This chapter provides a brief overview over current state-of-the-art techniques, how they perform, and in which cases one may be preferred over the other.

2.1 Knowledge-Based WSD

Any WSD method that uses pre-defined dictionaries, lexical resources or semantic ontologies can considered to be *knowledge-based*. Most of the resources that exist today are of *shallow* nature, which means

[1] For another survey on the topic of WSD, see [Agirre and Edmonds, 2007].

[2] Note that historically speaking, *unsupervised* may also refer to knowledge-based approaches that required no specific training. WSI techniques, in contrast, are knowledge-free.

that they mostly use or provide only lexical information on the word senses. For example, the fact that a *bank* may be related to *finance*, that it is a noun and that the financial sense often appears in the form of *Bank of ...* is shallow knowledge: it does not encode complex world knowledge such as that *banks have customers that have accounts from which they can withdraw money*, etc. This, on the other hand, is usually called *deep* knowledge. The border between shallow and deep knowledge is not clear-cut: For example, WordNet provides the information that *bank* is a type of *banking company*; however the information that it is a company is merely lexically encoded in the name of this entity.

Noun

- S: (n) **bass** (the lowest part of the musical range)
- S: (n) **bass**, bass part (the lowest part in polyphonic music)
- S: (n) **bass**, basso (an adult male singer with the lowest voice)
- S: (n) sea bass, **bass** (the lean flesh of a saltwater fish of the family Serranidae)
- S: (n) freshwater bass, **bass** (any of various North American freshwater fish with lean flesh (especially of the genus Micropterus))
- S: (n) **bass**, bass voice, basso (the lowest adult male singing voice)
- S: (n) **bass** (the member with the lowest range of a family of musical instruments)
- S: (n) **bass** (nontechnical name for any of numerous edible marine and freshwater spiny-finned fishes)

Adjective

- S: (adj) **bass**, deep (having or denoting a low vocal or instrumental range) *"a deep voice"; "a bass voice is lower than a baritone voice"; "a bass clarinet"*

Figure 2.1: Entry for word *bass* in WordNet 3.1.

Simplified Lesk Algorithm

The Simplified Lesk (SL) Algorithm [Kilgarriff et al., 2000] is one of the simplest disambiguation algorithms based on word lexica. Due to the requirement of a lexicon, it is a *knowledge-based WSD* algorithm. Its idea is the following: Given an ordinary, human-readable dictionary, and a word in context to be disambiguated, the Simplified Lesk algorithm takes the sense that, given its gloss (description) and example sentences, has the highest overlap with the context. It is therefore a *similarity-based* method. See Figure 2.1 for an example from WordNet for the word *bass*. See Algorithm 1 for an exact definition.

Vasilescu et al. performed an evaluation of Lesk-based algorithms (including SL) on the SENSEVAL2 task [Vasilescu et al., 2004]. Here they tried different settings: They used all context provided for disambiguation or only context in a window of e.g. 2 or 8 words left or right of the word. While their baseline, which always assigned a word's most frequent sense (MSF), scored a precision[3] of 57.99% and recall[4] of 57.62%, the best-scoring window was ± 2 words with a precision of 58.18% and recall of 57.66%. They also attempted using sense probabilities computed from a the WordNet-sense-tagged SEMCOR corpus, based on a Naive Bayes classifier using the word's context, but this approach did not bring any improvement.

[3] Precision = correctly assigned senses over total assigned senses

[4] Recall = correctly assigned senses over total number of senses to be assigned (not all contexts can be assigned to a sense, e.g. due to missing gloss overlap or ties between two or more senses)

They also found that, unless using a Naive Bayes classifier, (a) increasing the context window size always decreases WSD performance and (b) filtering certain stop words from the context increases WSD performance. This makes only sense, as the appearance *or absence* of a stop word like *the*, *to* or *from* usually provides little evidence for preferring one sense over another (b). Also, this means that words further away from the word to be disambiguated tend to provide less useful clues about the appropriate sense (a).

Algorithm 1: Simplified Lesk algorithm. OVERLAP computes the overlap between two sets of words, excluding stop words.

```
   input  : word, sentence
   output: best-sense
 1 best-sense ← most frequent sense for word;
 2 max-overlap ← 0;
 3 context ← sentence as bag of words (BOW);
 4 for each sense in senses of word do
 5 |   signature ← set of words in the gloss and examples of sense;
 6 |   overlap ← OVERLAP(signature, context);
 7 |   if overlap > max-overlap then
 8 |   |   max-overlap ← overlap;
 9 |   |   best-sense ← sense;
10 |   end
11 end
```

Other similarity-based approaches maximize the similarity among all senses to be chosen in a piece of text, instead of maximizing similarity between individual senses and sense definitions. More precisely, they minimize the total *distance* of the shortest paths connecting all individual word senses. Take for instance the sentence *He withdrew money from his bank account*. The correct sense of *bank*, referring to a financial institution, will have a shorter distance to *money*, *withdraw* and *account* than the sense referring to a park bank. Rada et. al [Rada et al., 1989] performed WSD based on hypernymy relations within WordNet.

2.2 Supervised Approaches

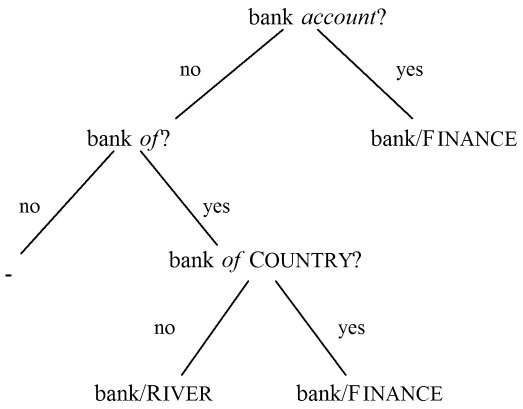

Figure 2.2: Decision tree for disambiguation of word *bank*, with the two possible senses *bank/RIVER* and *bank/FINANCE*

Decision trees are one of the simplest methods for object classification. Each edge connecting two nodes represents a decision, leaf nodes represent classes (i.e. the final “decisions”). In the context of WSD, each non-leaf (internal) node is usually a context feature, and outgoing edges from these nodes are their values. This may be just "yes" or "no", as illustrated in Figure 2.2. The actual observed features in context then determine the path to be taken. Decision trees have several advantages: often, they are relatively compact in size, and due to their logarithmic time complexity much faster than other WSD algorithms. Also, they are human-readable and intuitive to understand. However, other supervised WSD algorithms consistently outperform decision trees [Navigli, 2009, p. 17]. Experiments by Wee et al. [Wee, 2010] show that decision trees using bag-of-word features are unable to outperform a simple Most frequent sense (MFS) baseline. The experiments they conducted used the SensEval-2, SensEval-3 and SensEval-2007 tasks, their baseline picked the most frequent WordNet sense.

Neural networks, when applied to WSD, attempt to model the associative nature of human language: A *bank* may be associated with *money*, *rivers* or with *blood* (as in *blood bank*). A possible neural network can directly leverage this observation for the use of disambiguation, and link each of these (but not exclusively these) words, or other features (such as POS tags), to a corresponding sense or to other associated words. Relevant words or features that are useful for disambiguation are represented as nodes in the neural network, called *neurons*. Every neuron has a number of associated other neurons. For disambiguation, when a certain word or feature is observed in a context, this neuron is activated. Other neurons (of which their corresponding word or feature is not observed) are activated once a certain threshold of associated neurons is activated. Activation therefore spreads through the neural network until it reaches a stable level. The word sense with the highest activation level then indicates the most “fitting” sense according to this model. While this model is intuitive to understand, the training process to optimize parameters such as connection weights and activation thresholds is complex. Several experiments have been conducted on neural network performance for WSD, e.g. Leacock et al. 1993 [Leacock et al., 1993], Towell and Voorhees [Towell and Voorhees, 1998], Mooney [Mooney, 1996]. However, there are few to no recent studies taking current state-of-the-art insights about WSD into account. Therefore, we at this point make no assumptions about the performance of neural networks compared to other more recently studies approaches.

Support Vector Machines (SVMs) provide another supervised approach to object classification. SVMs generally decide on a “yes-or-no” basis, i.e. they can only be trained on a certain class: Given a feature vector, a SVM will determine whether this vector fits to a certain class or not, and how confident it is about this statement. This confidence is not to be confused with a conditional probability, but can indeed be used as an indicator of the same. Internally, a SVM projects the feature vector into an appropriate multi-dimensional space, in which a pre-determined hyperplane indicates whether the SVM’s class is a fit for the feature vector (the projected data point is below the hyperplane) or not (the data point is above the hyperplane). A confidence can be determined by taking the distance of the projected feature vector to the hyperplane into account. SVMs have shown to be the most successful mechanism for supervised WSD to date [Lee and Ng, 2002].

A probabilistic classifier that is relatively easy to use and implement, but also fast in processing speed is the *Naive Bayes classifier*. The “naive” in its name refers to its assumption that all features used for classification must be statistically independent. Using this assumption, the classifier approximates a conditional probability $P(s_k|C)$ for a sense s_k and a context feature vector C by simple multiplication of the individual conditional probabilities $P(c_i|s_k)$, together with multiplication of further factors. Since all these probabilities can be computed by mere counting of sense-feature frequencies, a Naive Bayes classifier does not require iterative training. While Naive Bayes classifiers are outperformed by more powerful methods such as SVMs and Maximum-entropy (MaxEnt) classifiers, they generally compare well despite their independence assumption for feature vectors [Mooney, 1996; Ng, 1997b; Leacock et al., 1998; Pedersen, 2007; Bruce and Wiebe, 1999].

Another classifier that can be trained on word-feature frequencies is the MaxEnt classifier. In contrast to Naive Bayes classifiers, it does not require individual features do be statistically independent. Its basic idea is that of maximum entropy: As long as we have no information regarding a specific probability distribution, assume all class probabilities are distributed evenly. For example, if we know that for a certain feature vector, sense s_2 appears 40% of the time, and sense s_5 appears 10% of the time (and there are 5 senses in total), then assume that the remaining 3 senses (s_1, s_3 and s_4) appear $(100-40-10)/3 = 50/3 \approx 16.6\%$ of the time each. Information encoded in the model is therefore minimized and its entropy is maximized. In other words, among the models that fit the training data, a MaxEnt classifier chooses the model with the least assumptions. Intuitively, this avoids overfitting of models to the training data, leading to better performance on unseen (i.e. held-out) data. That MaxEnt models indeed perform competitively when applied to WSD was shown by Tratz et al. [Tratz et al., 2007]. His WSD system using a MaxEnt model received the highest F-score for the fine-grained all-words English subtask of the Senseval 3 challenge.

Other supervised classifiers used for WSD include instance-based learning methods, e.g. k-Nearest Neighbor (kNN) classifiers. These methods compute the similarity of new instances to other, previously classified instances, and choose the class of the instance(s) with the highest similarity. They have been found to be among the highest-performing methods in WSD [Ng, 1997a; Daelemans et al., 1998]. Furthermore, ensemble methods [Klein et al., 2002; Florian et al., 2002] intend to combine the strengths of multiple different classifiers. Experiments have shown that the performance of such a compositional system, applied to WSD, can surpass the performance of each of the individual classifiers [Florian et al., 2002].

Lastly, *semi-supervised* methods employ unsupervised disambiguation techniques, but require a *seed* of sense-labeled training instances to start with, which is usually referred to as *bootstrapping*. For example, [Yarowsky, 1995] manually sense-tagged a small number of seed instances. They then applied a disambiguation technique based on two properties of natural language to sense-tag the remainder of the corpus: *one sense per collocation* [Yarowsky, 1993] and *one sense per discourse* [Gale et al., 1992]. They tested their system on 10 nouns with two coarse-grained senses each and reported a high accuracy of 96%. Notably, their system surpassed the performance of two previous supervised and unsupervised systems by [Schütze, 1992] when applied to the same subset of 4 nouns (*tank, space, motion, plant*)

2.3 Annotated Corpora for Training and Evaluation

Supervised approaches highly depend on training data in the form of hand-annotated disambiguation instances. Due to the need for comparability of evaluation results, as well as high cost of producing such data, there today exist a handful of resources that are repeatedly re-used.

One of the oldest, largest, and most widely used [Navigli, 2009] collection of training data is the Semantic Concordance (SemCor) corpus[5] [Miller et al., 1993]. It contains 352 English text documents, with in total around 234,000 sense annotations. MultiSemCor [Pianta et al., 2002], in contrast, contains both English as well as parallel Italian texts, each annotated with corresponding WordNet senses. Leacock et al. [Leacock et al., 1993] presented a corpus containing 4000 sense-tagged instances, each with a sense-annotated occurrences of one of the three words *line* (noun), *serve* (verb) and *hard* (adjective). Bruce and Wiebe [Bruce and Wiebe, 1994] manually annotated Longman Dictionary of Contemporary English (LDOCE) senses for 2369 occurrences of the word *interest* in their *interest corpus*. Ng and Lee [Ng and Lee, 1996] introduced a notably larger collection, the *DSO corpus* with a total of 192,800 sense-annotated occurrences of 191 words. Here, texts were taken from the *Wall Street Journal* as well as the *Brown* corpus [Kučera and Francis, 1967]. Also, Chklovski and Mihalcea [Chklovski and Mihalcea, 2002] presented a collaboratively created corpus, the *Open Mind Word Expert* data set, containing sentences

[5] `http://web.eecs.umich.edu/~mihalcea/downloads.html` (last accessed: 2015/04/30)

for 288 semantically annotated nouns. Biemann introduced *TWSI*, the Turk Bootstrap Word Sense Inventory [Biemann, 2012], which contains lexical substitutions for 1012 highly frequent English nouns in more than 118,000 contexts. Here, lexical substitutions of these words were collected for each of these contexts using a crowd-sourcing platform and word occurrences were clustered into senses by using their lexical substitution overlap, with an additional manual cluster verification step.

2.4 Unsupervised Approaches

Unsupervised approaches generally neither make use of handcrafted lexical resources, nor of hand-annotated training data. Their challenge is therefore to *induce* such sense inventories automatically from raw, unstructured text corpora, for which reason they are called Word Sense Induction (WSI) methods. Conceptually, two types of induction methods can be distinguished: methods using *vector space models* and methods using *graph clustering* techniques. Chapter 4 will introduce both approaches generally, and present popular implementations. However, before we turn to induction of word senses, we will introduce the concept of *Distributional Similarity*, a method for the computation of word similarities, which we need for WSI.

3 Distributional Similarity

You shall know a word by the company it keeps (J.R. Firth, 1957)

Word Sense Induction, the main topic of this thesis, is by definition the computation of different senses of words from unstructured text (i.e. a reference *corpus*), mostly without prior knowledge about the language used, and equally importantly, without prior knowledge about the target domain. Before we can induce different senses of a word in a computational manner, we therefore need a notion of inducing *structure* in the reference corpus, which we can in turn use to infer structure in the use of words, i.e. the word senses themselves. For this, we will in this chapter introduce a way to compute *similarities* among words. Based on these similarities, we will in Chapter 4 introduce a method to compute *word clusters*, which are ultimately nothing but the word senses were are looking for.

The foundation for computing word similarities is an implication of the *Distributional Hypothesis* [Harris, 1954], which can be summarized by the popular phrase of J. R. Firth *"You shall know a word by the company it keeps"* [Firth, 1957]. The *Strong Contextual Hypothesis* [Miller and Charles, 1991] that *"two words are semantically similar to the extent that their contextual representations are similar"* is a direct implication of this hypothesis. We can directly use this observation to compute word similarities: The more often two words appear in the same contexts, the more they are similar. A *context* can generally be anything from an entire surrounding paragraph to only a word's right neighbor. However, when referred to in this work, is is rather a small part of the surrounding sentence that, preferably, is in direct relation to the word in question. Consider the following sentence from the introduction:

*He decided to have grilled **bass** for dinner.*

The idea of the distributional hypothesis is that, if we could observe other words in the same context, e.g. *He decided to have grilled **fish** for dinner*, we would know that these two words are semantically similar. However, observations of this kind are very unlikely (rare, at least) due to the heterogenous nature of human language. Therefore, we need some way of observing when two words appear in *partially* equal contexts.

Without *parsing* of the text, with the exception of tokenization, we can already say several things: *bass* appears in conjunction with another word as *grilled bass*. This is usually referred to as a *bigram* (more generally: n-gram), or as *collocation* if the n-gram appears more often than expected by chance. Also, *bass* appears in the same sentence as *dinner*. The same observation can be made for the remaining words. This is called *co-occurrence*. For a more in-depth introduction of the features involved, refer to Section 3.2.

Having these *context features*, computing distributional similarity between two words is straightforward. In principle, it is a matter of counting how many of these (partial) contexts they share. For an exact definition, refer to the following section.

3.1 Definition

For our work, we used the definition proposed by Biemann et al. [Biemann and Riedl, 2013]. Given two words (or terms) t_1 and t_2 and their respective set of context features $feature(t_1)$ and $feature(t_2)$, distributional similarity can be defined as

$$sim(t_1, t_2) = \left| features(t_1) \cap features(t_2) \right| \tag{3.1}$$

To speed up processing, and to reduce noise, the actual definition we use is the following:

$$sim(t_1, t_2) = \left| rankedfeatures(t_1, p) \cap rankedfeatures(t_2, p) \right| \tag{3.2}$$

where $rankedfeatures(t, p)$ returns only the p most significant features per term, based on a given significance measure such as LMI or LL (for an overview of significance measures, see Table 3.3). The simplicity of this definition might surprise, however this approach to computing distributional similarity is not only scalable, but also superior to other known approaches in many cases, which will be briefly discussed in Section 3.4.

3.2 Distributional Features

Since we want to use distributional similarities to induce and also disambiguate different senses of a word, the similar terms produced by our algorithm is essential to us. Depending on the context features used to compute similarities, the resulting similar words differ: for example, using direct neighbors of a word as features will produce mostly syntactic similarities, while using sentence co-occurrences as features produces mostly semantic similarities [Bordag, 2006], [Curran, 2003]. We use this insight especially for unsupervised learning of a disambiguation model, which builds upon grammatical compatibility of similar words. Section 5.1 discusses this in detail.

For comparison of context representations, clustering methods and similarity measures, see [Purandare and Pedersen, 2004].

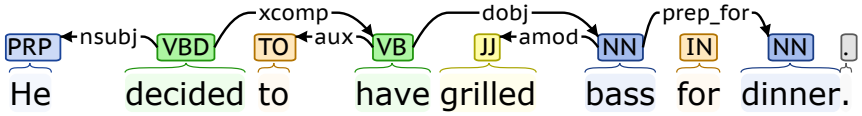

Figure 3.1: Dependency parse of example sentence from Stanford CoreNLP. Image taken from web visualizer at `http://nlp.stanford.edu:8080/corenlp/process` (last accessed: 2015/04/30).

Jo	Bim
bass	amod(•, grilled) prep_for(•, dinner) dobj(have, •)
He	nsubj(decided, •)
decided	nsubj(•, He) xcomp(•, have)

Table 3.1: Dependency context features for *bass*, *He* and *decided* from the parsed example sentence above. Note how an arbitrary character or symbol such as '•' acts as a placeholder for the head word (*jo*). This operation, called *holing operation* is the foundation for creating *distributional features*, i.e. features that can be shared by different words.

Co-occurrences are among the most studied features used for computing word similarities. They correspond to a simple bag-of-words (BOW) model of context representations: A word co-occurs with another

word if they appear anywhere within the same context. Due to the insensitivity to local structure around a target word, this distributional features yields word similarities that are rather topical: For example, *bass* might be similar to *fishing rod* or *singing*. These are not synonyms of any sort, but rather topically related words.

Stanford dependencies are a way to represent grammatical relations among words in a sentence. Consider the example sentence from Figure 3.1: Here, *He* is the subject (`nsubj`) of the first clause around the verb *decided*, while *bass* is the direct object (`dobj`) of the second clause around the verb *have*. An excerpt of distributional features that can be extracted from this example are listed in Table 3.1. Distributional similarities based on such dependencies are oriented more the local context: Using dependency features, *fishing rod* will likely receive a much lower similarity score relative to *bass* than e.g. *eel*, even though they are both highly related to *bass*. For example, a *fishing rod* may significantly often appear within the same context as the verb *to grill*, but it will never (or very rarely) have a direct dependency relation to this verb (in contrast to *bass*, which makes them less similar with respect to dependency features).

3.3 Distributional Similarity Algorithm

To compute distributional similarities, we used a pipeline as implemented in the *JoBimText* framework[1] [Biemann and Riedl, 2013]. Figure 3.2 illustrates this pipeline, which can be summarized as follows: First, distributional features are extracted on a per-word basis from a text corpus. In a second step, frequencies of words, features and word-feature co-occurrences are counted. Words, features and word-feature co-occurrences below a frequency of t_w, t_f and t_{wf}, resp., are discarded (for a listing of parameters, refer to Table 3.2). Based on these frequencies, significance scores for every word-feature co-occurrence are computed using the significance measure determined by sig. By ranking features using this significance score, only the p most significant features are chosen for every word. Using these features, word similarities are computed as described in Section 3.1. This is again followed by a pruning step in which only the l most similar terms are kept for every word. Finally, the thesaurus entry of every word is sorted to list most similar terms first. For details on the implementation, refer to Chapter 9.

MapReduce

To support massively parallel processing of this pipeline, JoBimText is implemented using MapReduce [Dean and Ghemawat, 2004]. The MapReduce programming model, in a nutshell, allows the specification of operations on splits of data that can be executed entirely independently from each other. Specifically for our implementation, this means that we, as a first step, created a data split for every sentence in the text corpus. This also means that context beyond the sentence boundary is not available for the induction or the disambiguation algorithm, however we assumed that this does not negatively impact performance. The first operation, performed on these sentence splits, yielded a new data split for every extracted context feature such as word dependencies in the key-value form of *(word, feature)*, which resembles the *map* step in the MapReduce paradigm. The MapReduce implementation then internally assigns a new cluster node to further process each key (here, the word) and sends all splits belonging to this key to the same node. This is usually referred to as *shuffling*. When splits belonging to the same key are received by the assigned cluster node, they can be processed in the complementary *reduce* step to, in this case, sum up the frequencies for each extracted feature for a specific word. For the remaining pipeline steps, we proceeded similarly, sticking to operations allowed in the MapReduce model.

[1] `http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/` (last accessed: 2015/04/30)

3.4 Differences to vector-space models

Traditionally, distributional similarity is computed using vector-space models (Schütze, 1993; Erk and Pado, 2008; Baroni and Zamparelli,2010) (for an introduction, see Section 4.1).[2] While vector-space models have the advantage of being widely known and well-researched, they lack the possibility of being scaled to very large datasets, due to their computational complexity [Gliozzo et al., 2013].

The major advantage of the JoBimText approach is that it has been found superior to others when applied to very large datasets in semantic relatedness evaluations [Biemann and Riedl, 2013, p. 78]. Specifically, Biemann et. al. compared the results of the JoBimText approach using LMI and LL scores with the results of [Lin, 1998] and [Curran, 2002], with the average WordNet Path Similarity as quality measure. For the 10 most frequent words in a corpus of 120M words, Lin's DT received a score of 0.279, Curran's DT a score of 0.254 and a DT produced by JoBimText a score of 0.283. JoBimText also produced results superior to the other systems for 10 infrequent nouns from the same corpus. This shows that, despite the simplicity of the JoBimText approach to computing distributional similarities, it compares very well to vector-spaced approaches, and even surpasses quality of vector-space-based DTs in many cases. Especially when applied to large corpora, this approach produces superior results, while scaling well due to its MapReduce-friendly computation of word similarities.

As potential reasons for the improvement of JoBimText over Lin's and Curran's system, Biemann et. al. discussed several possible reasons: Firstly, Lin's measure used for similarity computation likely puts too much emphasis on frequent relations. This tends to reduce noise, however may overly prefer frequent relations in larger corpora. Secondly, they used a different parser and other test words than Curran did, which could explain the comparably better performance at least to some degree. Lastly, they point out that Curran used a different evaluation method to test his system, which he likely also used for optimization of his method.

Parameter	Description
t_w	Minimum frequency of included words
t_f	Minimum frequency of included features
t_{wf}	Minimum word-feature frequency of included features
w	Maximum number of unique words a feature is allowed to co-occur with
p	Maximum number of features to use per word (remaining features are dropped on a per-word basis)
l	Number of similar words to compute for every word
sig	Significance measure used to score and rank features for every word. Can be either *LMI*, *LL* or *PMI*.

Table 3.2: Parameters involved in computing distributional similarities using either the JoBimText framework or our Spark implementation.

[2] For comparison of distributional similarity computations, see [Lin, 1998], [Lin and Dyer, 2010]

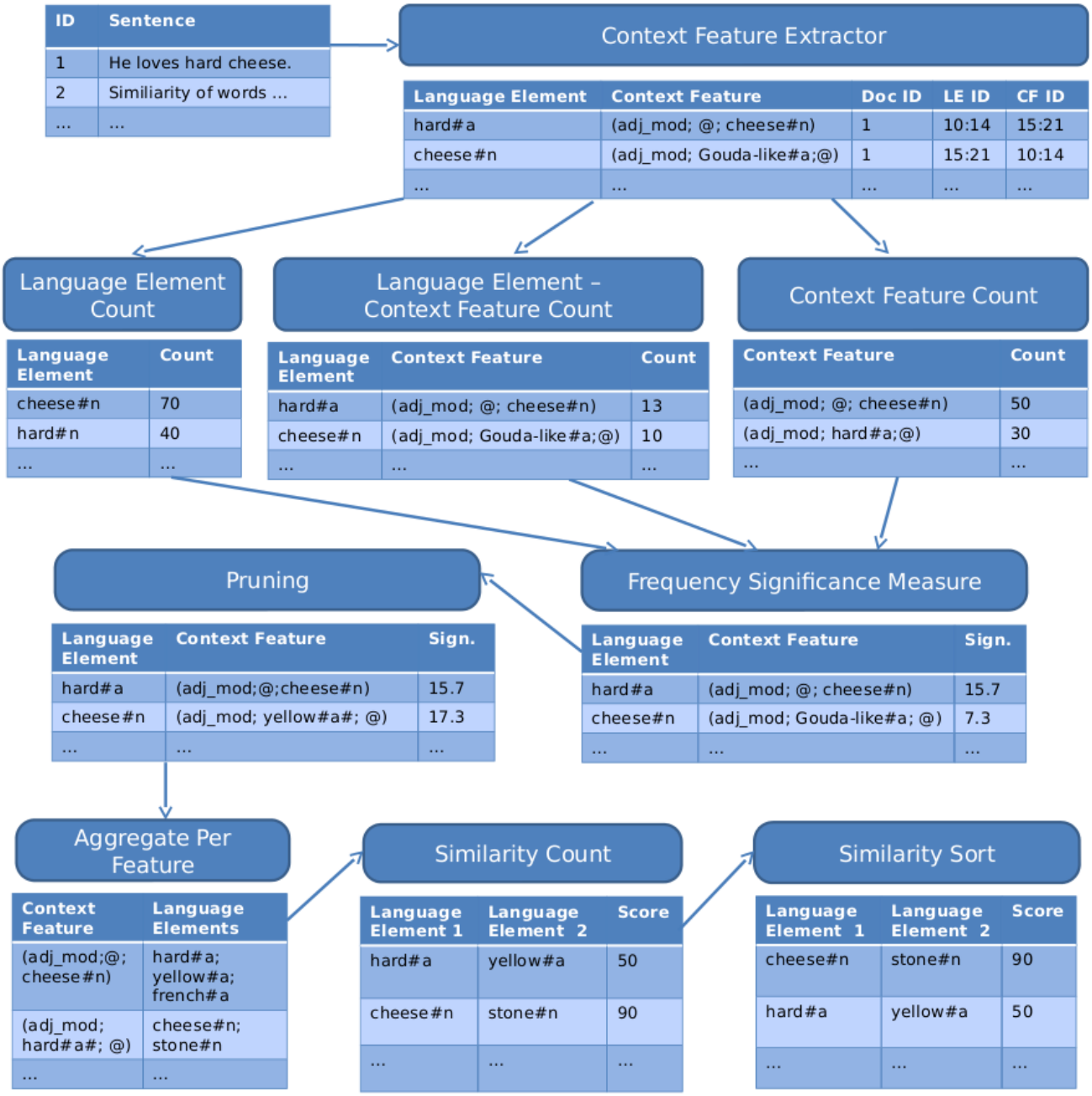

Figure 3.2: Processing pipeline of the JoBimText framework [Biemann and Riedl, 2013] used to compute Distributional Similarities. While the original version is implemented on Hadoop's MapReduce architecture, we in this thesis used a functionally nearly identical Spark implementation to allow for faster prototyping of additional components.

3.5 Feature Selection

Intuitively, certain context features of a word are more representative than others. For example, the context *grilled bass* is fairly descriptive, while *pale bass* is not, though definitely not impossible to appear within a corpus of larger size. With this intuition in mind, we are able to reduce the amount of features used to represent every word, by simply discarding unrepresentative contexts. This, in turn, facilitates

scalable computation of distributional similarities, by reducing data complexity. In fact, previous experiments have shown that up to a certain number, adding distributional features does not increase quality of the computed thesaurus [Biemann and Riedl, 2013], when compared to WordNet-based similarity scores. Choosing the number too high in some cases even decreases quality slightly. In the same vein, we in Chapter 6 show that using too many features produces an unfair bias in similarity scores towards more frequent words. Therefore, feature selection is not only a technique that allows for more scalable thesaurus computation, but is crucial to achieving high thesaurus quality.

First of all, as already mentioned earlier, reducing the number of features per word directly is done using the parameter p: For every word-feature pair, we calculate a significance score and keep only the p most significant features per word. For an overview of the used significance scores, see Table 3.3. Also, we cut off features with a word co-occurrence frequency below a certain threshold t_{wf}, which is usually set to relatively low values such as 2 or 3. This already significantly reduces noise; most context features co-occur only 1 time with a given word, a co-occurrence that is trivially insignificant.

Frequent words have more features

As briefly mentioned above, thesauri with a high number of features per word tend to have an unfair bias towards more frequent words. This is simply due to the fact that the more frequent a word is, the higher is the chance of it co-occurring with any given feature. Higher frequent words hence come with an inherent larger number of features. If p features are used to compute the similarity between a highly frequent word w_1 and an infrequent word w_2, and w_2 has $p_2 < p$ features in total, then the similarity between w_1 and w_2 is bounded by $\frac{p_2}{p} < 1$. For example, if word w_2 has 234 features, the similarity computation uses $p = 1000$ features, and w_2 shares all its features with w_1, then the similarity between these two is only 0.234, not 1. Yet, a higher frequent word with more than 1000 features that shares only a third of these features with w_1 will receive a higher similarity score. It is therefore crucial to find an optimal number of features that provides ”enough” contextual information, and at the same time does not overly prefer higher frequent words.

This frequency discrepancy between two words is in fact very common in a corpus of natural language. According to Zipf’s law, a few words of natural language text usually make up a substantial part of the corpus. For example, within the Brown corpus, the word *the* makes up 7% of the entire corpus [Gençay and Fagan, 2011]. Similarly, other common words (*be*, *a*, etc.) also make up a large part of the corpus. Conclusively, the remaining part of the corpus is split up among the other words (∼1 mio.), which yields a very broad distribution of word frequencies. This is an example of a *power-law distribution*: given words from a natural language text, sorted by their frequency, the frequency of two following words in this list drops significantly.

While for some applications, it is acceptable that words are more similar to words of equal frequency, this is not the case for the application of WSI. Since the sense clustering is only performed on n most similar words, this would lead to uncommon senses being cut off. Conclusively, p should be chosen low enough to be "fair" to less frequent words, without sacrificing too much accuracy by cutoffs.

Computation complexity

Another reason to reduce quantity of features is the complexity of the distributional similarity computation algorithm. For every feature that two words share, the algorithm must "emit" (as in the MapReduce paradigm) and later increment a word similarity count (for an in-depth introduction into the implementation employed, see Chapter 9). Given the number of words w_f that share a specific feature f, and assuming that w_{max} is maximum of this value among all features appearing in the corpus, the memory and time complexity of the computation algorithm is therefore $O({w_{max}}^2)$. The most straightforward way to control w_{max} is to specify a threshold above which features are cut off. However, this potentially

cuts off important features that are necessary to indicate similarity between two terms that are indeed related. Another way to control w_{max}, that circumvents this problem, is to cut off unrepresentative (i.e. "unimportant") features for every word as previously mentioned. This is usually done by ranking features according to their significance for a specific word, using one of the significance measures listed in Table 3.3. This leads to fewer shared features among words, reducing the overall time complexity of the similarity computation.

Significance Measures

As previously pointed out, selecting representative context features is crucial to obtaining meaningful word similarities using a Distributional Semantics model. To rank distributional features according to their significance for a specific word, several measures have been proposed in the literature:

Pointwise Mutual Information (PMI) is based on probability ratios: Given two events A and B, it states whether the likelihood of A increases with the incidence of B. Since the actual value of PMI is the logarithm of this ratio, it equals 0 when the events are statistically independent, i.e. have no mutual information. It is positive when they co-occur more often than by mere chance, and negative if they co-occur less often than by chance. In the latter case, the incidence of A could be interpreted as a "contra indicator" of B. PMI was first introduced to NLP applications by [Church and Hanks, 1990].

Since PMI is known to have a strong bias towards scoring low-frequency items higher than more common items, it has a major in the application for our purposes: Since features selected by the PMI measure tend to have a low frequency, they are shared by fewer words and are less suitable for computing word similarities. *Lexicographer's Mutual Information (LMI)* [Kilgarriff et al., 2004], also known as Local Mutual Information [Evert, 2005], tries to compensate the problem of this observation by multiplying the score by an co-occurrence frequency. This effectively mitigates the bias towards less frequent features, while still assigning a low score to insignificantly co-occurring features.

The *Log-likelihood ratio (LL)* [Dunning, 1993] is a significance measure that, when applied to ranking features for computing distributional similarities, performs almost equally to LMI [Biemann and Riedl, 2013]. Due to its lengthy definition, we simply point the reader to [Bordag, 2008].

Significance Measure	Definition
Pointwise Mutual Information	$\text{PMI}(A,B) = \log_2\left(\frac{n\ n_{A,B}}{n_A\ n_B}\right)$
Lexicographer's Mutual Information	$\text{LMI}(A,B) = n_{A,B}\ \text{PMI}(A,B)$

Table 3.3: Significance measures used to rank features per word. n_A and n_B denote the frequencies of event A and B, resp. $n_{A,B}$ denote their joint frequencies, and n is the total number of observations.

In this chapter, we outlined a method to compute Distributional Similarities among words using dependency features. To reduce noise, and to make the computation highly scalable, we kept only the most significant features per word based on the Lexicographer's Mutual Information (LMI) measure. Chapter 4 discusses a possible approach to induce word senses based on these word similarities, and compares this approach to other WSI techniques.

4 Word Sense Induction

As outlined in the previous chapter, two popular approaches for WSD are the use of either *knowledge-based* disambiguation methods or of *supervised* methods. Both of these methods suffer from a *Knowledge Acquisition Bottleneck* [Wagner, 2006]: They both assume that a tremendous amount of background knowledge (i.e., sense inventories or sense-labeled corpora) is readily available, which is only the case for popular languages such as English, Spanish or German, and only for narrow domains. In this chapter we discuss another possibility for WSD, which is fundamentally different from other approaches in that it does not require any manually annotated training data or hand-crafted resources: Word Sense Induction (WSI). In the literature, this method has also seen other names, including *Word Sense Induction and Discrimination* systems or *corpus-based unsupervised* systems [Agirre and Soroa, 2007].

Instead of relying on pre-defined sense inventories, these approaches *induce* word senses from a raw, unstructured text corpus. This is done using some form of clustering of words, either on vector space models or on graphs. Both models are somewhat similar, in the sense that vector space models can be represented as graphs by connecting nodes (the words) with edges weighted with the word's similarities in the vector space and cutting off edges below a certain threshold. In fact, graph representations are often created in exactly this way, as will be shown later.

In addition to WSI methods that induce word senses from word similarities, others cluster *contexts* or *instances* of words. In essence, these methods merely *discriminate* word senses rather than identifying specific senses from a sense inventory [Schütze, 1998]. In other words, they group instances with a similar meaning, without labeling this group in any way. Hence, no explicit word sense inventory is computed. Some of these approaches, however, induce an *implicit* (or hidden) sense inventory from a fixed, large background corpus. These can come in the form of prototypical vectors in a vector space representing each sense cluster, or also in the form of discovered latent topics in an underlying theoretical topic model. Some approaches, on the other hand, induce a clustering specific to a set of instances, i.e. these have no explicit or implicit set of fixed word senses. Such word-sense discriminations are hard to interpret, and a missing reference sense inventory often hinders re-use of such systems [Navigli, 2009].

One of the goals of our work is therefore to implement a WSI system that induces a fixed, explicit word sense inventory. We achieve this by clustering word-similarity graphs constructed from a Distributional Thesaurus. Before we describe this approach in detail, we briefly introduce other current state-of-the-art methods for computing word-sense clusters.

4.1 Vector Space Models

Vector space models are among the most studied models used for WSI. They represent a word or word instance by a high-dimensional vector of contexts, where the definition of 'context' can vary greatly. In the following we list a small portion of algorithms that can compute sense clusters in such a vector-space model.

Single-link clustering, *complete-link clustering* and *average-link clustering* are related forms of a *hierarchical* clustering. All of these have in common that they construct an initial set of trivial clusters from the individual elements in the vector space (here, words or instances) and iteratively merge similar clusters. Each iteration therefore creates a new level in a cluster hierarchy. In parlance of these clustering methods, cluster similarity is called cluster *cohesion*. The difference among these implementations lies in the way clusters to be merged are chosen: single-link clustering chooses the two clusters with the

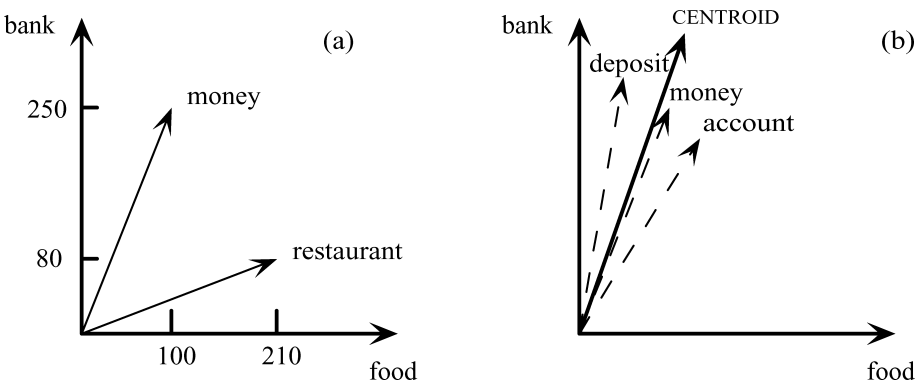

Figure 4.1: (a) One of many ways to represent words as vectors: *money* and *restaurant* are represented in two dimensions, each quantifying the co-occurrence frequency with the words *bank* and food, resp. (b) Shows a *context vector* for *stock*, represented as the centroid of the vectors of words appearing in the context. Illustration taken from [Navigli, 2009, p. 27].

minimum pair-wise distance of elements and complete-link clustering the two clusters with *maximum* pair-wise distance. Average-link clustering in contrast to these takes all elements of two clusters into account to compute cluster cohesion, by aggregating all of their element's pair-wise distances. Generally speaking, hierarchical clustering approaches like the aforementioned are considered to produce high quality clusters, however their application on large datasets is limited due to their quadratic time complexity [Steinbach et al., 2000].

Clustering by committee (CBC) [Pantel and Lin, 2002] is a two-staged clustering algorithm. In a first stage, it uses average-link clustering to find small and tight (i.e. fine-grained) sense clusters of highly similar words. These clusters are then input to a second stage, which iteratively identifies *committees* from these clusters: This is done by marking a small number of these clusters as committees until the remaining clusters are above a certain similarity threshold to one of these committees. The committee identification step is repeated until for every word, there is a committee with mean vector "close enough" to the word's average context vector (i.e. the similarity between mean vector and word vector is under a certain threshold). The result is therefore a minimal number of committees (senses) so that every word is represented well by one of these, determined by similarity threshold parameters. In a manual evaluation, Pantel and Lin found CBC to outperform other clustering methods when applied to WSI, including k-means and average-link clustering. In this evaluation, WordNet synsets served as gold senses.

$$\underset{s}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{4.1}$$

k-means clustering uses *prototype vectors* to represent clusters. The underlying assumption of this method is that an optimal clustering (i.e. the k prototype vectors) minimizes the squared distance of all elements in the vector space to their closest prototype vector. This measure is called the *within-cluster sum of squares (WCSS)*. For a definition, see equation 4.1; here S_i and μ_i denote sense clusters and corresponding prototype vectors, resp. A common algorithm implementing k-means is *Lloyd's algorithm*, which is a variant of an expectation-maximization (EM) algorithm. A common variant of this implementation starts by first picking k random elements in the vector space as initial prototype vectors[1] and then in an *assignment step* assigns each element to a cluster so that the WCSS is minimized. An additional

[1] This initialization is called the *Forgy* method. Strictly speaking, there are also other initialization methods like a random partitioning of all elements into k initial clusters. Lloyd's algorithm itself does not specify which initialization to use.

update step chooses new prototype vectors to be the mean of these updates clusters. These two steps are repeated until the clustering has stabilized. Often, k-means is considered to produce clusters inferior compared to hierarchical clustering approaches (like average-link clustering), although its time complexity that is linear in the number of elements in the vector space makes it applicable to larger clustering problems [Steinbach et al., 2000].

$$P(w|d) = \sum_{z=1}^{T} P(w|t=z)P(t=z|d) \tag{4.2}$$

Latent Dirichlet allocation (LDA) is a type of topic model. The conceptual idea of topic models is that documents in a collection contain a number of *latent* (hidden) topics. The probability of seeing each individual word in these documents varies depending on the topic. Therefore, a topic model contains a probability distribution of each word over the different topics $P(w|t=z)$. Also, it defines a probability distribution $P(t=z|d)$ of seeing a specific topic given a specific document. The assumption of using a topic model is that the probability of seeing a specific word w, given a document d, is then modeled by these probability distributions (see equation 4.2). LDA is a type of topic model that requires a specific number of topics k to be specified. In contrast to this, the *Hierarchical Dirichlet Process (HDP)* [Teh et al., 2006] is an extension of LDA that automatically induces the number of topics from the data.

4.2 Graph Clustering

For a given word, a list of the most similar words can contain useful information regarding the meaning of this word and possibly reveal multiple senses. For example, consider the word *tablet*. Depending on the meaning, top similar words may be *notebook*[2], *manuscript*, *headstone*, *smartphone* or *pill*, forming at least three distinct senses. When we put these words in a *word graph* and link words that are related, senses can be regarded as a *clustering* of this word graph: words belonging to the same sense are connected (related) to each other, while words from different senses are often unconnected. See Figure 4.2 for an illustration of this example. This sense induction process using word graphs is specifically similar to methods previously applied in [Dorow, 2007; Biemann, 2007; Hope and Keller, 2013a]. Also [Widdows and Dorow, 2002] applied a similar clustering method on word-cooccurrence graphs to induce word senses. The clustering process applied by our system is further explained in the following.

4.2.1 From Global Graphs to Local Neighborhood Graphs

To compute such a clustering, we started with a *global word graph* $G = (W, E)$, where W is a specific set of words from our text corpus, e.g. all nouns, and E a set of edges connecting related words. More specifically, this relatedness is based on distributional similarities from a thesaurus T, computed as described in Chapter 3.3: In G, we linked two words $w_1, w_2 \in W$ if w_2 is among the N most similar words to w_1, according to T. N and other parameters can be chosen by the user. Sense clusters are then determined for every word separately. These words are in the following called the *target word*. The pseudocode of the sense induction algorithm, including the following steps, is listed in Algorithm 2.

To compute these sense clusters, we did the following: for every target word $w_i \in W$, a subgraph $G' = (V, E')$ is constructed that consists of all neighbors V of this word (i.e. its N most similar words, therefore, $|V| \leq N$). The target word itself is not part of this subgraph. Words in G' are again connected

[2] Note that *notebook* can here refer to both a portable computer as well as a note-taking book, therefore being ambiguous itself.

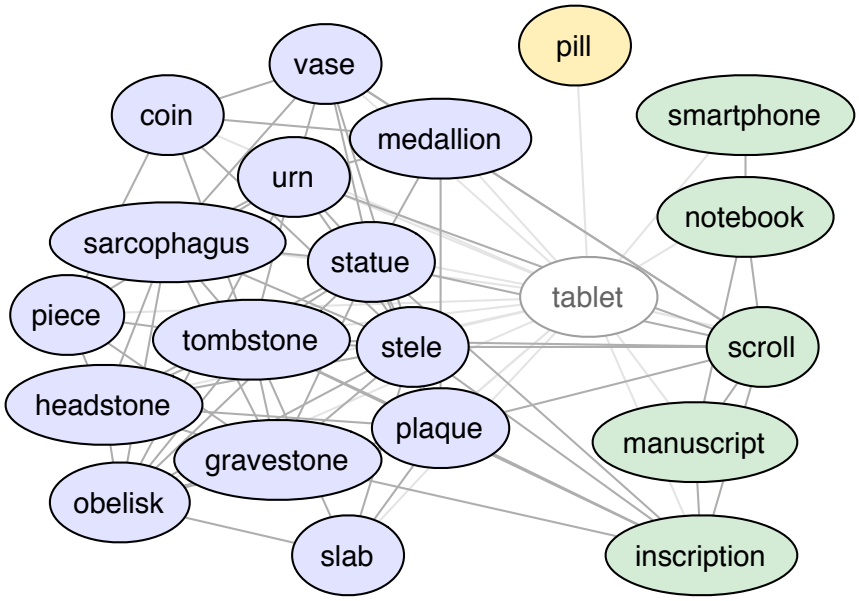

Figure 4.2: Illustrative, small neighborhood and possible clusters of *tablet*. Note that the word *tablet* itself is not part of this neighborhood graph, and only shown to better illustrate the construction of the neighborhood graph.

to their n most similar words within G' [3]. In the ideal case, each sense of the target word, represented by the similar words belonging to the sense, forms a highly connected cluster that shows only few connections to other clusters (i.e. senses). Figure 4.2 shows an exemplary neighborhood graph for the word *tablet*, constructed from the global graph G. If we remove the word *tablet* itself, such as in Figure 4.4, we can see the strong tendency that words with similar senses tend to be connected among each other, while having less connections to words from other senses, therefore forming clusters.

Before we come to the specific clustering algorithms, we can therefore observe that there are at least two parameters for our WSI algorithm. First of all, the *number of neighbors* (similar words) to choose for a target word w, which we refer to as parameter N. Second, the *maximum number of connections* one of these neighbors v is allowed to have within this subgraph, which we refer to as n. The latter, n, is bounded by N, as nodes in the subgraph cannot have connections to nodes that are not in the subgraph. To be more specific, the subgraph operation first retrieves all outgoing edges of each v that point to other nodes within the subgraph. Of these, only the top n similar words are kept. n is therefore only an upper bound for the number of outgoing edges of a $v \in V$, e.g. for $N = 200$ and $n = 100$, there will be many out of the 200 subgraph nodes that have less than 100 outgoing edges within the subgraph. Figure 4.3 shows a neighborhood subgraph generated for $N = 10$ neighbors of *tablet* with a maximum of $n = 3$ connections. Note that many nodes have less than 3 connections (and some have more, see next paragraph for details).

The "Random Walker" as Conceptual Foundation of Graph Clustering

To compute clusters in this graph, we (as shown later in detail) utilize the notion of a "random walker": If a walker starts at an arbitrary node w, and randomly follows paths in the graph, at which node(s) is it most likely to end up? If there is a high probability that this walker ends up at nodes $v_1, .., v_k$ then these nodes should, if possible, be part of the same cluster. More specifically, we in this thesis assume that the word graph is undirected. If the graph contains directed edges, e.g. from w_1 to w_2, this would

[3] This graph is a true subgraph of G, as an edge in G' can only exist if there is such an edge in G.

Algorithm 2: Pseudocode of our sense induction algorithm.

input : T // distributional thesaurus
W // set of terms in T to cluster
N // subgraph size
n // subgraph connectivity

output: for each term w_i in W, a clusterings S_i of its N most similar terms

```
G ← graph with terms from T as nodes, initially no edges
foreach w ∈ W do
    K ← N most similar terms of w according to T
    Add to G an edge from w to every k ∈ K, weighted by the resp. similarity
end
foreach w_i ∈ W do
    V ← direct neighbors of w_i in G
    G' ← graph with V as nodes, initially no edges
    foreach v ∈ V do
        Add to G' the n highest-weighted edges in G between v and other terms in V
    end
    S_i ← MCL(G')
end
```

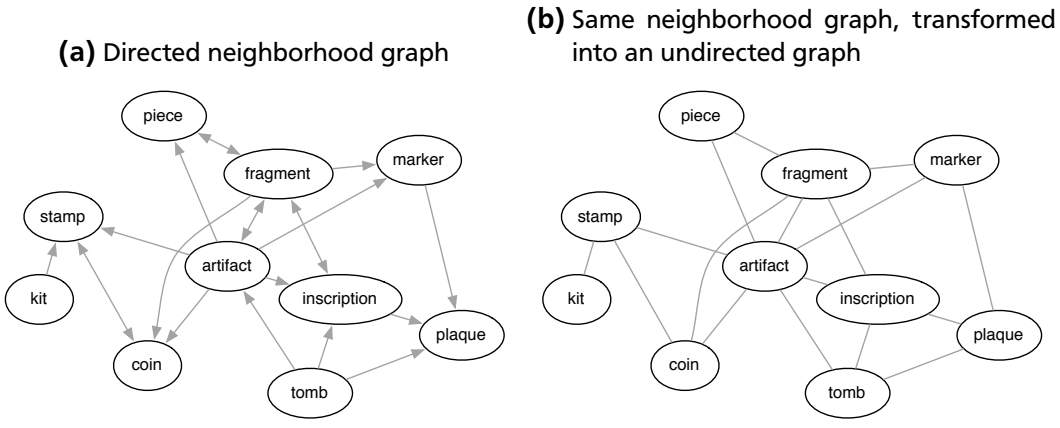

Figure 4.3: Neighborhood graph of word *tablet*. Parameters for building the neighborhood subgraph were $N = 10$ and $n = 3$. This neighborhood graph corresponds to the variable G' in Algorithm 2

imply that a random walker can walk from w_1 to w_2, but not the other way around[4]. While this could have beneficial effects on the resulting sense clusters, it is out of scope of this thesis to find this out. Also, undirected edges are due to their symmetric nature more intuitive and thus easier to comprehend. We will in the rest of this thesis therefore assume that all word graphs are undirected.

Directed vs. Undirected Subgraphs

If you look at Figure 4.3 (b) again, you will notice that *inscription* has more than 3 connections. Since the graph connectivity parameter n chosen in this example is 3, this might come as a surprise. To explain this, have a look at Figure 4.3 (a). This is the actual directed subgraph that is produced by the subgraph operation as explained above. First we should note that the similarity measure employed by our system is symmetric, i.e. if $sim(a, b) = s$ then $sim(b, a) = s$ and vice versa. However, since only the top n edges per node are added to the subgraph, the following happens. Given $n = 1$, for each node there will be only one outgoing edge. However, since nodes may appear as the most similar words in multiple other words in the subgraph, this node will nonetheless have multiple ingoing edges. If we simply transform every directed edge into an undirected edge, this will lead to some nodes having more than n (undirected) connections.

Limitations

If a word sense has many substitutable words or synonyms, then this sense tends to dominate the list of similar words. Since only the top (e.g. 200) similar words are taken into account to build a word graph, this can effectively eliminate other word senses. An example is the word *spring*, which can have at least three senses: a *source of water*, a *device* (e.g. made of spring steel) or the spring *season*. The latter here dominates the thesaurus and makes up somewhere around 90- 95% of the thesaurus entries for this word (mostly words representing some period of time, like *1970*, *afternoon*, *mid-september*, or even *election*). The result is that similar items of ”device” sense do not even appear in the constructed word graph, and can therefore not be discovered by graph clustering methods.

4.2.2 Markov Chain Clustering

Markov Chain Clustering (MCL) [van Dongen, 2000] is a graph clustering algorithm based on Markov Chains. Given is a graph of which its edge weights are considered to be transition probabilities between two nodes. Its core idea is that of a “random walker”: Starting at a given node and following a random path according to these probabilities, the algorithm stochastically determines in which part (cluster) of the graph the walker is most likely to end up. The underlying principle of this is that nodes tend to have more connections to nodes within its cluster than to other nodes. Algorithm 3 shows the pseudocode of MCL.

The two central parts of the algorithm are the *inflation* and the *expansion* steps in each iteration. While the expansion operation allows “flow to connect different graph regions” [van Dongen, 2000, p. 6], the inflation operation emphasizes and deemphasizes current. The parameter γ belonging to these operations therefore controls the granularity of the resulting clusters.

Another parameter that, according to van Dongen, has similar effects on the granularity, is the pruning threshold p. Every edge with a transition probability below this value is removed. Like γ, this results in stronger contrast between high and low edge weights, resulting in a different granularity of the clusters.

[4] In terms of the CW algorithm (see Section 4.2.3), this means that w_2 can “receive” a label from w_1, but not vice versa.

Algorithm 3: Pseudocode of MCL algorithm. The pruning step in line 6 is merely optional to speed up convergence.

input : A // adjacency matrix of graph
γ // inflation parameter
p // pruning parameter
r // maximum residual
output: list of clusters

1 add self-loops to A;
2 normalize rows in A;
3 **while** *maximum residual in* $A > r$ **do**
4 | expand A by power of 2;
5 | inflate A by γ;
6 | *prune entries in* $A < p$;
7 **end**
8 interpret A as clustering;

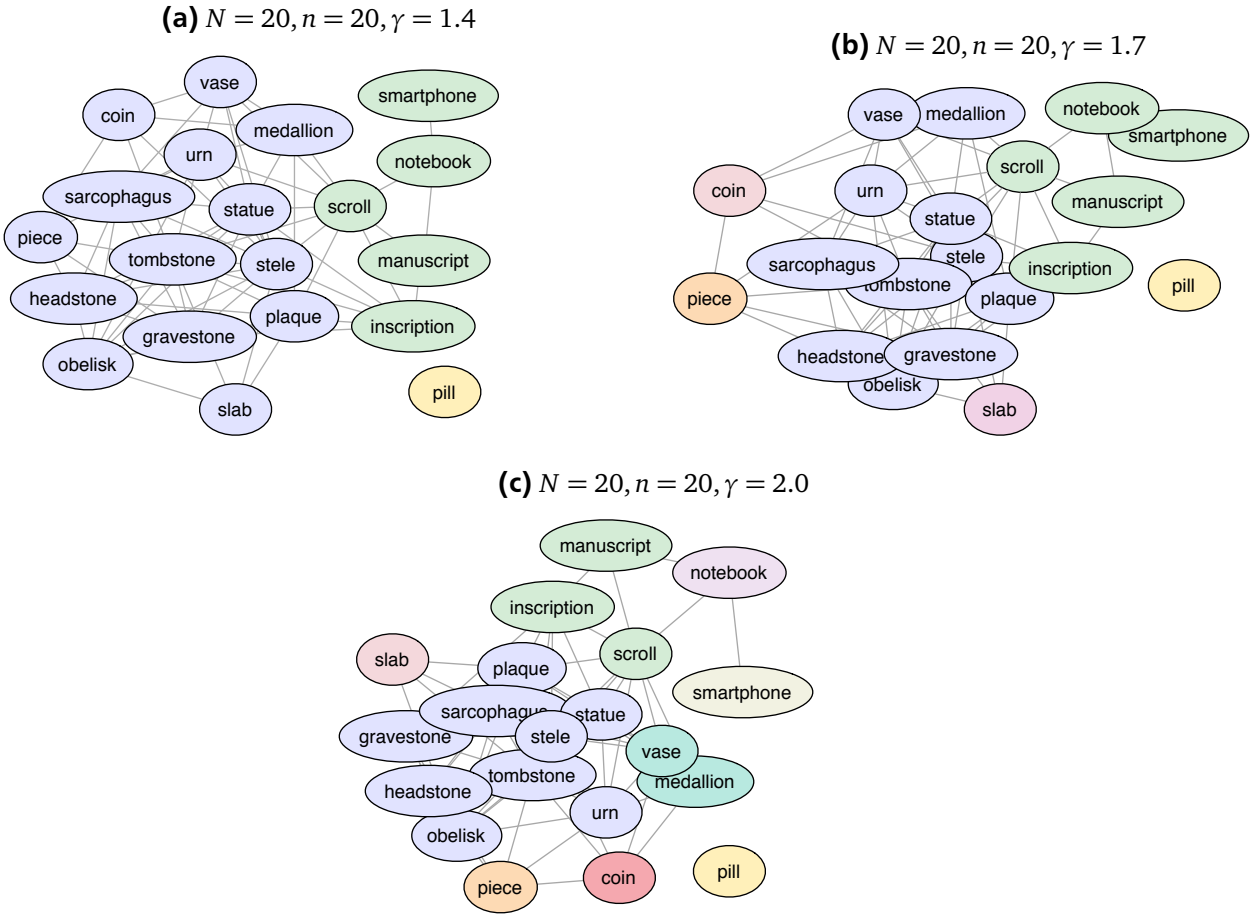

Figure 4.4: Clustering of neighborhood graph of *tablet* using MCL. The clustering granularity parameter (γ) is varied, other parameters are fixed. As a result, the number of clusters increases from 3 ($\gamma = 1.4$) to 9 ($\gamma = 2.0$).

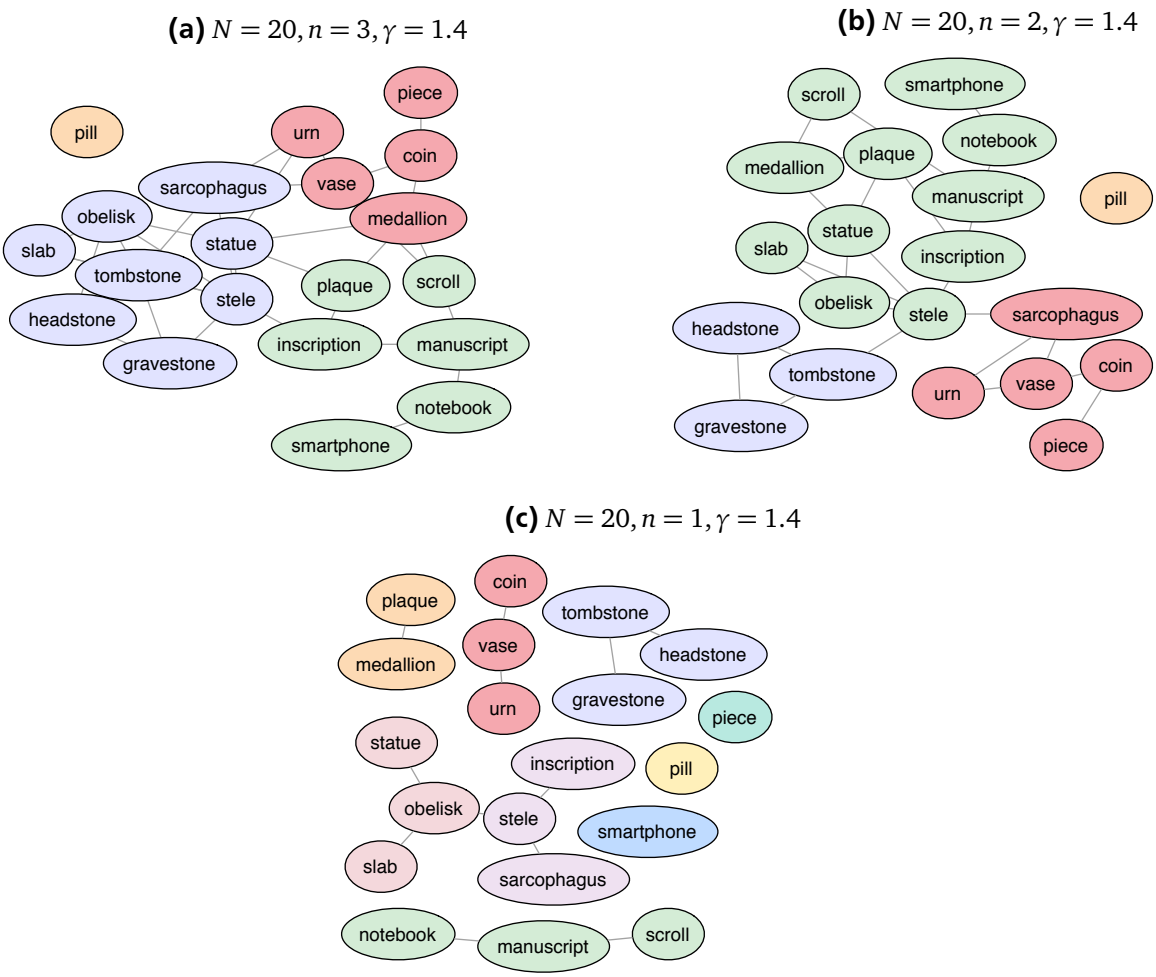

Figure 4.5: Clustering of neighborhood graph of *tablet* using MCL. The maximum number of neighbors per word (n) is varied, other parameters are fixed. As a result, the number of clusters increases from 4 ($n = 3$) to 9 ($n = 1$).

4.2.3 Chinese Whispers

Chinese Whispers (CW) [Biemann, 2006] is a special case of MCL. It takes a more drastic optimization approach than pruning does for MCL: instead of pruning column values in A larger than a specific threshold, it keeps only the largest column value after each iteration. This is equivalent to "labeling" each node in every step with the label of the dominant node in its neighborhood. This is also the reason why this algorithm is called "Chinese whispers": It aims at finding groups of nodes that "broadcast the same message to their neighbors" [Biemann, 2006].

The worst-case time complexity of CW ($O(k * |E|)$) equals that of MCL ($O(k * n^2)$) if the graph is fully connected, in which case $|E| = n^2$. However, due to the sparsity of word graphs, this complexity is much lower in practice. Also, the CW algorithm can be implemented more efficiently, as it does not need to copy the transition matrix in every iteration: to propagate node labels, it is sufficient to keep the current label per node stored separately from the transition matrix. Especially with a high number of iterations with only a few operations each, this dramatically effects the run time.

In contrast to MCL, CW itself is parameter-free, making it well suited for the unsupervised discovery of word senses, where the number of senses is not known in advance. However, it is indeterministic: as it

randomizes the order in which it propagates class labels, it does not guarantee that the outcome of the clustering is the same if CW is applied on the same graph a second time.

Algorithm 4: Pseudocode of CW algorithm.

```
input  : A graph G(V, E)
output: A class assignment class(v) for all v ∈ V
forall the v_i ∈ V do
    class(v_i)=i
end
while changes in class labels do
    forall the v ∈ V in randomized order do
        class(v)=highest ranked class in neighborhood of v;
    end
end
```

4.2.4 MaxMax: A Soft-Clustering Algorithm

MaxMax [Hope and Keller, 2013a] is a *soft-clustering* algorithm that is time-linear in the number of edges. It is a two-stage algorithm in that it first transforms the weighted, undirected input graph (G) into an unweighted, directed graph (G'). The resulting graph is then clustered by finding maximal *quasi-strongly connected* (QSC) subgraphs.

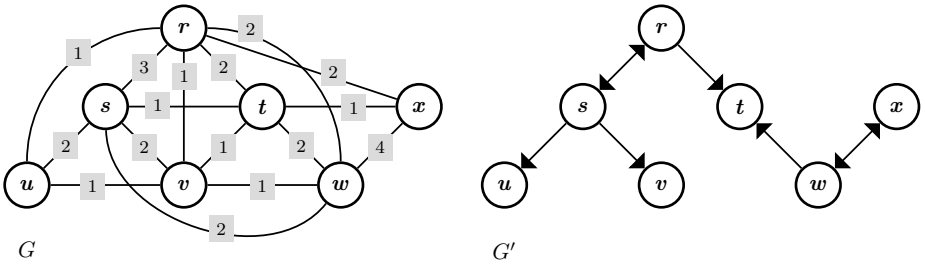

Figure 4.6: Example of a word graph G and its transformation into an unweighted, directed graph G' in preparation for application of the MaxMax algorithm. The graphic is taken from [Hope and Keller, 2013a]. This transformation can be regarded as a special case of our method of pruning the local word neighborhood subgraph (see Section 4.2.1).

For the first stage, it uses the principle of maximal affinity to transform G into an unweighted, directed graph G': The word v with highest edge weight $w(u, v)$ to a word u is defined to have maximal affinity to u. Consequently, v is considered the *maximal vertex* (with vertex being the generalization of a word) to u. The resulting graph G' contains an edge from v to u if v is a maximal vertex of u. Note that this transformation is effectively equivalent to a special case of our pruning method for the local neighborhood (see Section 4.2.3) of a word in the global word graph, with n set to 1. The only exception here is that that directed edges are not transformed into undirected edges, as done by our clustering algorithm.

In a second step, the MaxMax algorithm identifies quasi-strongly connected subgraphs in G', which are the resulting sense clusters. A QSC is simply a subgraph that contains a node (the *root*) from which every other node in the subgraph can be reached, i.e. there is a path from the root to every other node in the subgraph. For example, identified roots in graph G' (see Figure 4.6) are r and w, as every other node in their respective subgraphs can be reached following a path starting at these roots. The clustering algorithm performs a soft clustering, as nodes that are leafs of a resulting QSC subgraph can at the same

time be leafs of multiple such subgraphs. An example of this is node t in Figure 4.7. Since these nodes are by definition part of all such QSC subgraphs, they are assigned to all respective clusters.

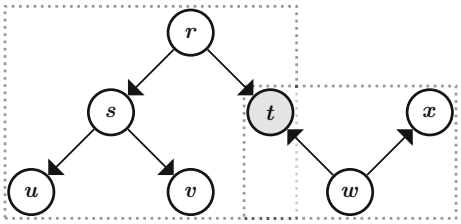

Figure 4.7: Clustering result of the above example graph G using the MaxMax algorithm. The graphic is taken from [Hope and Keller, 2013a]. The node t is an example of a *soft-clustered* node: it is part of both resulting clusters.

This algorithm is similar to the clustering method that we utilized for two reasons: First, as mentioned above, its graph transformation step is essentially a special case of our graph pruning step. Secondly, our notion of a *random walker* in the graph to be clustered can also be applied here: A walker that starts in one of the resulting clusters and follows a random path will always end up in the same cluster, as there is by definition no path from one QSC subgraph to another. However, we see one problem with this approach: Since every node is only allowed to have a directed edge to the node with highest affinity, this comes with an inherent loss of information. More specifically, in a word-clustering application, this means that a word is only represented by a single other, most significantly related word. In contrast, we represent every word by *several* related words, which avoids an early pruning of word senses represented by other related words. However, the soft-clustering capability of MaxMax has the benefit that e.g. the word *notebook* as similar word of *tablet*, as illustrated in Figure 4.2, can be assigned to both a *computing* device sense (which would in this case be formed together with *smartphone*) as well as a sense referring to a *note-taking* device.

4.3 State-of-the-Art Systems

There are various systems today that perform WSI in some form or another. A lot of them differ strongly: Some are instance-based clustering systems, a few others induce a sense inventory on a word graph. Almost all systems include a WSD component, with the exception of the system of Pantel and Lin. Furthermore, both vector-space models as well as graph-based models were utilized. Some of these systems participated in WSI evaluations, such as the SemEval challenge.

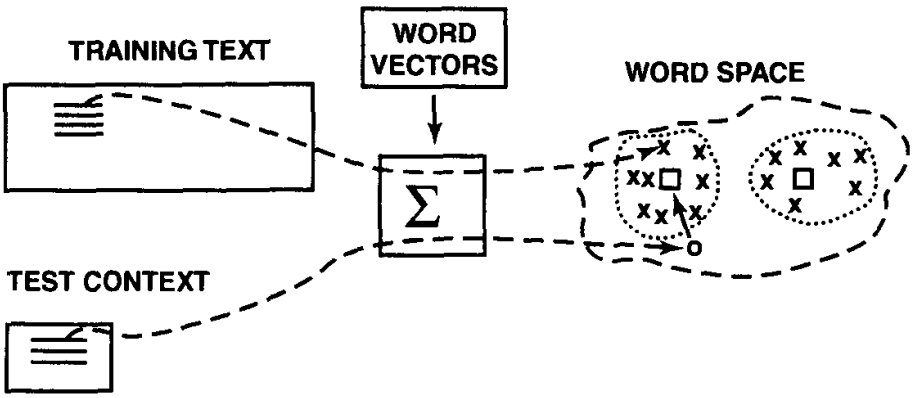

Figure 4.8: Basic design of Schütze's Word Sense Discrimination system.

The oldest system presented here is that of [Schütze, 1998] (see Figure 4.8). They extracted word co-occurrence frequencies from 17 months of New York Times news text, with roughly 435 megabytes or 60.5 million words of text. Based on these co-occurrence features, word vectors were constructed and reduced to 100 dimensions using Singular Value Decomposition (SVD). This word vector space was then used to compute word clusters using the k-means algorithm. These clusters represented their induced word senses. Test instances were also drawn from the New York Times corpus and word senses were discriminated by choosing the cluster with the mean that is closest to the test instance's context vector. Schütze specifically distinguished this approach, which he referred to as word-sense *discrimination*, to WSD as the discovered word senses were merely abstract clusters in this word space, and never labeled in any way. On an own evaluation dataset of 20 ambiguous words, they reported an accuracy of 94%, beating a MFS baseline[5].

Another more recent system is *HyperLex* by [Véronis, 2004]. They first selected 10 polysemous nouns that were found to be hard to sense-tag for humans [Véronis, 1998]. Then, they compiled a sub-corpus for each word from web pages using the meta-engine Copernic Agent[6]. From this corpus, co-occurrence frequencies were extracted. A co-occurrence word graph was built by drawing edges between co-occurring words, and weighting these with $\big(1 - P(w_1|w_2)\big)$, where $P(w_1|w_2)$ is the conditional probability of seeing word w_1 if we already observed word w_2. Edges with weight below 0.9 are discarded, where a value of 0 means that two words are always associated; reversely, 1 indicates that two words are never associated. Root "hubs" in this graph are identified and interpreted as word senses. Disambiguation is performed by computing the distance between context words and root hubs in this graph; the root hub with the lowest average distance to the context words is then picked as the best-matching sense. In an own evaluation, their system received 97% precision and 82% recall, compared to a 73% precision baseline. The test dataset contained 100 contexts for each of the 10 words, i.e. 1000 contexts in total. These were manually checked by a single expert to receive the precision/recall values. 7 out of 10 words were tagged with 100% precision. A problem we see here is that no comparison to standardized evaluation testsets were performed. Also, all evaluation was manually done by a single human expert, therefore no agreement to other annotators was measured. Lastly, compiling a sub-corpus for every word would be impractical in an all-word sense induction setting, which we aim to enable with the system described in this thesis.

[Pantel and Lin, 2002] introduced *Clustering by Committee (CBC)* as a new vector-space clustering method. They represented each word by a feature vector of context features such as bigrams (for example *"sip _"* as context feature for wine). Pointwise Mutual Information (PMI) was used as value for the individual features. To extract context features, they parsed about 144 million words (1 gigabyte) of English newspaper text from the TREC collection[7]. Their system does not implement a sense disambiguation or discrimination component. However, they evaluated the word clusters produced by their system against WordNet: They first automatically mapped clusters to WordNet senses, and then determined a precision as the ratio between the number of WordNet senses that were discovered and the actual number of senses that are found in WordNet. Recall measured the ratio between the correct number of discovered senses for a word, and the number of senses in WordNet. In this evaluation, their clustering algorithm out-performed others like k-means and average-link clustering.

The *AI-KU* system [Baskaya et al., 2013] is based on lexical substitution and vector-space clustering: In a first step, it for each instance identifies the 100 most probable lexical substitutes of the respective target word. This is done using a probabilistic 4-gram model that is computed from the ukWaC background corpus, which in turn is realized using Fastsubs [Yuret, 2012], an algorithm that finds the n most probable lexical substitutes. The substitute word vector of each instance is considered to be a distributional representation of the same in a 100-dimensional vector space. The values of the dimensions of this

5 Overall score of the baseline was not explicitly reported.

6 `http://www.copernic.com` (last accessed: 2015/04/30)

7 This collection consisted of a 1988 AP Newswire, a 1989-90 LA Times, and a 1991 San Jose Mercury.

context vector are equivalent to the probabilities of the individual substitute words, normalized to sum up to 1. All test instances, represented by these vectors, are finally clustered using k-means. The system participated in the SemEval-2013 WSI task, and received the overall highest scores in the WSD setting of the evaluation. Here Baskaya et al. picked 22 as value for k, knowing that the test set contained words with 3 to 22 senses.

Unimelb [Lau et al., 2013] is a system based on the Hierarchical Dirichlet Process (HDP) [Teh et al., 2006], a topic model. Except for the application it is entirely indifferent to the system already described in [Lau et al., 2012]. Since Lau et al. participated in the SemEval-2013 WSI task, they also used the ukWaC corpus to train their model. The HDP model they used is a parameter-free extension of LDA, specifically not requiring a number of topics to be specified, and estimated using an implementation of Gibbs sampling. To reduce the computational complexity of the topic model estimation, they limited the number of training instances to 5% and 50000, each of these limitations constituting a system configuration. In contrast to many other topic models, their topic model is based on positional co-occurrence features (3 words left, 3 words right, with positional information) additionally to "ordinary" co-occurrences (i.e. BOW context representations) previously used in such models. Latent topics discovered in the training instances (specific to every target word) were interpreted as possible word senses. To assign word senses for a specific word context, they conclusively also interpreted the probabilistic assignments of these topics for this context as the respective sense probabilities. In the SemEval-2013 WSI task, their system's performance was competitive, notably it produced the best Fuzzy B-Cubed score in the cluster comparison setting of the evaluation.

The *University of Sussex (UoS)* [Hope and Keller, 2013b] presented a system that is based on dependency-parse features and the MaxMax graph clustering algorithm (see Section 4.2.4). WSI was performed on the dataset provided by the SemEval-2013 WSI task. Based on the dependency-parsed ukWaC corpus, they for each target word constructed a graph consisting of the 300 highest-ranked words found in a dependency relation with this target word. To compute a similarity-like score between two words w_1 and w_2, and to determine such a ranking, they counted the number of times these words co-occurred in a dependency relation. The actual score is then the Normalized Pointwise Mutual Information (NPMI) measure of these co-occurrence frequencies. The same score is used to weight edges in this word graph. This graph is then clustered using the MaxMax algorithm. In a post-processing step, the resulting fine-grained clusters are merged based on a degree of *cohesion*[8] and *separation*[9], with the intuition that two clusters are merged if they have a high semantic similarity. Disambiguation of instances is performed by assigning the sense with the lowest separation score between the instance's context words (minus the head word) and the words of the sense cluster. The performance of this system in the SemEval-2013 WSI task was competitive, however it was outperformed by other systems in all 6 evaluation settings (namely by *AI-KU* and by *Unimelb*).

Differences to our system

Arguably the system presented here that is most similar to ours is *UoS*: Their clustering method is graph-based, and their step for constructing a word graph for each target word is similar to ours (cf. Section 4.2.4). They also used dependencies as context features for each target word, however there are many subtle differences: First, they only counted the number of times two words co-occurred in any dependency relation, ignoring the type of this relation (i.e. subject vs. object of a clause, etc.). Also, edges in the word graph were weighted by the NPMI of these co-occurrence frequencies, whereas our system uses similar significance measures to select various word-characteristic features (i.e. dependency relations) that serve as comparison between two words. Edges in our word graph are therefore the

[8] Cohesion of a set of words is here defined as the average NPMI score between all words in this set.

[9] In essence, their separation score is a dissimilarity measure between two sets of words: It is the inverse of the weighted overlap, where this weight is the NPMI of the two respective words.

number of shared features, not the significance of co-occurrences. Also, their disambiguation component selects word senses based on an overlap between context and a sense's cluster words, whereas we perform disambiguation by using a probabilistic model.

A system presented here that does implement sense assignments based on a probabilistic model is *Unimelb*: For each context feature of a target word, it estimated a probability distribution of the discovered latent topics over this feature. These distributions over context features are aggregated to determine the most likely topic of an instance of the target word (which is not included in the instance's context). However, the way topic models are computed poses a fundamental difference to concepts utilized in our system, i.e. Distributional Similarity and graph clustering.

Furthermore, one of the questions that guided us through this thesis is whether we can induce a *fixed* word-sense inventory, to enable linking this inventory to other, pre-defined lexical or ontological resources like WordNet or FreeBase. We found that some of these presented systems may be adapted to facilitate such a linking step, however not all of them. For example the *Unimelb* system discovered latent topics for each target word from a fixed set of training instances that were extracted from a background corpus. In their approach, these topics were equivalent to the words's senses. Here, it would be interesting to assess whether the words most representative of each latent topic may be used to label this topic and to link it (i.e. the respective word sense) to other sense inventories. However, such a linking step is infeasible for some other presented systems. The *AI-KU* system can here be used as a counter-example: It induces word senses from a vector space that is constructed individually for a specific set of instances. Since these found clusters therefore change every time the system is applied to another set of instances, it would be of little benefit to link these clusters to other sense inventories.

To summarize, we in this chapter presented a method to automatically induce word senses based on a word similarity graph. This word similarity graph was constructed by using Distributional Similarities computed as described in Chapter 3. Since we also need means to assign induced senses in context, the next chapter discusses a novel approach for Word Sense Disambiguation on induced sense inventories.

5 Word Sense Disambiguation for Induced Sense Inventories

A key challenge in the construction of a fully-unsupervised WSI system is the disambiguation of induced senses in context. To pick up the example from the introduction, Table 5.1, lists induced senses for the word *bass* as produced by our system.

Sense	Top cluster words
bass.0	funk, jazz, music, blues, reggae
bass.1	guitar, drum, saxophone, percussion, piano
bass.2	trout, catfish, perch, pike, eel
bass.3	tenor, baritone, alto, soprano, orchestra
bass.4	vocal, accompaniment, chorus, riff, rhythm

Table 5.1: Induced senses of word *bass*. These can be interpreted as referring to bass music[a], an instrument, a fish species, an adult singer with low voice, and a low voice itself, resp. (cf. WordNet senses listed in Figure 2.1)

[a] `http://en.wikipedia.org/wiki/Bass_music` (last accessed: 2015/04/30)

Now also reconsider the following example context, for which the disambiguation step of our system has to choose an appropriate sense from the list above:

> *He decided to have grilled* ***bass*** *for dinner.*

While to the human reader, it is obvious that this context refers to sense **bass.2**, this is impossible to deduce for a disambiguation algorithm, given only the information from Table 5.1. Specifically, there is no overlap between the context and any of the sets of cluster words from the individual senses. Therefore, simple disambiguation methods like *Lesk*-based algorithms will have no success when applied to this context. However, this example is objectively unambiguous, given common-sense knowledge like *people grill fish, but (usually) not instruments*.

Other approaches that exploit information that is available in the underlying Distributional Thesaurus include the utilization of similarities among sense clusters and other context words [Rada et al., 1989; Véronis, 2004; Hope and Keller, 2013a]. More specifically, Hope and Keller utilize a so-called *separation score* to grade induced senses based on context words and words from the individual sense's clusters.[1] Also, for a simpler notion of this separation score, see [Biemann, 2010]: they induced sense clusters on a graph constructed from sentence-based co-occurrences of a target word and used the resulting clusters directly to compute an *overlap* between context words and sense cluster words. The sense cluster with the highest overlap is then assigned in this context.

These systems therefore utilize the similarity of context words to the target word for disambiguation. While this works well with word similarity scores that express some form of semantic relatedness (such as *fish* being closely related to *dinner*), they would produce inferior results for similarity scores that are based on syntactic similarity: *grilled* as an adjective, due to its inherent syntactic difference to the

[1] Hope and Keller used the same score to merge fine-grained sense clusters produced by the MaxMax algorithm. Their *UoS* system participated in the SemEval-2013 WSI task, with mixed results (see Chapter 7).

noun *fish*, would likely receive a score lower than any unrelated noun in the context, confusing a purely similarity-based disambiguation algorithm. Since our dependency-parse features, due to their local and grammatically motivated nature, yield rather syntactically similar items, this disambiguation method is therefore unsuited for use with our Distributional Thesaurus.

5.1 Obtaining Sense-Tagged Training Instances for Learning a Disambiguation Model

The same observation however motivated us to a new disambiguation approach that exploits the nature of syntactically similar items: First, we formulated the assumption that syntactically similar words are, to some extent, grammatically substitutable. This assumption is undermined by a few illustrative examples in the following. Based on this assumption, it is possible to extract a wealth of contextual instances *sensitive to a specific word sense*, by using the context instances of the syntactically most similar cluster words from each sense cluster and (virtually) replacing the cluster words with the target word (i.e. *bass* in our case) itself.

Cluster word	Context
trout	Most • such as lake • live in freshwater lakes and/or rivers exclusively.
catfish	• have inhabited all continents at one time or another.
perch	The general body type of a • is somewhat long and rounded.
eel	There are two styles of grilled •, the topic of which is covered more precisely under kabayaki.

Table 5.2: Actual contexts of words belonging to sense cluster **bass.2**, as found in Wikipedia, which served as text corpus for our computations. Cluster words are replaced by '•' to indicate their position and for visual evidence of a certain substitutability of the cluster words and the target word *bass*.

From a standpoint of judging semantic *correctness* of the extracted contexts, these are obviously inaccurate to some extent, e.g. it is untrue that most *bass* appear exclusively in freshwater environments. However, the merely partial validity of the extracted contexts is sufficient for our purpose of sense disambiguation. In effect, these instances provide a large amount of sense-tagged *training instances* we may use for learning a disambiguation model.

For example, as can be seen in Table 5.3 we can extract several properties applicable to the sense referring to a fish species that are perfectly valid. Notably, these properties can be extracted in the same manner that we previously used to extract distributional features for computing the thesaurus. For example, the properties from the table may all be deduced by extracting corresponding dependency features as listed in the right-hand column of the same table.

Property	Context feature
bass.2 can *live*	`nsubj(live, •)`
bass.2 may *inhabitate* places	`nsubj(inhabitate, •)`
bass.2 may *have a (body) type*	`prep_of(type, •)`
bass.2 may be *grilled*	`amod(•, grilled)`

Table 5.3: Properties characterizing *bass* in the sense of a fish species, extracted from contexts of words from the sense cluster **bass.2**. Cluster words are replaced by '•'. Note how the observed properties can be represented using the same distributional context features that were previously used to compute distributional similarities.

Note that these properties are much more accurate then their corresponding word co-occurrences:

*The hammering sounds of his drum and **bass** music literally **grilled** the speakers.*

This context means bass as a type of music, i.e. **bass.0**, however it mentions the adjective *grilled*, which makes it difficult for a purely co-occurrence based WSD approach to choose the appropriate sense.

A disadvantage of using only dependency-based context features for disambiguation is their *sparsity*. On average, every context contains only 1-3 dependency features for a given word, e.g. the single feature `nn(music, •)` for the example above. These few features may provide strong clues for disambiguation, but fail to provide such clues for certain contexts, such as *Bass is a type of fish*. The latter example contains only the dependency feature `nsubj(type, •)` for the word bass, therefore providing no clear hint towards the fish species, though the context does contain the word *fish*. To mitigate this problem, and to leverage the strengths of both dependency-based, as well as co-occurrence-based disambiguation, we decided to use co-occurrences as *complementary*, wide-coverage features to the low-coverage, but high-precision dependency features we outlined above.

5.2 Scalable Aggregation of Context Clues

By explicitly generating sense-tagged training instances for every ambiguous word (by replacing the cluster word, e.g. *eel* with the ambiguous word itself, e.g. *bass*), it is possible to train classifiers for a small number of ambiguous words. However, it is impractical to do so in an all-words disambiguation setting in which the goal is to induce a sense inventory and disambiguation model for all relevant nouns, verbs, adjectives and adverbs found in a specific text corpus. The run time of this approach is high: For every unique word, it requires scanning though all similar terms in each sense cluster, i.e. 100 words in our setting, retrieving all sentences containing these words, replacing their occurrences with the ambiguous target word.

While this approach could prove to be useful for providing additional training data to existing, supervised WSD systems such as SVM-based disambiguation models, it is not necessary for our purposes. In fact, we can directly utilize the word-feature co-occurrences that we previously extracted for the computation of a distributional thesaurus. Obtaining a list of context features specific to a sense cluster of, e.g. *bass*, is then a matter of collecting all context features across all words belonging to this sense cluster.

5.3 Scoring Sense Clusters

Given a context C, there are various possibilities to compute a score for each sense cluster s_i, based on these context features (F_i). Notable implementations of a scoring function $w(s_i)$ include:

1. Counting the context overlap: $w(s_i) = \left|F_i \cap C\right|$
2. Weighting every overlapping feature individually: $w(s_i) = \sum_{f \in (F_i \cap C)} \pi_f$
3. Approximating the conditional probability of s_i in the context: $w(s_i) = P(s_i|C)$

While the first is easiest to implement, it suffers from the inability to distinguish between highly descriptive and less descriptive features. For example, the adjective *quite* may be used to describe fish, however it is much less of a clue for this sense than the adjectives *wet* or *streamlined* when a classifier has to decide between sense **bass.0** (bass music) and **bass.2** (fish species). The second scoring function attempts to overcome this problem, however it still lacks the notion of modeling what we are actually looking for: the conditional probability of seeing a sense s_i, given a context C.

It is infeasible to directly capture this probability, by counting how often a sense appeared with *exactly* this context C. Therefore this probability may in practice only be estimated. Namely, the *Naive Bayes* assumption allows a critical simplification: Assuming all features are statistically independent, the probability of a class (i.e., sense) may be computed using only individual probabilities that depend on single features.

More formally, assume that we are given a given a word $\mathbf{w}$ with $\mathbf{K}$ induced senses. Each induced sense $\mathbf{s_k}$ with $k = 1 \dots K$ has $\mathbf{I_k}$ number of cluster words. Also, every cluster word $\mathbf{v_{k,i}}$ with $i = 1 \dots I_k$ has a word frequency $\mathbf{f(v_{k,i})}$. The total number of word occurrences in the text corpus is $\mathbf{N}$. Lastly, we are given a context $\mathbf{C = c_1 \dots c_J}$ of size $\mathbf{J}$.

Using the chain rule and independence assumption that every context feature c_j is independent from the others, we can simplify the joint probability $P(s_k, C) = P(s_k, c_1, \dots, c_J)$ in order to compute $P(s_k|C)$:

$$
\begin{aligned}
P(s_k|C) &= \frac{1}{P(C)} P(s_k, C) && (5.1)\\
&= \frac{1}{P(C)} P(s_k, c_1, \dots, c_J) && (5.2)\\
&= \frac{1}{P(C)} P(s_k)\ P(c_1|s_k)\ P(c_2|s_k, c_1)\ \dots\ P(c_J|s_k, c_1, \dots, c_{J-1}) && (5.3)\\
&= \frac{1}{P(C)} P(s_k)\ P(c_1|s_k)\ P(c_2|s_k)\ \dots\ P(c_J|s_k) && (5.4)\\
&= \frac{1}{P(C)} P(s_k) \prod_{j=1}^{J} P(c_j|s_k) && (5.5)\\
& && (5.6)
\end{aligned}
$$

The WSD task is then equivalent to a word-specific function assigning a sense index to every context:

$$
\begin{aligned}
sense_w(c_1, \dots, c_J) &= \underset{k \in K}{\operatorname{argmax}}\ P(s_k) \prod_{j=1}^{J} P(c_j|s_k) && (5.7)\\
& && (5.8)
\end{aligned}
$$

Estimation of Sense Frequencies And Joint Sense-Feature Frequencies

Since we cannot directly count any sense frequencies $f(s_k)$ or joint sense-feature frequencies $f(s_k, c)$ for a context feature c from our text corpus (there are no explicit sense tags), we have to estimate these frequencies using the information available to us. As outline above, we can utilize an implication of our syntactic similarities: since two similar words are assumed to be substitutable, we can assume any occurrence of a cluster word $v_{k,i}$ to be interchangeable with an occurrence of s_k. The result is a (merely theoretical) parallel corpus where any occurrence of $v_{k,i}$ is replaced by a new word ${s_k}^*$. This word can be considered a "distributional extension" of sense s_k, and vice versa. The frequency of ${s_k}^*$ is then given by $f({s_k}^*) = \sum_i^{I_K} f(v_{k,i})$. The same principle can be applied analogously to determine a joint frequency $f({s_k}^*, c)$.

The probability of ${s_k}^*$ is then given by

$$P(c|{s_k}^*) = \frac{P({s_k}^*, c)}{P({s_k}^*)} \tag{5.9}$$

$$= \frac{f({s_k}^*, c)}{f({s_k}^*)} \tag{5.10}$$

$$= \frac{\sum_i^{I_K} f(v_{k,i}, c)}{\sum_i^{I_K} f(v_{k,i})} \tag{5.11}$$

This relatively simple formulation has a major drawback: It weighs cluster words proportionally to their frequency, and therefore overly "prefers" more frequent words. To elaborate, consider this example: Given a word w with a sense cluster s consisting of exactly two other words v_1 and v_2. Their frequencies are $f(v_1) = 10{,}000$ and $f(v_2) = 1{,}000{,}000$.[2] Also, they both co-occur with a certain feature c 5000 times, therefore $f(v_1, c) = f(v_2, c) = 5000$. Their conditional probabilities, given the feature c are therefore $P(v_1|c) = 0.5$ and $P(v_1|c) = 0.005$. The resulting cond. probability of s^*, given the feature c, is therefore $P(s^*|c) = (5000 + 5000)/(10{,}000 + 1{,}000{,}000) = 10{,}000/1{,}010{,}000 \approx 0.01$, which is much closer to $P(v_2|c)$ than to $P(v_1|c)$. Hence, v_2 had a much stronger influence on the probability than v_1.

A solution is to normalize the joint frequencies with respect to the word's frequency, i.e. divide the joint frequency by the word frequency. This is effectively equal to averaging the conditional probabilities

$$P(c|v_{k,i}) = \frac{f(v_{k,i}, c)}{f(v_{k,i})} \tag{5.12}$$

over all cluster words $v_{k,i}$. Therefore, an alternative is to compute a probability-like score that is based on the sum of individual probabilities:

$$\hat{P}(c|{s_k}^*) = {I_k}^{-1} \sum_i^{I_K} \frac{f(v_{k,i}, c)}{f(v_{k,i})} \tag{5.13}$$

The multiplication with the inverse of I_k is effectively the previous division by the sum of all cluster word frequencies (equation 5.11): Since they were normalized, all words appear with a theoretical frequency of 1. The sum of these is therefore equal to I_k, the number of cluster words. Again, note that this definition is not a true probability anymore (due to the frequency normalization), but rather a probability-like score. We will therefore in the following be speaking of scores, instead of probabilities, and denote this by the additional 'hat' above the score function $\hat{P}(\cdot)$.

This solves the problem of dominating higher frequency cluster words. However, we made another observation: As we cluster a large number of similar words (we found $l = 100$ to be a good setting in this task), there often is a high discrepancy among the similarities of these words to w. Conclusively, some words are more suited as substitutes for w than others. To capture this in our score function, we introduced an additional weighting coefficient $\omega_{k,i}$ that is equal to the similarity between $v_{k,i}$ and w. The factor ${I_k}^{-1}$ is therefore replaced by the division of the sum of all these weights:

[2] To be mathematically exact, these frequencies must be set relative to the total number of word and word-feature observations N_w and N_{wc}, resp. (cf. equations 5.19 and 5.20). However, for simplification purposes of this illustrative example, we drop this constant factor here, which does not effect the illustrated problem.

$$\hat{P}(c|s_k{}^*) = \Omega_k{}^{-1} \sum_i^{I_K} \omega_{k,i} \frac{f(v_{k,i}, c)}{f(v_{k,i})} \tag{5.14}$$

$$= \Omega_k{}^{-1} \sum_i^{I_K} \omega_{k,i} \, P(c|v_{k,i}) \tag{5.15}$$

$$\Omega_k = \sum_i^{I_K} \omega_{k,i} \tag{5.16}$$

The score resembling the prior probability of each sense is computed in a similar manner, again taking into account the weighting coefficient $\omega_{k,i}$. The actual probability of each word $v_{k,i}$ is simply equal to its frequency in the corpus:

$$\hat{P}(s_k{}^*) = \Omega_k{}^{-1} \sum_i^{I_K} \omega_{k,i} \, P(v_{k,i}) \tag{5.17}$$

$$= \Omega_k{}^{-1} \sum_i^{I_K} \omega_{k,i} \, f(v_{k,i}) \tag{5.18}$$

Finally, frequencies can be computed by counting the number of word occurrences, and word-feature co-occurrences, and dividing these by the total number of such observations (N_w and N_{wc}, resp.):

$$f(v_{k,i}) = \frac{\text{count}(v_{k,i})}{N_w} \tag{5.19}$$

$$f(v_{k,i}, c) = \frac{\text{count}(v_{k,i}, c)}{N_{wc}} \tag{5.20}$$

The Zero-Count Problem

According to Zipf's law, most words in a corpus appear very rarely. Conclusively, the chance of seeing new words in before unseen text is high. Unseen context words will, however, result in $f(s_k, c_j) = 0$, which would effectively out-rule every word sense with which it wasn't seen in the training corpus. Since a word sense may always appear in partly new contexts, without its meaning changed, this is not an adequate way to deal with unseen words to disambiguate word senses.

A possible solution is to add *smoothing* to $\hat{P}(c|s_k{}^*)$. We will refer to this value as α, which results in an alternative definition our score function:

$$\hat{P}_\alpha(c|s_k{}^*) = \hat{P}(c|s_k{}^*) + \alpha \tag{5.21}$$

$$= \Omega_k{}^{-1} \sum_i^{I_K} \omega_{k,i} \frac{f(v_{k,i}, c)}{f(v_{k,i})} + \alpha \tag{5.22}$$

	$\hat{P}(s_k{}^*\|C)$
Naives Bayes	$\frac{\hat{P}(s_k{}^*)}{P(C)} \prod_{j=1}^{J} \hat{P}_\alpha(c_j\|s_k{}^*)$
Maximum probability	$\frac{1}{P(C)} \max_{j=1\ldots J} \hat{P}(s_k{}^*\|c_j)$
Average probability	$\frac{1}{P(C)\,J} \sum_{j=1}^{J} \hat{P}(s_k{}^*\|c_j)$

Table 5.4: Functions imitating the conditional probability of the distributional extension of sense s_k. All three functions resemble a joint probability (given the context C) based on individual probabilities (given a context word c_j).

5.5 Merging of "fuzzy" Clusters

A fine-grained clustering is often beneficial as it allows to distinct between more different senses. In some cases these finer-grained senses are relatively easy to distinguish using their context clues, and thus increase WSD performance. However, in individual cases, sense distinctions might become too vague, even if they are correct. For example, in a specific fine-grained clustering setting, the word *magazine* is distinguished into the magazine as a *company* (the abstract sense), a journalist (the magazine as an individual actor) and a *newspaper/journal* (the magazine's publication). While with high-precision context clues such as dependencies, a distinction in context might be possible, it will be relatively difficult to do so with only co-occurrences as context clues. The result is a decrease in WSD performance.

A possible solution is to optimize a clustering for a specific type of context features, e.g. co-occurrences in a bag-of-words model. Similar to features characterizing individual words, aggregated co-occurrences can characterize a whole cluster as well. Analogously to computing distributional similarities between words, this can be used to compute similarities between clusters. Clusters that are too similar, with respect to e.g. their co-occurrences, may be merged. This comes at a cost of sense recall (due to fewer sense distinctions) in a WSD task, but usually showed to have a more positive influence on WSD precision. Notably, the coarse-grained result of merging a fine-grained clustering often differs from a "raw" coarse-grained clustering with a similar number of senses per word.

Word	s	Cluster words
magazine	0	label:0.123 program:0.122 company:0.122 ...
magazine	1	publisher:0.119 critic:0.098 blogger:0.097 ...
magazine	2	newspaper:0.323 publication:0.285 journal:0.264 ...

Table 5.5: Example fine-grained MCL clustering of *magazine*. Numbers denote similarity of cluster words to *magazine*.

5.6 Pseudocode & Example

Algorithm 5 lists the pseudocode of an implementation of the context clue aggregation phase. Note that for better readability, we simplified the mathematically exact notation used in Section 5.4. Input to this algorithm are a target word w, sense clusters for this specific word, counts extracted from a background corpus, and word similarities from a Distributional Thesaurus. The resulting output is a list of scored context clues.

For each sense k of w, and for each context clue c, the algorithm determines a score resembling a likelihood of seeing sense k given the context c. These are exactly the context clue scores used for disambiguation. To compute them, it iterates over all cluster words and determines a *weighted average* of all conditional probabilities $P(c|s)$ for each cluster word s. $P(c|s)$ here is the conditional probability of seeing context feature c given word s, according to a background corpus.

Algorithm 5: Algorithm performing context clue aggregation and scoring.

input : w // target word
$S_1, ..., S_K$ // sense clusters of a specific target word
C // set of all context features
$f(w)$ // word frequencies
$f(c)$ // context feature frequencies
$f(w,c)$ // word-context joint frequencies
$\text{sim}(w_1, w_2)$ // word similarities
output: sets $D_1, ..., D_K$ of scored context clues

for $k = 1...K$ **do**
  foreach $c \in C$ *with* $f(s,c) > 0$ **do**
    $P_{k,c} \leftarrow 0$ // score of sense k for context clue c
    $\text{sim}_{\text{total}} \leftarrow 0$
    foreach $s \in S_k$ **do**
      $P_{k,c} \leftarrow P_{k,c} + \text{sim}(w,s)\frac{f(s,c)}{f(s)}$ // add weighted $P(c|s)$
      $\text{sim}_{\text{total}} \leftarrow \text{sim}_{\text{total}} + \text{sim}(w,s)$
    end
    $P_{k,c} \leftarrow P_{k,c}/\text{sim}_{\text{total}}$ // weighted average of all $P(c|s)$
    Add $(c, P_{k,c})$ to D_k
  end
end

To illustrate the aggregation and scoring of context clues using an example, consider Table 5.6 which lists two possible, simplified sense clusters of the word *bass*. In the following, we want to disambiguate *bass* in the (illustrative) context *grilled bass*. In this case, the cluster words in Table 5.6 alone do not provide any useful disambiguation hints. Hence, we need to extract meaningful hints, i.e. context clues, from somewhere else. In our system, this is done using word-feature counts of the cluster words extracted from a background corpus[3]. These features may themselves be other words appearing in the context (i.e. word co-occurrences) or dependency features. However, this simplified example uses word co-occurrences for disambiguation only. Exemplary word counts and word-feature counts are listed in Table 5.7.

Sense	Cluster words
bass.0	fish:0.7, eel:0.4
bass.1	music:0.5, jazz:0.3

Table 5.6: Simplified example sense clusters for word *bass*.

Using these counts, and assuming that our background corpus consisted of a total of $N = 1000000$ words, we can compute *prior* scores for each sense and conditional scores for each sense given a specific feature[4]:

[3] In fact, the relevant counts are the same that are used in the sense induction step.
[4] For a detailed explanation of the involved equations, refer to Section 5.4

Word	Context	Count
fish	grill	210
eel	grill	90
music	grill	20
jazz	grill	48

Word	Count
fish	3210
eel	603
music	8903
jazz	5329

Table 5.7: Exemplary word and word-feature counts for cluster words from Table 5.6

$$
\begin{aligned}
P_{\text{bass.0}} &= \left(0.7 * \frac{\text{count(fish)}}{N} + 0.4 * \frac{\text{count(eel)}}{N}\right) / (0.7 + 0.4) \\
&= \left(0.7 * \frac{3210}{1000000} + 0.4 * \frac{603}{1000000}\right) / 1.1 \\
&= 0.002 \\
P_{\text{bass.1}} &= \left(0.5 * \frac{\text{count(music)}}{N} + 0.3 * \frac{\text{count(jazz)}}{N}\right) / (0.5 + 0.3) \\
&= \left(0.5 * \frac{8903}{1000000} + 0.3 * \frac{5329}{1000000}\right) / 0.8 \\
&= 0.007 \\
P_{\text{bass.0, grill}} &= \left(0.7 * P(\text{grill}|\text{fish}) + 0.4 * P(\text{grill}|\text{eel})\right) / (0.7 + 0.4) \\
&= \left(0.7 * \frac{210}{3210} + 0.4 * \frac{14}{603}\right) / 1.1 \\
&= 0.050 \\
P_{\text{bass.1, grill}} &= \left(0.5 * P(\text{grill}|\text{music}) + 0.3 * P(\text{grill}|\text{jazz})\right) / (0.5 + 0.3) \\
&= \left(0.5 * \frac{20}{8903} + 0.3 * \frac{48}{5329}\right) / 0.8 \\
&= 0.005
\end{aligned}
$$

With these scores, we can proceed to compute an overall context score for each sense. This is done by multiplying the prior sense score and the conditional sense scores for each feature, in the style of a Naive Bayes classifier. In essence, this assumes the computed scores to be approximations of the respective prior probabilities and conditional probabilities[5]. The overall scores of each sense for this context are thus given by

$$
\begin{aligned}
\text{score}_{\text{bass.0}}(\text{"grilled bass"}) &= P_{\text{bass.0}} * (P_{\text{bass.0, grill}} + \alpha) \\
&= 0.002 * (0.050 + 0.00001) = \mathbf{0.000100} \\
\text{score}_{\text{bass.1}}(\text{"grilled bass"}) &= P_{\text{bass.1}} * (P_{\text{bass.1, grill}} + \alpha) \\
&= 0.007 * (0.005 + 0.00001) = \mathbf{0.000035}
\end{aligned}
$$

where α is a smoothing added to avoid context clue scores of 0 causing an overall sense score of 0. According to these scores, **bass.1** is therefore the best match for the context *grilled bass*.

[5] Mathematically speaking, this is in fact not a valid way to estimate the joint conditional probability of a sense given a context. However, it is entirely sufficient for our purposes of scoring senses according to their appropriateness in a given context.

To sum up, we in this chapter described a novel approach to assign senses from an induced sense inventory in context. For this we used context features of cluster words, extracted from a large text corpus. A scoring method was proposed that computes an overall score for a sense cluster and a given context using these context features.

6 Evaluation of Word Similarities

Quality of the Distributional Thesaurus (DT) has a direct influence on the quality of the WSI outcome. For this reason, several simple evaluations were performed to assess the quality of the computed thesauri. The most essential questions with a high impact on the thesaurus quality are the following:

1. How many features should be used per word?
2. Which feature significance measure should be chosen?
3. Which cutoff parameters reduce noise "well enough" without sacrificing too much data?

6.1 Pseudo-Sense Injection

One of multiple possible criteria for a high-quality DT is the reduction of “noise” to a minimum. We will here refer to “noise”, in a strict sense, as errors in word-feature frequencies that are produced by inaccurate pre-processing results during tokenization, POS-tagging, lemmatization, or parsing. In a wider sense, also word-feature frequencies that are relatively low can be considered noisy as they are equivalent to only a few random samples from the underlying theoretical probability distribution. A possible solution that we will follow here is to use cut-off (threshold) parameters that allow to select only a subset of, ideally less noisy, features based on different criteria. Intuitively, we should be able to achieve a certain amount of noise reduction just by using thresholds for a minimum word-feature frequency. Also, since it was shown earlier that a high number of features per word overly prefers frequent words over less frequent words, we are interested in quantifying the effect of using fewer features on noise in word similarities. In fact, using less features per word also dramatically improves the processing speed of our similarity computation (cf. Section 3.5).

To be able to measure noise in a specific DT parameter setting, we performed a simple test that splits every word into two pseudo-senses.[1] To do so, every word occurrence in the reference corpus is randomly replaced by a placeholder for either one or the other “pseudo” sense. This is comparable to splitting the reference corpus into two equally large parts, and comparing the resulting thesauri. Given two similar words, the difference between the resulting similarities (two instead of one) allows quantification of specific types of errors.

The assumption here is the following: If there is a high difference between thesauri computed from both halves of the reference corpus, then the computed similarity scores do not capture qualities that are representative to these words well enough. The main objective of the evaluation is to minimize this error.

This can be compared to experiments of Jurgens et. al. in *Measuring the Impact of Sense Similarity on Word Sense Induction* [Jurgens and Stevens, 2011]. To assess the word sense discrimination capabilities of their system, they formed *pseudo words* by joining two different words into one and replacing all their occurrences with this pseudo word. The word sense induction and disambiguation model then had to determine which word was originally meant in a specific context. In our evaluation, instead of

[1] This can be compared to the pseudo *words* that Gale et. al. created by combining two semantically unrelated words into one, e.g. *ability/mining* [Gale et al., 1992]. They used these as unsupervised means to evaluate the capability of WSD systems to correctly split instances of these pseudo word into the two original subsets. In essence, we here perform the inverse of this operation: We create two pseudo *senses* from one single word, and evaluate the ability of our system to group these into one sense cluster.

joining two words into one, we split one word into two. As opposed to the latter evaluation, here the word sense similarity model has to determine that two words are equal, instead of determining that they are distinct. It should be noted that these two evaluations share similar ideas, but are fundamentally different. One such difference is that our evaluation is independent of the WSD process, and therefore allows to evaluate the WSI process in an isolated manner.

To compute this similarity error between a word v and a similar word w, with pseudo senses w_1 and w_2, we can use the absolute error

$$e_{abs} = \left| sim(v, w_1) - sim(v, w_2) \right| \tag{6.1}$$

or a relative error

$$e_{rel} = \frac{\left| sim(v, w_1) - sim(v, w_2) \right|}{max\left(sim(v, w_1),\ sim(v, w_2) \right)} \tag{6.2}$$

The result is the total error $\overline{e_{abs}}$ and $\overline{e_{rel}}$, resp., averaged over all words in the thesaurus. A "perfect" DT will have $\overline{e_{\{abs,rel\}}} = 0$. For example, $tablet_1$ (the first pseudo sense of "tablet") may have the similar words $(phone_1, 0.35)$ and $(phone_2, 0.39)$, which yields an absolute error of $e_{abs} = 0.39 - 0.35 = 0.04$ or a relative error of $e_{rel} = (0.39 - 0.35)/max(0.39, 0.35) = 0.04/0.39 \approx 0.1$ for this particular similar word.

However, there is one problem with using solely this measure: It is trivial to construct a DT that receives a perfect e_{avg}, yet is completely useless: This is e.g. possible by defining $sim(w_1, w_2)$ for every combination of all words to be 1, or any other value. Similar effects will be achieved by thesauri that assign only relatively low similarity scores (e.g. due to too many cutoffs) to decrease the absolute error, or only high similarity scores to decrease the relative error. To compensate this, we added a second measure that signals how well the computed DT is able to discriminate between different similar words: Given a word and a list of N similar words $w_1..w_N$, sorted in descending order by their similarity score, this measure is defined to be

$$d_{avg} = \sum_{i=2}^{N} (w_{i-1} - w_i)/w_{i-1},$$

which essentially is the average relative difference between any two terms appearing as "neighbors" in the word similarity list. A higher score means that the DT discriminates more between individual similar words. The aforementioned trivial similarity scoring functions that would "fool" our error measure and result in a perfect $e_{avg} \approx 0$ will receive a penalty in the form of a low discriminative score $d_{avg} \approx 0$. An ideal parameter setting therefore yields a DT with a low average error and a high average discriminative score.

Tables 6.1a and 6.1b show that:

1. A smaller value of t_{wf} in almost all tested cases outperforms larger values of t_{wf}.
2. The differences between $t_{wf} = 10$ and $t_{wf} = 20$ are negligible
3. For $p >= 500$, increasing t_w from $10,000$ to $100,000$ has a negligible effect on e_{avg} but significantly increases the DT's discriminative capabilities

(a) w=10,000

t_{wf}	p	$\overline{e_{rel}}$	$\overline{d_{rel}}$
2	200	0.3956	0.0753
2	500	0.3446	0.0806
2	1000	0.3069	0.0776
5	200	0.4083	0.0808
5	500	0.3577	0.0726
5	1000	0.3273	0.0629
10	200	0.4247	0.0730
10	500	0.3780	0.0591
10	1000	0.3459	0.0468
20	200	0.4245	0.0730
20	500	0.3778	0.0591
20	1000	0.3459	0.0468

(b) w=100,000

t_{wf}	p	$\overline{e_{rel}}$	$\overline{d_{rel}}$
2	200		
2	500		
2	1000		
5	200	0.4459	0.1259
5	500	0.3692	0.0983
5	1000	0.3279	0.0781
10	200	0.4370	0.1111
10	500	0.3798	0.0770
10	1000	0.3443	0.0564
20	200		
20	500		
20	1000		

Table 6.1: Table showing influence of different parameters on e_{avg} and d_{avg}

4. A higher p decreases the relative error e_{avg}, but also decreases the discriminative score d_{avg}. Since a higher p was earlier shown to prefer higher-frequency words over lower-frequency words, a good trade-off must be found especially for this parameter.

While these numbers allow conclusions on the impact of different cut-off parameters on the DT quality, there is no proof that higher or lower values for d_{avg} or e_{avg}, resp., do in fact yield a "better" DT. Therefore, these values are only one of many quality indicators, and DTs that performed worse in this test might still perform better in specific application scenarios.

6.2 Previous Evaluations on Quality of Distributional Thesauri

The effect of different parameter settings on the quality of distributional thesauri has been studied before. Biemann et. al. [Biemann and Riedl, 2013, p. 22] analyzed the effect of the chosen significance measure and the value of p in respect to the *WordNet Path Similarity* of 1000 infrequent nouns from the computed thesaurus. They concluded that $p = 100$ produces best for 1M sentences, $p = 300$ for 10M sentences and $p = 1000$ for 120M sentences. Differences are small though, so they recommend $p = 500$ to $p = 1000$ for "very large corpora". Also, they found that LMI produces best results when the thesaurus is compared to WordNet-based similarities. Noticeably, PMI showed to rank infrequent similar words higher, allowing for better discovery of infrequent senses. However, overall DT quality with LMI as significance measure was better.

In this chapter, we evaluated the effect of two different cut-off parameters for distributional features on noise in a Distributional Thesaurus. By splitting words up into two *pseudo-senses*, we were able to measure noise as a discrepancy of similarities received for these two (identical) senses. Surprisingly, we found that a low word-feature count threshold $t_{wf} = 2$ yields the lowest overall noise when only the most significant features are kept per word. We also found that in our setting, the maximum number of features p per word should be between 500 and 1000, as less features increase noise significantly. More features, on the other hand, reduce the ability of the Distributional Thesaurus to discriminate word similarities. Since we are also interested in evaluating the quality of the actual induced senses, Chapter 7 will discuss two different evaluations for WSID systems.

(a) $p = 200$

word	sim.	freq.
football	1.0	412441
soccer	0.495	51527
hockey	0.48	72730
basketball	0.455	108596
rugby	0.415	44181
baseball	0.39	91575
lacrosse	0.365	8199
cricket	0.35	63244
tennis	0.34	48840
volleyball	0.32	20794

(b) $p = 1000$

word	sim.	freq.
football	1.0	412441
basketball	0.43	108596
hockey	0.405	72730
soccer	0.394	51527
baseball	0.361	91575
rugby	0.327	44181
volleyball	0.306	20794
athletics	0.292	20278
cricket	0.29	63244
lacrosse	0.285	8199

Table 6.2: Similar words to *football*. Clearly visible is the stronger correlation of word frequency to word similarity with $p = 1000$ vs. $p = 200$. In this case, this leads to *soccer* receiving an (intuitively unfair) penalty due to its relatively low frequency.

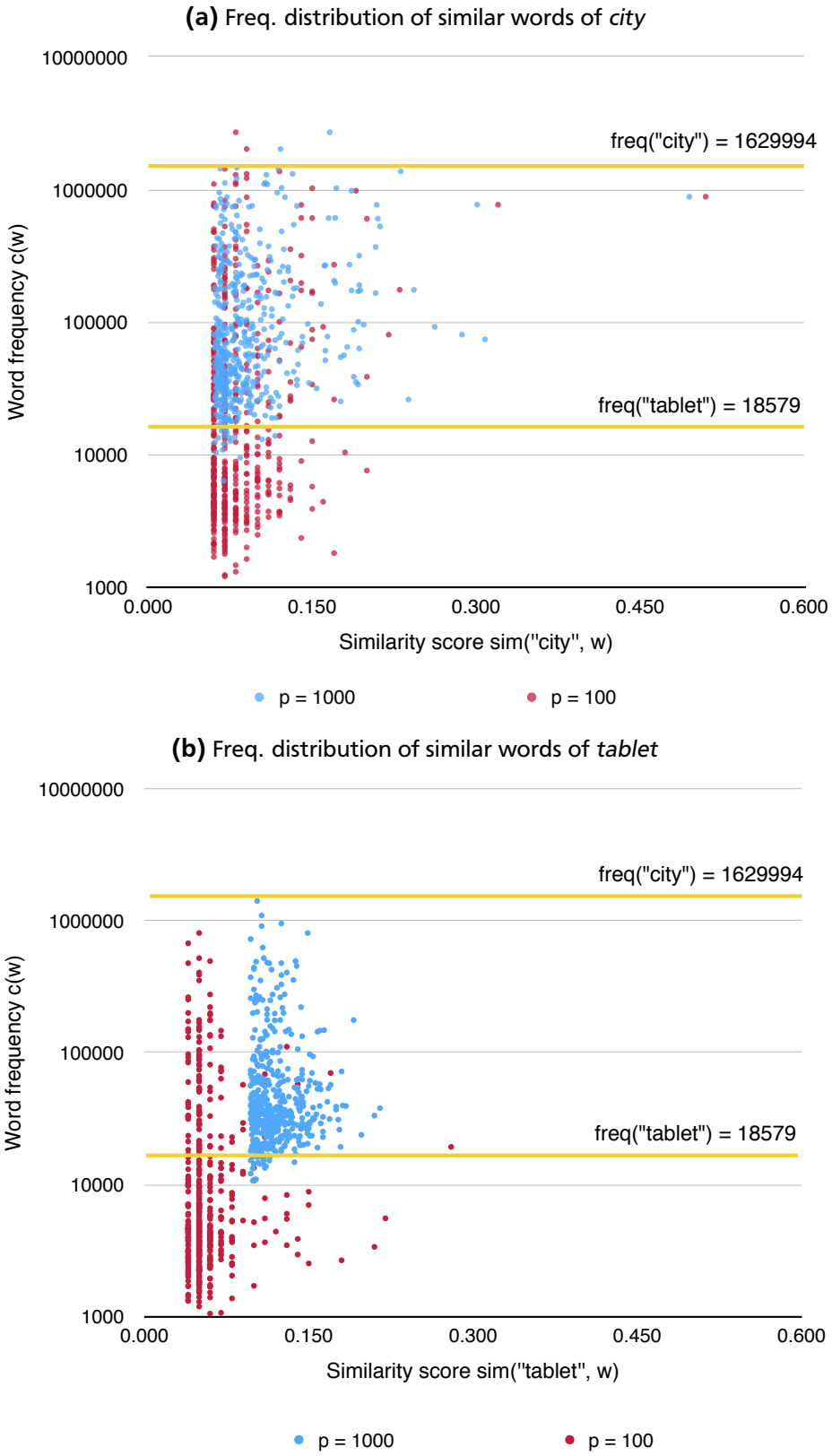

Figure 6.1: Two plots showing the correlation between word frequency and word similarity for the top 1000 similar words of a common word (*city*) and a less common word (*tablet*). The bottom graph clearly shows the bias towards more frequent words with a high (maximum) number of features p (here 1000 vs. 100). This is visible as a right-shift of the blue data points ($p = 1000$) compared to the red data points ($p = 100$), which means that frequent words are generally more similar to *tablet* than infrequent words.

7 Evaluation of Word Senses

The central theme in this work is the hypothesis that we can perform WSD on an *induced sense inventory* with performance comparable to that of existing instance-clustering WSI systems. This necessarily requires a standardized evaluation that has previously been used to assess the quality of such existing WSI systems. We therefore chose to evaluate our system against the dataset provided by the *SemEval-2013* WSI task, which will be introduced in section 7.2.

Additionally, to allow for optimization of the various parameters of our WSI algorithm, we extracted a large-scale WSD dataset from Wikipedia by utilizing its link structure. This specifically allowed us to evaluate on context instances from the same corpus that we used to both induce our sense inventory and learn a disambiguation model.

7.1 Two Methods for Evaluating WSI Systems

Often, two complementary types of WSI evaluation methods are used (e.g. [Agirre and Soroa, 2007; Manandhar, 2010]): a *mapping-based* method that maps induced senses to gold senses and then proceeds to evaluate in a traditional WSD setting, and a *clustering-based* method that compares the instance clusters formed by sense labels to a gold clustering.

Evaluation in a WSD setting

To compare sense labels of an induced sense inventory with manually-tagged sense labels from a reference lexical resources (such as WordNet), it is necessary to *map* each induced sense to a corresponding reference sense. This supervised evaluation framework has e.g. been described in [Agirre et al., 2006]. The mapping is usually done by splitting the WSD instances into two sets as e.g. done in [Manandhar, 2010]. The first split is sense-tagged by the WSI system, and a mapping between induced senses and reference senses is chosen that maximizes the labeling accuracy on this split when compared to the reference sense-tags. The accuracy of the sense labels on the second split, when converted to sense labels from the reference resource according to this mapping, is then used to measure the performance of the WSI system. The actual performance is then measured according to various different scores. The following three are specifically used in the SemEval-2013 WSI subtask.

The *Jaccard Index* measures the degree of agreement between two sets of sense labels X and Y for an instance. It is defined as $\left|\frac{X \cap Y}{X \cup Y}\right|$.

Given two rankings of sense labels for an instance, the *positionally-weighted Kendall's* τ measures the correlation between these rankings. Notably, this correlation coefficient weights correctness of higher ranks more than that of lower ranks. It is denoted by K_δ^{sim} and defined as

$$K_\delta^{\text{sim}} = 1 - \frac{K_\delta(x, y)}{K_\delta^{\max}(x)} \tag{7.1}$$

Here, $K_\delta(x, y)$ is the number of item swaps required to make two sequences x and y identical, where each rank is weighted by variable penalty function δ. Therefore, it resembles a *distance* function. $K_\delta^{\max}(x)$

is the maximum possible distance between x and any other sequence, therefore normalizing the subtrahend of the equation to be between 0 and 1. To convert this distance measure to a similarity measure, it is subtracted from 1.

The *weighted variant of Discounted Cumulative Gain* (WDCG) is another way to compare an applicability rating of senses against a baseline. Given a gold standard applicability rating w_i of k senses and another rating $\hat{w}_i$ that is to be compared, it is defined as

$$WDCG = \sum_{i=1}^{k} \frac{\frac{min(w_i,\hat{w}_i)}{max(w_i,\hat{w}_i)}\left(2^{w_i+1}-1\right)}{log_2(i)} \tag{7.2}$$

where i is the index of the sense in descending order of applicability according to $\hat{w}_i$. It is based on the Discounted Cumulative Gain (DCG) [Järvelin and Kekäläinen, 2002], which is defined as $DCG = \sum_{i=1}^{k} \frac{2^{w_i+1}-1}{log_2(i+1)}$. The *Ideal DCG* (IDCG) is its maximum value and achieved by an identical applicability *ranking*, irrespective of the specific rating $\hat{w}_i$. Every deviation in this ranking is penalized by a lower score, where higher ranks are weighted less due to the denominator $log_2(i+1)$. The crucial difference of the WDCG to the DCG is that additionally to penalizing a deviating rank of a specific sense, an additional factor $\frac{min(w_i,\hat{w}_i)}{max(w_i,\hat{w}_i)}$ is added that also penalize a discrepancy between the respective applicability *ratings* w_i and $\hat{w}_i$. To normalize the WDCG, it is divided by the Ideal WDCG, which is defined analogously to the IDCG. The result is therefore called the *weighted variant of Normalized Discounted Cumulative Gain (WNDCG)*.

Comparing Clusterings

Alternatively, sense labels may be considered to be a *clustering* of instances into groups, where each group is formed by instances with the same sense label (also referred to as *class*, for an illustration see Figure 7.1). Since both the sense tags of the WSI system, as well as the gold sense tags can be considered to be such clusterings, the performance of the WSI system can be measured by comparison of these two clusterings. Conclusively, we need means to compare two clusterings. The following is a list of cluster-comparison measures that have been used in previous WSI evaluations. For each sense label forming one cluster, we will use the term *class*, to conform to definitions in the literature.

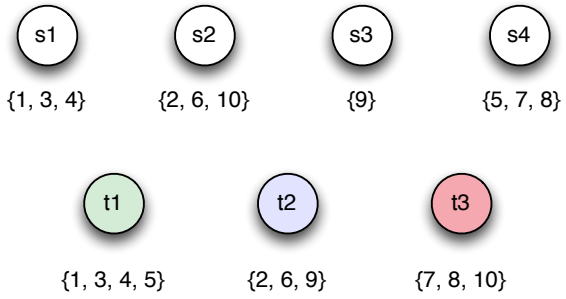

Figure 7.1: Two example clusterings $T = \{t_1, t_2, t_3, t_4\}$ and $S = \{s_1, s_2, s_3\}$ of the same 10 instances. Circles indicate clusters along with their cluster label (s1, s2, ..), numbers indicate instances. Clusters in T are colored to visually distinguish them from S.

Purity [Zhao and Karypis, 2001] and *Inverse Purity*: As the name already says, this measure quantifies the *purity* of a clustering, with respect to a reference clustering, of which the latter specifies the element's classes: For each cluster, it is equal to the number of elements with the pre-dominant class in this cluster,

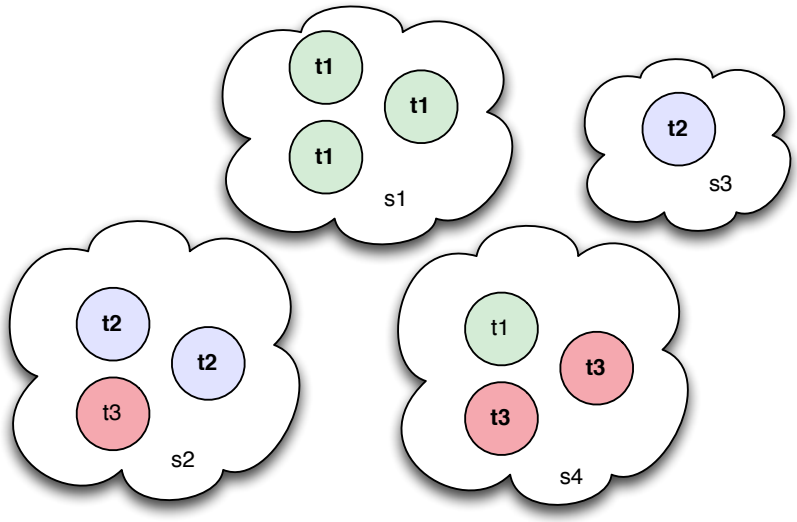

Figure 7.2: Same clusterings as in Figure 7.1, visualized from the perspective of a mapping of classes from S to classes from T: For example, in the cluster formed by class s_2 (bottom-left), two instances are also labeled with t_2, making this the dominating class from clustering T in class s_2.

divided by the cluster size. For example, purity of the cluster s_2 with respect to T in Figure 7.2 is 2/3. The purity of a clustering is given by the average purity of all its clusters. *Inverse purity* then refers to the purity of the other clustering, with the first clustering specifying the element's classes. See Figure 7.6 for an illustrative example along with an explanation. An overall result is achieved by combining both measures with the harmonic mean (i.e., F-measure) [Van Rijsbergen, 1979], as done in e.g. [Amigó et al., 2009]. The advantage of this combination is that a baseline assigning all elements to one cluster is hard to beat (see experiments below). Also, it is a relatively intuitive measure without any lengthy or complex equations, making it easier to interpret. However, a drawback is that it is insensitive to cluster changes that are not in the dominating class of a sense cluster. Figure 7.3 illustrates this problem: Clusters s_1 and s_2 contain elements with the same class and should therefore be merged to improve the clustering S with respect to T. However, both purity as well as inverse purity are insensitive to this change: the dominating class of t_1 (which is s_3) would not change, and neither would the dominating class of s_1 or s_2.

$$P = \frac{|F(S) \cap F(T)|}{|F(S)|} \tag{7.3}$$

$$R = \frac{|F(S) \cap F(T)|}{|F(T)|} \tag{7.4}$$

Adaptations of *Recall*, *Precision* and *F-measure* were also suggested for cluster comparison [Zhao and Karypis, 2002]. To compare a clusterings S with a gold standard clustering T, these can be applied by first generating all possible pairs of elements in each cluster s_i and t_i. Let $F(S)$ be the set of such instance pairs generated from S, and $F(T)$ the set of instance pairs generated from T. Precision is then defined as the number of shared pairs between $F(S)$ and $F(T)$, divided by the total number of pairs in $F(S)$ (eq. 7.3). Recall is analogously defined as the number of shared pairs in $F(S)$ and $F(T)$, divided by the total number of pairs of the gold clustering in $F(T)$ (eq. 7.4). The F-measure is as usual computed as the harmonic mean of both recall and precision. This adaptation of the three measures has especially been used in SemEval-2010 task 14.

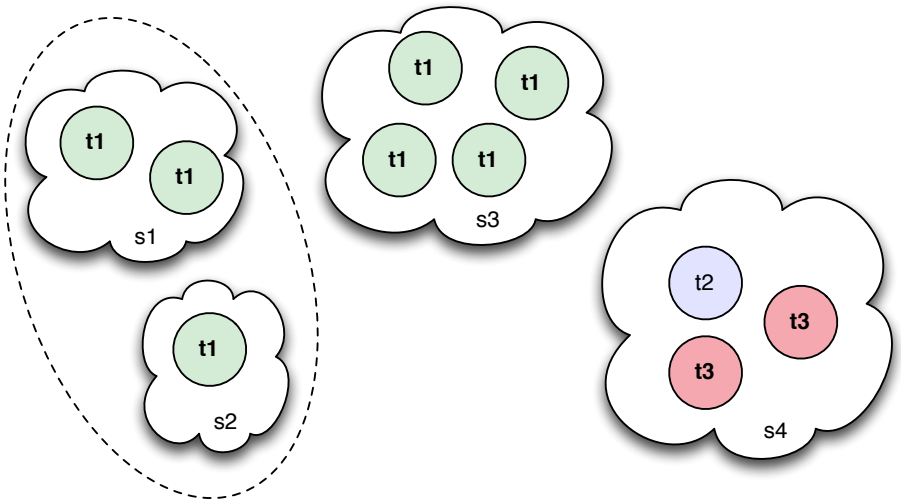

Figure 7.3: Illustration of a problem with purity and inverse purity: Clusters s_1 and s_2 contain elements with the same class and should therefore be merged to improve the clustering S with respect to T. However, both purity as well as inverse purity are insensitive to this change: the dominating class of t_1 (which is s_3) would not change, and neither would the dominating class of s_1 or s_2.

$$\text{B-Cubed Precision} = \operatorname*{avg}_i \left[\operatorname*{avg}_{j \neq i \in \cup \mu_t(i)} P(i,j) \right] \tag{7.5}$$

$$\text{B-Cubed Recall} = \operatorname*{avg}_i \left[\operatorname*{avg}_{j \neq i \in \cup \mu_s(i)} R(i,j) \right] \tag{7.6}$$

$$C(i,j,S) = \sum_{k \in \mu_s(i) \cup \mu_s(j)} 1 - |w_k(i) - w_k(j)| \tag{7.7}$$

$$P(i,j,S) = \frac{\min\left[C(i,j,S),\ C(i,j,T)\right]}{C(i,j,S)} \tag{7.8}$$

$$R(i,j,S) = \frac{\min\left[C(i,j,S),\ C(i,j,T)\right]}{C(i,j,T)} \tag{7.9}$$

Fuzzy B-Cubed [Jurgens and Klapaftis, 2013] is a measure that quantifies a similarity between two clusterings with *fuzzy covers*, i.e. clusterings with multiple, graded assignments of items to clusters. This cluster comparison measure may especially be used in a sense clustering setting where multiple applying senses may partially be assigned to instances. This was e.g. required in the SemEval-2013 WSI task, where this measure was also first introduced. As suggested by its name, it is a generalization of the *B-Cubed* measure [Baldwin et al., 1998] to fuzzy covers. B-Cubed precision and recall are defined in equations 7.5 and 7.6, resp. Here, $\mu_s(i)$ denotes the set of cluster in S that i is member of. B-Cubed precision is therefore the average item-based precision of item i respective to all other items j that share at least one cluster with i. B-Cubed recall is defined analogously. Item-based recall and precision are defined as follows. The extent of a cluster overlap between two items i and j with respect to a clustering S is defined as $C(i,j,S)$ in equation 7.7. Here, $\mu_s(i)$ again denotes the set of clusters in which i is a member, and $w_k(i)$ denotes the membership weight of item i in cluster k. $C(i,j,S)$ is therefore 1 when

i and j are present in identical clusters and their membership weights are identical. Reversely, it is 0 if i and j have no mutual clusters. Item-based precision and recall can then be defined as in equations 7.8 and 7.9, resp. They can be best explained using an example: Assume you want to compare two cluster assignments of word instances i and j with multiple cluster assignments allowed per instance, and in sense clustering S they are placed in the same clusters, i.e. $C(i,j,S) = 1$. Further suppose that in the reference sense clustering T their cluster assignments overlap only to 50%, i.e. i and j share only 50% of their multiple sense labels, and therefore $C(i,j,T) = 0.5$. In other words, the word senses used in i and j are semantically different according to T, yet have a certain semantic overlap. The resulting item-based precision is then $P(i,j,S) = 0.5$, as the extent of their cluster overlap in S is twice their overlap in T. In other words, S "over-estimates" the semantic overlap of the senses of instances i and j. It however correctly detects their senses' semantic overlap to at least 50% (although it over-estimates it) , therefore the respective recall $R(i,j,S)$ is 1. Therefore, whenever S over-estimates the semantic overlap of the senses of two instances, recall is 1 and precision below 1; reversely an under-estimation yields a recall below 1 and a precision of 1.

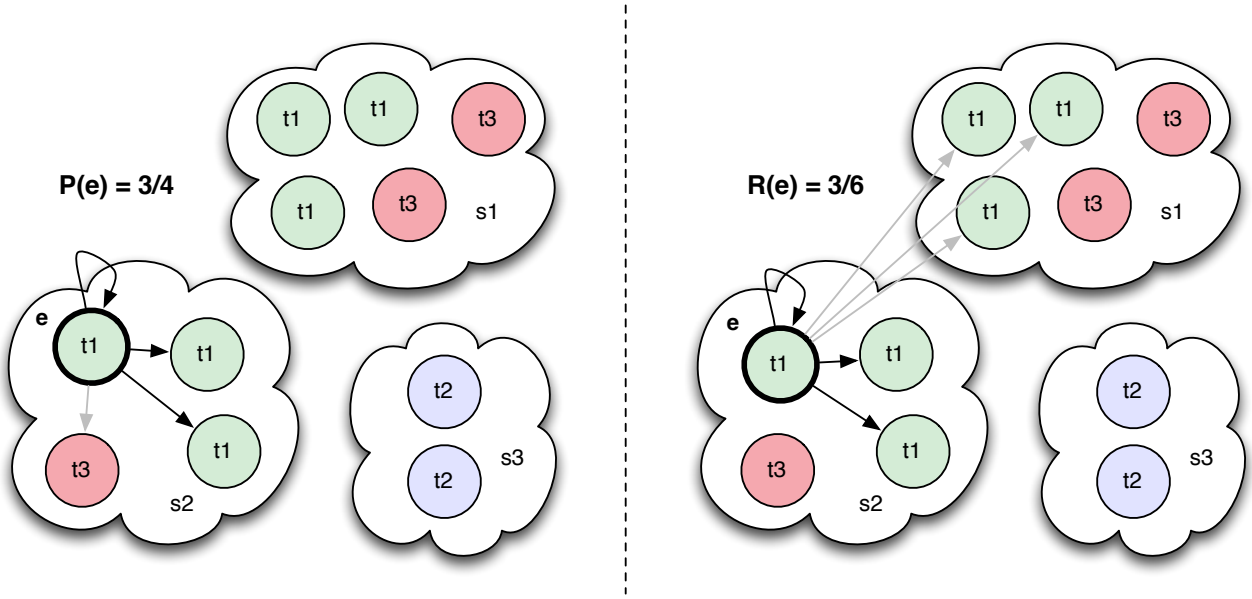

Figure 7.4: Illustration of B-Cubed measure for item e. If one were to look for items with the same class as e in cluster s_2, then this "retrieval" operation would come with a precision of $3/4$ and a recall of $3/6$.

A clustering of a set of instances can also be represented by a random variable X_k for each cluster k that is defined over all instances and has 0 (not part of the cluster) and 1 (part of the cluster) as possible outcomes. *Mutual Information* can be used to measure the dependency between two such random variables X and Y, therefore resembling a similarity measure between two clusters. In a graded sense assignment setting (such as found in the SemEval-2013 WSI task), these random variables can be extended to take on any value in $[0,1]$, representing a *probability* of each instance being assigned to a certain sense cluster. Based on this assumption, *Fuzzy Normalized Mutual Information* [Jurgens and Klapaftis, 2013] measures the dependency of two fuzzy clusters X and Y. Equation 7.10 defines the mutual information $I(X;Y)$ of two such random variables X and Y. Here, $H(X)$ denotes the entropy of X and $H(X|Y)$ the entropy of X conditioned on Y. The *Normalized Mutual Information* may be determined by dividing the mutual information by an upper bound, with $max\left[H(X),H(Y)\right]$ being recommended in [Vinh et al., 2010].

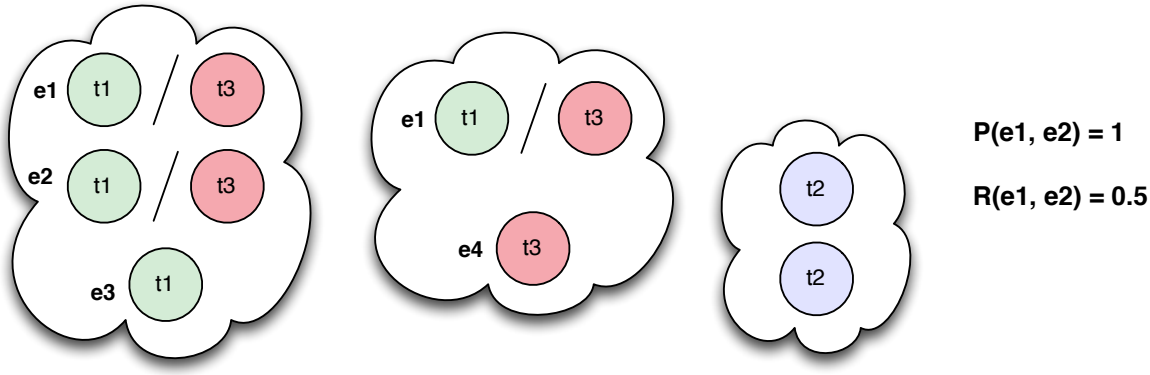

Figure 7.5: Illustration of extended B-Cubed precision and recall for the item pair $\{e_1, e_2\}$. The extended version allows multiple cluster and class assignments for each element. Here, both e_1 and e_2 are assigned to classes t_1 and t_3. Moreover, e_2 is assigned to only one cluster, while e_1 is assigned to two clusters simultaneously. Precision for $\{e_1, e_2\}$ is 1, as for every shared cluster, there is a shared class. Recall is 0.5, as for only half of the shared classes, there is a shared cluster.

$$I(X;Y) = H(X) - H(X|Y) \tag{7.10}$$

$$NMI(X;Y) = \frac{I(X;Y)}{max\left[H(X), H(Y)\right]} \tag{7.11}$$

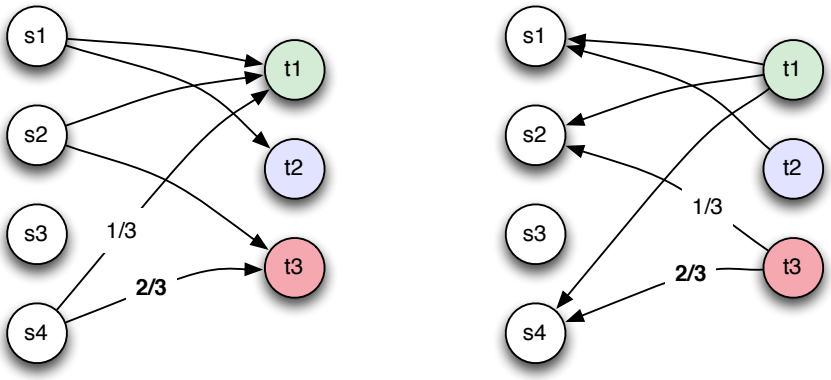

Figure 7.6: Possible mapping of the two example clusterings above. Since instances labeled with s4 appear in cluster t3 in 2/3 of the cases, and in cluster t1 in 1/3 of the cases, s4 is mapped to t3. The *purity* of this mapping is therefore 2/3. The same can be done for the other direction (mapping clusters from T to clusters from S), here t3 would receive an *inverse purity* of 2/3. Purity values are then averaged over the clusters in S, while inverse purity values are averaged over T. Computing the harmonic mean (F1 measure) of both allows to quantify the similarity of both clusters: The higher the F1 score, the more similar both clusterings are.

7.2 WSI Evaluation Datasets

There have been various different evaluation tasks for WSI in the past, however we here list three evaluations that we find most noteworthy.

SemEval-2007 task 2 [Agirre and Soroa, 2007]: This task re-used data from task 17, the English lexical sample subtask [Pradhan et al., 2007a]. Disambiguation instances were taken from the Wall Street Journal corpus, and were manually sense-tagged with OntoNotes [Pradhan et al., 2007b] senses. 6 teams participated, and were asked to cluster word instances into groups, each group representing a distinct word sense. The manually tagged OntoNotes senses were notably coarser than senses found in WordNet. The task contained both an unsupervised cluster-comparison setting, as well as a supervised evaluation setting, in which induced senses were mapped to OntoNotes senses.

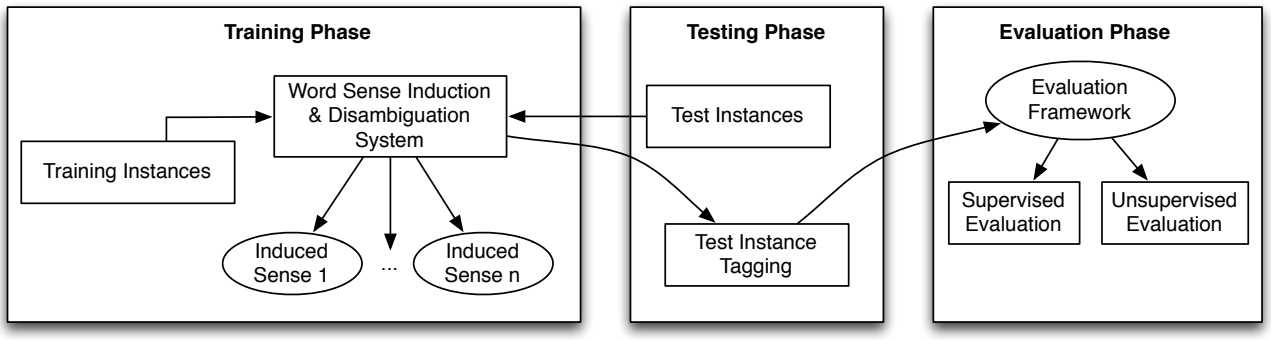

Figure 7.7: Setup of SemEval-2007 task 14. The graphic has been adopted from [Manandhar, 2010].

SemEval-2010 task 14: Word Sense Induction & Disambiguation [Manandhar, 2010]. The main difference of this task compared to the SemEval-2007 WSI task (see above) is that Manandhar et. al. split the dataset (i.e. word instances) into two parts, namely into a *training* set and a *test* set. Figure 7.7 shows the general setup of this task and the conceptual distinction between training and test phase. The training set had to be used by the participating systems solely to induce word senses. Notably, no other resources were allowed (neither structured nor unstructured), hence also no background corpus could be used to extract additional statistics like n-gram frequencies. The test set was in turn only used for sense disambiguation, and participants had to draw the set of word senses from the induction process in the training phase. The term *disambiguation* is here therefore chosen deliberately in order to contrast this second step to mere sense *discrimination*, i.e. grouping of instances with the same word sense. The difficulty in this task therefore lies in the assignment of before *unseen* instances to previously induced senses. The problem with many WSID systems that this tasks aims to uncover, i.e. that many systems are unable to sense-label unseen instances, is exactly one of the problems the system described in this thesis aims to address. Disambiguation instances used in this task were constructed using a semi-automatic method. Target words were issued as query to the *Yahoo!* search engine and expanded by further terms found in each of WordNet's senses of this word[1]. The set of target words is comparatively large, containing 100 target words in total; namely 50 nouns and 50 verbs. For each of these target words, and for each of WordNet's senses of these, at most 1000 documents were downloaded. For each WordNet sense, only text fragments were retained that contained the target word and matched the target word's POS tag. The remaining text fragments were the sense-tagged instances[2].

[1] Related words that were considered were hypernyms, hyponyms, synonyms, meronyms and holonyms of a specific synset.

[2] Note that the sense tags were not visible to participants, and only used in the test phase to evaluate performance of the systems

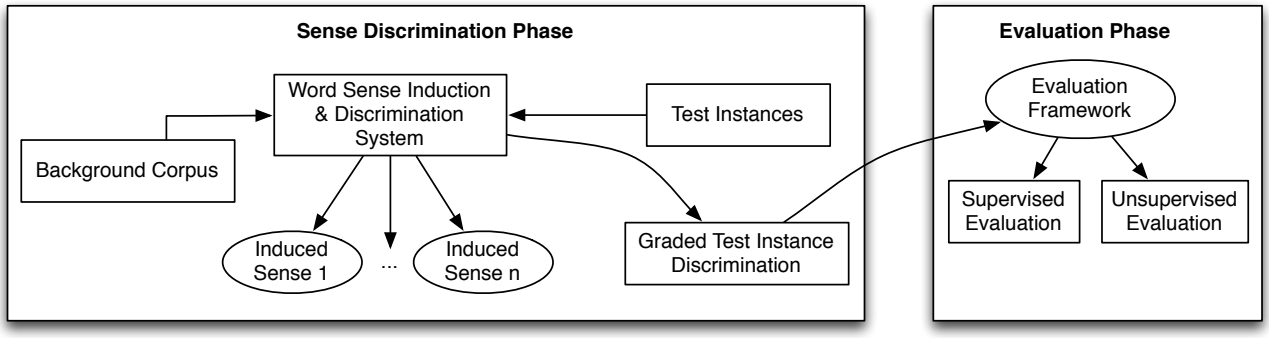

Figure 7.8: Setup of SemEval-2013 task 13. Note how the first two phases of the SemEval-2007 WSI task are here collapsed into one.

SemEval-2013 task 13: Word Sense Induction for Graded and Non-Graded Senses [Jurgens and Klapaftis, 2013]. This task provides 20 nouns, 20 verbs and 10 adjectives in WordNet-sense-tagged contexts. It contains 20-100 contexts per word, and 4664 contexts in total, which were drawn from the Open American National Corpus[3] [Ide and Suderman, 2004]. Participants were asked to cluster the 4664 instances into groups, with each group corresponding to a distinct word sense. No specific training instances were provided in this task. However, to support systems in the sense induction, participants were provided with the ukWaC[4] corpus. This corpus consisted of roughly 2 billion words from crawled web pages and was POS-tagged and lemmatized using the TreeTagger[5]. In contrast to the SemEval-2007 WSI task, systems were not required to perform sense disambiguation separately from sense induction. Instead, this task focused on the ability of participating systems to discover the overlapping use of multiple word senses in a specific instance. For example, the adjective *dark* may mean both *deficient in light* as well as *secret*. Usages of the latter sense can therefore also be a mixture of both senses. For example, the instance *Bad guys always hide in* ***dark*** *places* can in fact refer to a *secret* place, though at the same time subsuming the absence of light. Goal in this task was therefore to detect instances in which multiple senses applied, and if so score each sense according to its applicability (using a score between 0 and 1). To compare performance of participating systems in both graded as well as non-graded sense discrimination settings, instances with multiple applying senses were tested separately.

7.3 Semi-Automatic Evaluation Using Wikipedia's Link Structure

Despite the availability of evaluation datasets for WSI, we decided to construct our own for three major reasons: First, existing evaluations mostly concentrate on a relatively small number of words (e.g. 20 nouns in the SemEval-2013 challenge). Optimizing the various parameters of our system specifically for these few words would likely yield an overfitting to this dataset, making the system less suited for application on previously unseen words. This, however, is one of the fundamental goals of WSI systems. Secondly, existing evaluations often included multi-word expressions and words from all word classes (nouns, verbs, adjectives and adverbs). Since we chose to concentrate on nouns with a single token to simplify the thesaurus computation, the number of remaining evaluation instances from existing datasets was too low to be useful for optimizing parameters. Thirdly, the discrepancy between the domain of the text corpus used by the system to induce senses, and the corpus used by the evaluation imply a certain error that is hard to quantify. To avoid this, we extracted a disambiguation dataset from Wikipedia, i.e. the same corpus that we used to induce our sense inventory from.

[3] `http://www.americannationalcorpus.org/OANC/index.html` (last accessed: 2015/04/30)
[4] `http://wacky.sslmit.unibo.it/` (last accessed: 2015/04/30)
[5] `http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/` (last accessed: 2015/04/30)

We did so by utilizing Wikipedia's link structure: Ambiguous terms are often linked to their corresponding page, e.g. *vision* is often linked to the page *Visual_perception*, but also to *Vision_(spirituality)*, which both represent two distinct word senses. Link frequencies give a hint at the frequency distribution of the various word senses. In the following, we will refer to the word or text being linked as *link text* and to the linked page as *link target*.

Link text	Freq.	Wikipedia senses (filtered link targets)
soul	9162	Soul_music:4892 Soul:3981
canon	3899	Canon_(priest):2047 Canon_(fiction):588
feud	2695	Feud_(professional_wrestling):2220 Feud:378
organ	6669	Organ_(music):3751 Organ_(anatomy):1311 Pipe_organ:1089
chain	2631	Chain_store:806 Chain:540
resistance	4259	Electrical_resistance_and_conductance:1483 Resistance_movement:426
rotation	2714	Rotation:1438 Rotation_(mathematics):899
type	3561	Data_type:1148 Type_(biology):686
chicken	4054	Chicken:2672 Chicken_(food):1137
head	3425	Head:812 Head_(linguistics):352

Table 7.1: Table showing 10 randomly selected words of the 100 polysemous nouns chosen for this evaluation. The right-most column shows the extracted Wikipedia senses (with uncommon senses removed), the numbers denote their frequency.

To build the evaluation dataset, we first extracted sentences along with links from a Wikipedia dump[6]. Sentences that are too long or too short are dropped. From this link-annotated text corpus, we picked 100 polysemous nouns with at least two different senses (link targets). To filter out link targets that do not represent actual word senses, we used a simple heuristic: a link target is only accepted as distinct word sense if it makes up at least 10% of all instances of its particular link text. This reduced the number of word senses from several dozen per word to just a few. See Figure 7.1 for a list of extracted word senses for 10 exemplary polysemous nouns. Mostly, the filtering step removed link targets that represent conceptual instances (e.g. *Brooklyn_Bridge*) rather than concepts (e.g. *Bridge_(music)*) or sub-concepts (e.g. *Beam_bridge*), due to their lower frequencies. Finally, for every of the 100 words, we compiled a list of 100 sentences containing one of these filtered links, i.e. word senses. We chose this number so that frequent words have the same number of evaluation instances as infrequent words, though there are far more than 100 contexts for frequent words. The resulting dataset therefore contains 10,000 sense-tagged disambiguation instances. With manual filtering of undesirable links, the process to extract this dataset from a specific Wikipedia dump took a few hours for a single person[7].

Evaluation Process

The evaluation process is straightforward: The filtered targets of wiki-links served as gold senses, and were considered to be a gold clustering of the disambiguation instances. The clustering implied by the contextualized induced senses formed another clustering. For an exemplary contextualization of the polysemous words *vinyl* and *bond*, see Table 7.2. We then performed a cluster comparison as described in Section 7.1: both clusterings were compared using purity, inverse purity and their harmonic mean (F-measure).

[6] These are the same sentences we used to induce word senses, only with additional link annotations.

[7] Some effort also went code for processing Wikipedia dumps and extracting relevant link structure, however this code may be re-used without further effort to extract datasets from other versions of Wikipedia, including more recent dumps and dumps from other languages.

word	gold sense	s	lemmatized context words (original order)
vinyl	Gramophone_record	1	Barry Gray 's score receive **vinyl** release ...
vinyl	Gramophone_record	1	the album be release on both compact disc and **vinyl** ...
vinyl	Gramophone_record	1	this be the first of two Pink Floyd album ...
vinyl	Gramophone_record	1	originally 5,000 **vinyl** triple LP (3xlp) copy be press .
vinyl	Gramophone_record	1	Shrimpton have release various singles on **vinyl** and cd ...
vinyl	Gramophone_record	1	Cosmos be available for pre-order ...
vinyl	Vinyl	0	Soap scum on **vinyl** shower curtain have be report ...
vinyl	Vinyl	1	Urban **vinyl** figure have become collectible item .
vinyl	Vinyl	0	Lau be widely credit as the founder of the urban **vinyl** style ...
bond	Chemical_bond	1	for example oxygen can form very strong **bond** ...
bond	Chemical_bond	1	this process form strong au-sr **bond** and release h2 .
bond	Chemical_bond	1	one major use of NMR be to determine the **bond** connectivity within a organic molecule .
bond	Chemical_bond	1	the " stretch frequency " of **bond** between fluorine ...
bond	Bond_(finance)	0	the major designer and promoter of the Formosa **bond** ...
bond	Bond_(finance)	0	if , today , $ 1,000 be "put into" some **bond** or stock ...
bond	Bond_(finance)	0	this perception of instability in United States monetary ...
bond	Bond_(finance)	0	in September 2011 , Caxton FX issue four year non-transferable company **bond** ...
bond	Bond_(finance)	0	inflation-indexed **bond** pay a periodic coupon ...
bond	Bond_(finance)	0	the purchase price be equal to the **bond** 's face value ...

Table 7.2: Example contextualization subsets with gold labels where WSD performed well. Notably, all 100 instances of *bond* where correctly distinguished into the two gold senses.

We used several baselines to compare our system with. The first one assumes one sense per word, which makes contextualization trivial. The others assign $s \geq 2$ senses to each instance randomly. With $s >> 100$ (the number of instances per word), this baseline is effectively equivalent to one that assigns one sense per instance. We therefore have several baselines that indicate performance between these two extremes of trivial disambiguation methods (one sense per word and one sense per instance).

Results

First of all, we were able to directly observe the effects of the different parameters on the induced senses: Higher values of n resulted in a more coarse-grained sense clustering, while higher values of N and γ resulted in a more fine-grained sense clustering.

The results of our evaluation are listed in Table 7.3 (for the baselines) and Table 7.4. The first observation from these results is that the baseline assigning one sense per word is quite competitive: only few configurations of our system were able to surpass the score of this baseline.

However, there were two specific configurations that outperformed this baseline. The first is a coarse-grained clustering ($\gamma = 1.4$, $n = 100$), with an improvement of 0.83% F-measure over the baseline. The second is a rather fine-grained clustering with an additional cluster merging step that collapses this fine-grained clustering to a coarse-grained clustering, with an improvement of 2.1% F-measure over the baseline. We explain these results as follows: As the baseline with $s = 1$ performs well with regard to

# senses	Purity	Inv. Purity	F-1
1	0.6587	1.0	**0.7942**
2	0.6621	0.5544	0.6035
3	0.6622	0.4048	0.5025
10	0.6889	0.1772	0.2819
100	0.8549	0.0627	0.1168

Table 7.3: Evaluation results (Baseline)

the overall F-1 value, the best strategy for our WSI algorithm is to stick to a single sense per word unless there is enough evidence to add a second or third sense. Therefore, the configurations with the lowest average sense number perform highest. A notable exception are fine-grained clusters that are merged in a second optimization step, which we briefly elaborate on in the following.

DT (p, t_{wf})	MCL (N, n, γ, g)	Cluster Opt. (p^*, s)	Avg. # senses	Purity	Inv. Purity	F-1
(1000, 2)	(100, 100, 1.4, 0.0)	–	1.28	0.6884	0.9620	0.8025
(1000, 2)	(100, 10, 1.4, 0.0)	–	1.58	0.7082	0.9069	0.7953
(1000, 2)	(100, 5, 1.4, 0.0)	–	2.36	0.7468	0.7980	0.7716
(1000, 2)	(100, 5, 1.4, 0.0)	(10000, 0.6)	1.09	0.6704	0.9947	0.8010
(1000, 2)	(100, 5, 1.4, 0.0)	(10000, 0.7)	1.48	0.7163	0.9457	**0.8152**
(1000, 2)	(100, 100, 1.7, 0.0)	–	1.55	0.7054	0.9249	0.8004
(1000, 2)	(100, 50, 1.7, 0.0)	–	1.57	0.7064	0.9190	0.7988
(1000, 2)	(100, 5, 1.7, 0.0)	–	5.23	**0.7931**	0.6080	0.6883
(1000, 2)	(100, 5, 1.7, 0.0)	(10000, 0.6)		0.6790	0.9843	0.8036
(1000, 2)	(100, 5, 1.7, 0.0)	(10000, 0.7)		0.7321	0.8857	0.8016
(1000, 2)	(200, 5, 1.4, 0.0)	–	3.99	0.6919	0.7844	0.7353
(1000, 2)	(200, 5, 1.4, 0.0)	(10000, 0.7)		0.7332	0.9029	0.8093
(1000, 2)	(200, 5, 1.4, 0.0)	(10000, 0.6)		0.6788	0.9903	0.8055

Table 7.4: Evaluation results (Markov Chain Clustering). The average number of gold senses within this set are 2.32.

Cluster Merging

As outlined previously, we tested whether merging of clusters with only subtle distinctness is beneficial for WSD performance (see Section 5.5). For this, we merged clusters that shared a fraction greater than s of their p most common co-occurrence features. Figures 7.9 and 7.10 show an exemplary cluster merging of the words *bay* and *bond*, resp., and the results of this merging on the performance in our evaluation. As the test results show, best results were achieved *with* merging of clusters, specifically with $s = 0.7$. With this additional step, we were able to *surpass the baseline by 2.1% F-measure*, in contrast to 0.83% before. Notably, this setting produced a *higher* number of senses than the best-performing setting without cluster merging. As this setting performed better nonetheless, this is a strong indicator that our additional step of merging clusters indeed produces qualitatively better clusters.

(a) Fine-grained cluster of *bay* with two senses that are hard to distinguish for our WSD.

(b) Results in same evaluation after merging similar clusters with $s \geq 0.7$.

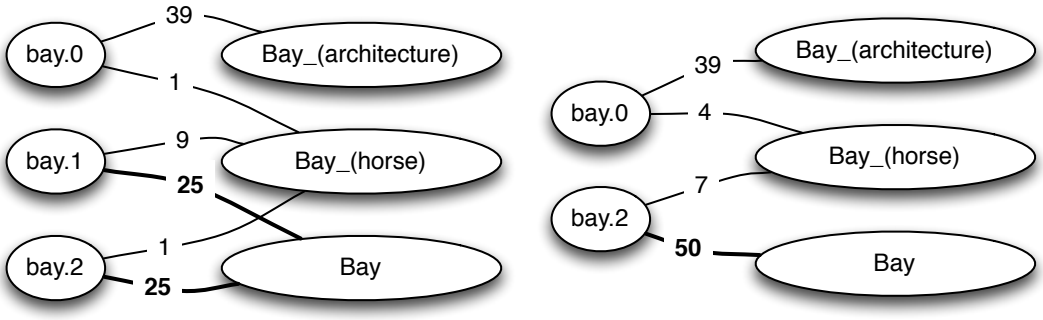

Figure 7.9: Contextualization results of 100 gold-sense-annotated test instances for word *bay*.

(a) Fine-grained cluster of *bond* with two senses that are hard to distinguish for our WSD. The clustering induced 7 senses, of which only the first 4 are shown here.

(b) Results in same evaluation after merging similar clusters with $s \geq 0.7$.

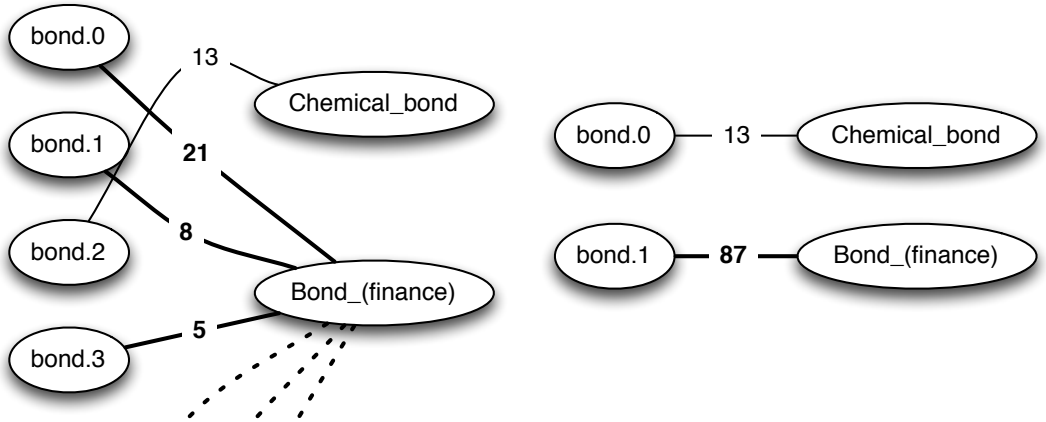

Figure 7.10: Contextualization results of 100 gold-sense-annotated test instances for word *bond*.

Limitations

Even though the automatically extracted evaluation dataset described in this section has the major advantage of offering a high number of contexts for a large number of words at little manual effort, it also has its downsides.

First of all, gold senses extracted in this dataset tend to have distorted *prior probability distributions*: Situations, in which the sense of a term is obvious are not specifically linked. For example, the word *capital* may not necessarily be linked to `Capital_city` in a context such as *The capital city of Germany is Berlin*. As another example, the term *host* is linked to `Host_(biology)` 5 times more often than to `Presenter`, which intuitively does not represent the actual probability distribution within common language. Therefore, this might distort the evaluation results with respect to how well a system is capable of falling back to a meaningful MFS when only little contextual information is available.

Furthermore, Wikipedia comes with inherent inconsistencies due to its collaborative nature: For the term *chicken*, there are two main Wikipedia articles: `Chicken_(food)` and `Chicken`. While the second refers to the animal species, some mentions in food-specific contexts are also linked to the latter page. This,

however, is not a new problem: inter-annotator agreement has been a long-standing issue with manually labeled data [Pradhan and Xue, 2009].

Summary

In this evaluation task, gold senses were relatively coarse-grained with an average number of 2.32 senses per word. Here, the best strategy turned out to be to produce a lower number of high-quality clusters, as opposed to producing an average number of clusters similar to the gold standard. Namely, a system configuration that yielded an average of 2.36 clusters ($n = 5, \gamma = 1.4$) received a lower F-1 measure than a system with only an average of 1.28 induced senses per word ($n = 100, \gamma = 1.4$). A possible explanation is that all F-1 scores are relatively close to the baseline's F-1 score with 79.42%. Many configurations even yielded a lower score. Therefore, systems performed best if they used more than 1 sense cluster only if there is strong evidence to do so. Without additional cluster merging, the best-performing system was therefore the most coarse-grained clustering configuration with $n = 100$ and $\gamma = 1.4$.

Secondly, it turned out to make little difference whether a more fine-grained clustering is achieved by increasing the inflation parameter γ or by decreasing the graph connectivity n. The performance (when no cluster merging is performed) was mostly dependent on the average number of induced senses.

However, this changed when "fuzzy" clusters were merged: we joined two clusters if their p^* most frequent co-occurrence features overlapped to a fraction of more than s. This was motivated by the observation that two clusters were hard to distinguish for our disambiguation algorithm if their context clues were similar. The evaluation results show that this improved results significantly. Best results were notably achieved by choosing a fine-grained clustering with $n = 5$ and $\gamma = 1.7$ and merging clusters that shared more than a fraction of $s = 0.7$ of their $p^* = 10,000$ most frequent context clues.

Lastly, for sense disambiguation, we found a smoothing of $\alpha = 1 \times 10^{-5}$ for context clue scores[8] to produce best results in this evaluation, independent of other parameter settings.

7.4 SemEval-2013 Task 13

To assess the performance of our WSID system in comparison to other existing systems, we ran evaluations against the SemEval-2013 WSI subtask dataset. For this, we took the three best-performing configurations according to our own evaluation, without performing any further modifications specific to this task. This way, we aimed at receiving a fair comparison to previous participants, as these submitted their system at a point when the test data was unknown.

Participating Systems

Participating teams in this task were *AI-KU*, *Unimelb*, *UoS* and *La Sapienza*[9]. Details of their respective systems are described in section 4.3. Notably, only UoS uses a pre-computed sense inventory, while all other participating teams performed sense clustering directly on the disambiguation instances.

[8] This was necessary as during disambiguation, a context clue score of 0 for a single feature from a word context would result in an overall score of 0 for this specific sense.

[9] The *La Sapienza* system is in fact a WSD system, and relies on WordNet as sense inventory instead of inducing senses.

[10] Baseline without mapping, senses directly from WordNet

Participant/System	F-1	NMI	B-Cubed
$F = \{coocs\}, p^* = 1000, s = 0.7$	61.24%	0.031	53.80%
$F = \{coocs, deps\}, p^* = 1000, s = 0.7$	61.30%	0.024	**54.43%**
$F = \{coocs, deps\}$, no cluster-merging	61.12%	**0.064**	49.24%
AI-KU/Base	**65.34%**	0.047	34.53%
AI-KU/Add1000	60.09%	0.023	28.75%
AI-KU/Remove5-Add1000	62.90%	0.026	42.10%
Unimelb/50k	60.46%	0.039	44.08%
Unimelb/5p	59.61%	0.035	42.15%
Sapienza/System-1	54.97%	0.033	13.09%
Sapienza/System-2	55.05%	0.030	12.53%
UoS/Top-3	62.39%	0.030	42.31%
UoS/WN	60.52%	0.032	17.01%
All Instances, One Sense	**60.28%**	0	**58.25%**
One Instance, One Sense (1c1instance)	0	**0.055**	0%
Semcor MFS[10]	47.72%	0	57.00%

Table 7.5: Evaluation results in SemEval-2013 Task 13 for all 20 nouns in a single-sense setting. The first row lists results of our system. Parameters of our system were $N = 100, t_{wf} = 2$. Best system performances and best baselines are marked bold.

Results

Tables 7.5 and 7.6 list the results of our system compared to other participants. Surprisingly, our system received the highest scores for both evaluation measures (Fuzzy NMI and B-Cubed) in the cluster comparison setting. Note, however, that we used Wikipedia as background corpus and not ukWaC like other participants. Yet, we did not perform any optimization or adaptation of our system to this evaluation. Furthermore, it was the only system to beat the Fuzzy NMI score of the `1c1instance` baseline. However, in the supervised setting of this evaluation[11], performance was only 1.02% over the F-1 measure of the all-instances-one-sense baseline, while the best-performing system with respect to this measure achieved a 5.06% improvement over this baseline.

We see three possible reasons for these results: Most evidently, we used a different background corpus to induce word senses. Due to the high quality of text found in Wikipedia compared to the ukWaC corpus (which is extracted from web crawls), this might have a crucial impact on the quality of discovered word senses. Secondly, we optimized our system to a cluster-comparison setting very similar to the one used in this evaluation. This might explain the discrepancy of the performance of our system in the supervised and the unsupervised evaluation setting. Thirdly, the setting that optimized performance in our own evaluation produced a relatively low number of senses per word (1.48 on average). While this was optimal for senses we extracted from Wikipedia links (2.32 per word on average), this is less suited in a setting where induced senses are mapped to much more fine-grained WordNet senses (as done in this WSI task): The average number of induced senses for other participating systems was between 5 and 9 senses per word, while WordNet contains an average of 8.66 senses for the test words.

7.5 Summary

We in this chapter evaluated the performance of our WSID system using two datasets, namely a Wikipedia-based WSD dataset, and the SemEval-2013 WSI task. In the first evaluation, our system

[11] For details on this setting, see Section 7.1.

Participant/System	Jacc. Index	WNDCG	Fuzzy NMI	Fuzzy B-Cubed	#S
$F = \{coocs\}, p^* = 1000, s = 0.7$	17.15%	31.30%	0.037	58.55%	1.3
$F = \{coocs, deps\}, p^* = 1000, s = 0.7$	17.19%	31.29%	0.033	**59.16%**	1.3
$F = \{coocs, deps\}$, no cluster-merging	17.22%	32.67%	**0.076**	52.93%	2.35
AI-KU/Base	17.64%	**39.35%**	0.066	38.18%	21.8
AI-KU/Add1000	17.64%	20.47%	0.033	31.65%	21.8
AI-KU/Remove5-Add1000	**22.81%**	33.07%	0.040	46.33%	20.45
Unimelb/50k	19.83%	38.39%	0.060	49.44%	10.35
Unimelb/5p	19.77%	37.36%	0.056	47.54%	
Sapienza/System-1	16.39%	28.54%	0.049	14.84%	9.05
Sapienza/System-2	16.39%	20.41%	0.046	13.00%	9.05
UoS/Top-3	21.98%	36.98%	0.044	45.14%	21.9
UoS/WN	17.15%	29.83%	0.046	18.59%	9.05
All Instances, One Sense	17.15%	30.22%	0	**63.09**%	1
One Instance, One Sense (1c1instance)	0%	0%	**0.072**	0%	95.35
Semcor MFS	47.89%	35.65%	-	-	1
WordNet MFS	**57.95%**	**43.14%**	-	-	1

Table 7.6: Evaluation results in SemEval-2013 Task 13 for all 20 nouns with multiple (graded) sense assignments allowed. The first row lists results of our system. Parameters of our system were $N = 100, t_{wf} = 2$. Best system performances and best baselines are marked bold.

was able to beat a simple baseline assigning one sense per word, and improved even further with merging of "fuzzy" clusters. The best system configurations according to this first evaluation also performed competitively in a subset of the SemEval-2013 WSI task. Since these are very promising results, the next chapter discusses possible further use cases of our system that build on an alignment of the induced sense inventory to ontologies.

8 An Outlook: From Word Senses to a Proto-Ontology

To give an introductory example up-front, consider the following sentence in which multiple words are to be disambiguated:

Thomas and Mario are strikers playing in Munich.[1]

There are several ambiguous words in this example: *Thomas* and *Mario*, as well as *striker* and *to play*. While the first two are so-called *entities*, more specifically *Named Entities (NE)*, the latter two are words as found in an ordinary dictionary. In a specific context like this one, all four words are unambiguous: *Thomas* refers to the soccer player *Thomas Müller*, *Mario* to *Mario Gómez*[2], *striker* refers to a soccer player and the verb *to play* to the specific sense of performing a sport.

Yet, most approaches to WSD will have problems disambiguating the word *striker*: While the former can refer to an *employee on strike* (rather unlikely in this context), it can also refer to a *hitter* in a cricket game, which is distinct from a striker in a soccer game. Since WSD systems rarely have access to world knowledge, such as the fact that *Bayern Munich is a soccer club* and that *Munich is a short form of the same club*, it is hard to know for such systems which sense applies here.

With ontological resources such as DBPedia [Bizer, 2014], FreeBase or YAGO [Suchanek et al., 2007] having reached a critical mass of useful information [Auer et al., 2007], a possible solution is to link word-sense inventories to such ontologies. This has, in fact, been discussed only recently in the literature, as e.g. in [Moro et al., 2014]. They even go one step further and formulate the hypothesis that the *lexicographic knowledge* as used in WSD can bring in useful information for Entity Linking (EL) systems as well.

8.1 Bridging The Gap Between Unstructured And Structured Resources

Thinking beyond the combination of manually built dictionaries and ontologies, [Hovy et al., 2013] discuss this from a more abstract point of view: They propose the complemental use of both *structured* and *unstructured* resources. For this, they categorized knowledge resources in four categories following their level of structure:

- *Unstructured resources* such as specific text corpora, or the entire web as a whole. These are the resources that contain most knowledge, yet they are not directly accessible to machines.
- *Thesauri*: collections of related terms, often specifically focused on synonyms and antonyms.
- *Taxonomies*, which add a hierarchy level to thesauri, specifically *is-a* relations such as *striker is-a soccer player*.

[1] This example has been taken from [Moro et al., 2014].

[2] Note that at the time this example was first mentioned, Gómez was indeed the only player named Mario playing for Munich. In fact, this has since changed with the transfer of Mario Götze to Munich in 2013.

- *Ontologies*, representing the highest level of structure and providing a "fully-structured knowledge model, including concepts, relations of various kinds, and possibly, rules and axioms". Notably, these often provide *lexical forms* of such concepts that linking them to terms that express it by means of language.

While, according to them, ontologies provide information of the highest quality, they come with several problems that can indeed be compared to the problems we mentioned as the very problems of hand-crafted lexical resources: Creation and maintenance effort, coverage (i.e. the problem of covering too few domains) and missing up-to-date information.

A current trend in the NLP research community motivated by this observation is to view structured and unstructured resources as being *complementary*: fully-structured ontologies provide high-quality information, but require manual creation and maintenance by linguistic experts. Thesauri, on the other hand, can be computed in a fully automatic manner, as done by our system using the notion of Distributional Semantics, and can therefore be easily adapted to various domains and be updated on a regular basis. However, they lack the full depth of information found in ontologies.

A possible solution is therefore to bridge the gap between unstructured or semi-structured resources such as thesauri, and fully structured resources, namely ontologies: With the possibility of automatically inducing a word-sense inventory along with a full disambiguation model from a Distributional Thesaurus, as described in this thesis, this would on one hand facilitate the unification of WSD and EL systems. On the other hand, it could help overcoming the knowledge-acquisition bottleneck that fully-structured ontologies are facing, and assist linguists in creating domain-adapted ontologies or indicate when new concepts have developed and should be introduced to an explicit ontology[3] (as e.g. suggested in [Widdows and Dorow, 2002]).

In this chapter, we sketch a novel approach to building a *hybrid aligned* resource based on a thesaurus, forming what we call a *proto ontology*.

8.2 From Thesauri to a Proto-Ontology

The path from unstructured resources to thesauri has already been anticipated by previous chapters: By utilizing Distributional Semantics, it is possible to compute a *Distributional Thesaurus* in a fully automatic manner. As a next step, we here briefly discuss the possibility of linking word-sense inventories induced from such thesauri to ontological resources. Specifically, we show how EL systems can benefit from this step.

To pick up the example from above, the word *striker* is by our system correctly disambiguated as **striker.2** (see Table 8.3), which refers to the concept of a soccer player, as opposed to e.g. a cricket player. This is done based on stochastic information that e.g. the words *play*, *Munich* or *Mario* significantly co-occur with words from the cluster **striker.2**. This knowledge is extracted fully automatically in an unsupervised manner, without relying on databases of any sort. In effect, this step is way to mitigate the *Knowledge Acquisition Bottleneck*, and to bring in world knowledge in an automated manner.

In a previous step, **striker.2** can be linked to the ontological concept *forward*[4] from FreeBase. This linking procedure would require two main steps: *candidate generation* and *candidate scoring*. In the first step, candidate concepts that *may* correspond to an induced sense, such as striker.2, are gathered by e.g. retrieving all concepts that have an equal name or alias (cf. Table 8.2 for aliases of the FreeBase concept *forward*). In a second step, these may be scored according to various possible heuristics, including

[3] Take, for example, the word *tablet*: As of version 3.1, WordNet does not mention a sense referring to a computing device, though this sense has been in common use since 2010.

[4] `http://www.freebase.com/m/02sdk9v` (last accessed: 2015/04/30)

Proto concept	Top cluster words	Is-a labels
striker.0	shortstop, pitcher, infielder, outfielder, catcher	position, fielder, player
striker.1	actor, actress, singer, journalist, musician	artist, professional, people, figure
striker.2	midfielder, defender, goalkeeper, footballer, **forward**	sport, player, team, member

Table 8.1: Induced proto concepts for *striker*. Note that the third sense clearly refers to *soccer players*, while the first sense can rather be identified with a *cricket player*. Also, **striker.2** specifically names *forward* as similar item, providing a clear link to the equally-named FreeBase concept. Is-a labels on the right may help to both generate as well as score possible candidate concepts from the ontology.

matching of cluster words to name and aliases, and scoring of the concept description (as done in Table 8.3). This linking procedure for an induced sense inventory and an ontology like FreeBase must be performed only once.

Field	Contents
title	**forward**
notable type	`/soccer/football_position`
description	Forwards are the players on an association football team who play nearest to the opposing team's goal, and are therefore most responsible for scoring goals. (...)
aliases	ponta, **striker**, attacker

Table 8.2: Excerpt from FreeBase entry for concept *forward*. Note the alias *striker*, which can be used to retrieve *candidate* concepts that may be linked to a specific induced word sense. Using the description text, as well as the type information referring to football (as opposed to e.g. cricket), the corresponding entry in the induced word sense inventory can be determined fairly accurately.

Sense	$P(s\|C_{forward})$	$P(s\|C_{example})$
striker.0	2.173×10^{-11}	3.364×10^{-3}
striker.1	4.348×10^{-12}	2.398×10^{-8}
striker.2	0.999...	0.997...

Table 8.3: Scores for induced senses of *striker* (denoted by s) for description of the FreeBase concept *forward* and the example sentence from above (denoted by $C_{forward}$ and $C_{example}$, resp.). While the alias *striker* of the FreeBase concept already provides a clear link to **striker.2**, these scores could be used to *verify* linking of this candidate concept.

When this is done, and *striker* is (to use our own system as example) disambiguated correctly to **striker.2**, this additional background knowledge is tremendously useful for further linking of the entities *Thomas*, *Mario* and *Munich*: In FreeBase, there are 20,571 people with first name *Thomas*. However, there are only 53 such people that are linked to the concept *forward*, which was identified by our exemplary WSD system. While this would not fully disambiguate the entity *Thomas*, it severely cuts down the number of alternatives to choose from and can provide useful hints for EL systems.

9 Implementation Details

To process the large text corpus required by our WSI algorithm, a high parallelization level was necessary in order to cut processing time to a minimum. Since many variables were unknown in the beginning, especially number and type of distributional features that would facilitate a well-performing WSI system, we incrementally improved our system step by step, using manual inspection and evaluation results. This was only possible having a pipeline that can process all data in a relatively short period of time.

The foundation of our implementation was the JoBimText project[1] [Biemann and Riedl, 2013], which can be used to compute word similarities based on raw text. This project makes heavy use of Hadoop[2], a massively parallel computation framework based on the MapReduce programming model [Dean and Ghemawat, 2004]. This programming model, in a nutshell, allows the specification of operations on splits of data that can be executed entirely independently from each other. To allow for faster prototyping of new components, which were necessary to build a Word Sense Induction and Disambiguation system on top of this pipeline, and for quick experimental adjustments in the thesaurus computations, we realized a light-weight alternative implementation of the JoBimText pipeline in Spark[3] (for an introductory paper on Spark, see [Zaharia et al., 2010]). Though Spark does not directly implement the MapReduce programming model itself, it allows a superset of the operations supported in MapReduce. Notably, Spark provides a native Scala interface, allowing a higher-level programming than Java, which is the standard programming interface for Hadoop. Additionally, Spark proved to be easier to set up and run on local machines, which enabled a more test-driven incremental development using small-scale datasets.

Our cluster in total consisted of 24 nodes, with around 20 virtual processor cores each. This cluster setting led to a total memory amount of about 700 GB and 65 terabytes of Hadoop Distributed File System (HDFS)[4] disk storage. Average processing time to extract dependency features for all nouns in a Wikipedia corpus using the MaltParser [Nivre et al., 2006] and to compute a distributional thesaurus using these was under a day. Aggregating context clues over extracted sense clusters for 100 words (and 100 similar terms per word) took an average of one hour.

The WSI pipeline consisted of several steps (see Figure 9.1), which have already roughly been outlined in Section 3.3. First, a large text corpus is extracted from a recent Wikipedia dump. Then, context features are extracted using a Unstructured Information Management Architecture (UIMA) [5] pipeline executed on Hadoop, consisting of DKPro[6] analysis component wrappers for an OpenNLP[7] tokenizer, segmentizer, lemmatizer and a dependency parser from the MaltParser project. In a third step, word similarities are computed using either JoBimText, or our Spark-based alternative implementation. An illustration of this computation for a *distributional thesaurus*, see Figure 3.2, or the project's documentation[8]. For an in-depth explanation of the steps required to compute distributional similarities using a JoBimText-like pipeline, see [Biemann and Riedl, 2013]. The result can be converted into a sparsely connected word graph, by connecting only words with a similarity over a certain threshold, discarding all other edges,

1 https://sourceforge.net/projects/jobimtext/ (last accessed: 2015/04/30)
2 https://hadoop.apache.org/ (last accessed: 2015/04/30)
3 http://spark.apache.org/ (last accessed: 2015/04/30)
4 For an introductory paper on HDFS, see [Shafer et al., 2010]
5 http://uima.apache.org/ (last accessed: 2015/04/30)
6 https://www.ukp.tu-darmstadt.de/research/current-projects/dkpro/ (last accessed: 2015/04/30)
7 http://opennlp.apache.org/ (last accessed: 2015/04/30)
8 http://maggie.lt.informatik.tu-darmstadt.de/jobimtext/documentation/ (last accessed: 2015/04/30)

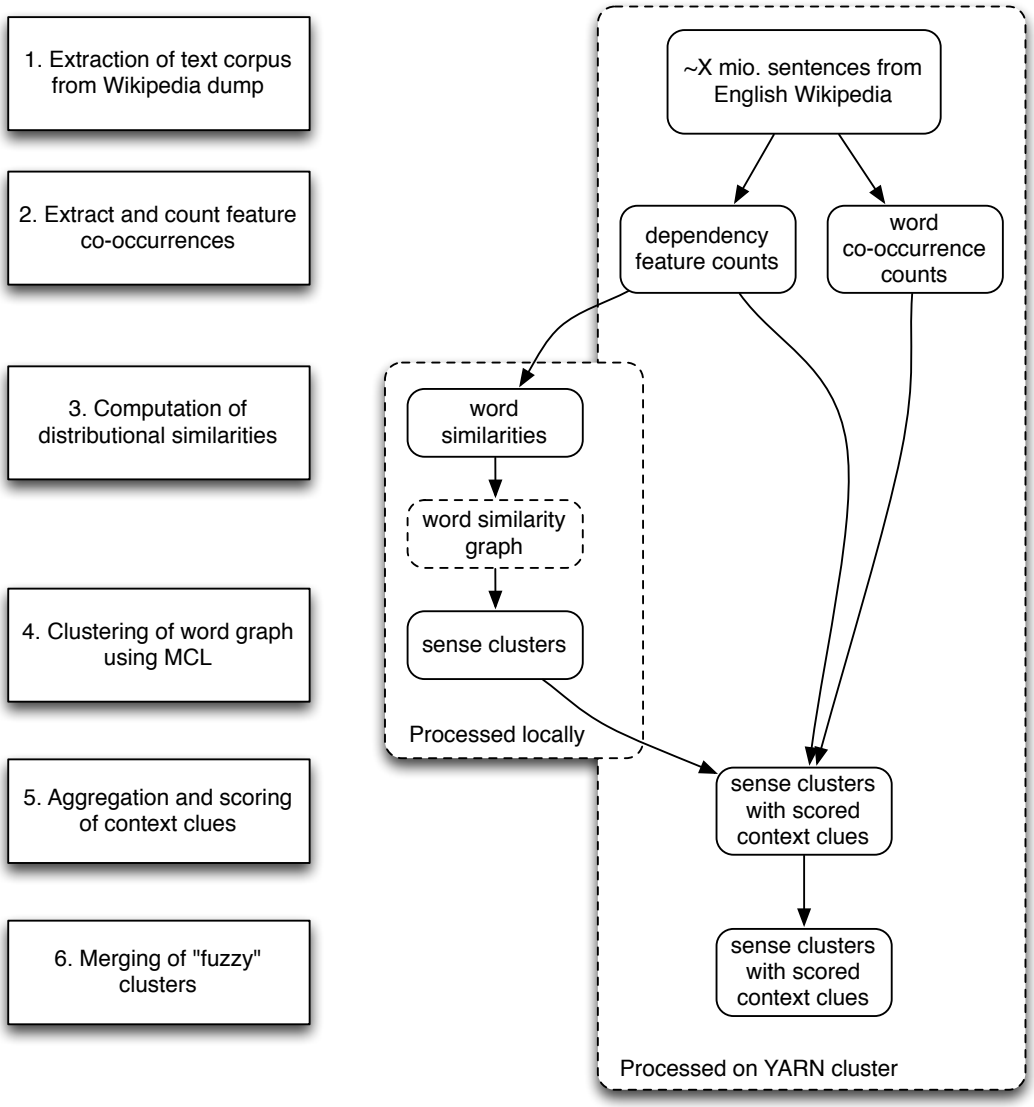

Figure 9.1: Order and dependencies of several steps required in producing a word-sense inventory along with a contextualization model (consisting of scored *context clues*)

and weighting every remaining edge with the respective (symmetric) word similarity. This graph is then, in a fourth step, clustered using a custom, efficient MCL Java implementation internally using a sparse matrix representation. *Context clues* are aggregated for each sense cluster as described in Section 5.2. Lastly, ”fuzzy” clusters with similar context clues are merged to improve disambiguation precision. The result is a fixed, yet domain-aware, word-sense inventory with an augmented, Bayesian disambiguation model.

The disambiguation of words in context is relatively straightforward: Using the pre-computed sense inventory, extracted features from the word context can be compared to the context clues found in the respective sense clusters of the ambiguous word. The pre-computed scores for the extracted context features, as found in each sense cluster, are then combined (usually by mere multiplication), and the sense with the highest score is chosen. Alternatively, if several senses have an equally high score, multiple senses can be assigned. For an illustration of this process, see Figure 9.2.

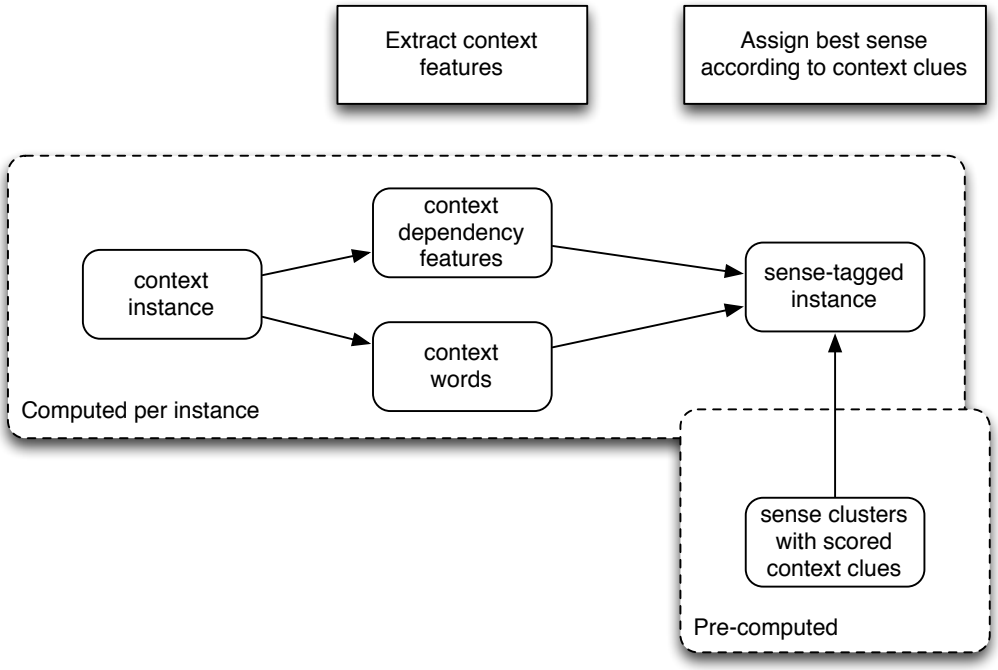

Figure 9.2: Word Sense Disambiguation using the induced sense inventory augmented with context clues

10 Conclusion and Future Work

10.1 Conclusion

We introduced a fully unsupervised Word Sense Induction and disambiguation (WSID) system that does not rely on any existing sense inventories. Also, it does not require any training data for learning a model for contextual disambiguation of induced senses; the underlying probabilistic model is constructed fully automatically from the same text corpus that is used to induce senses. In contrast to most other state-of-the-art WSI systems that merely cluster instances, it produces a fixed sense inventory that allows for consistent sense-labeling for varying inputs. Also, our approach scales to very large text corpora in the magnitude of terabytes, yet we showed that it compares competitively with other unsupervised systems. In the SemEval-2013 *Word Sense Induction for Graded and Non-Graded Senses* subtask, the most recent task in this line, it is the only system that outperforms the baseline's Fuzzy NMI score in the cluster comparison evaluation when applied only to nouns. Also, it receives the highest Fuzzy B-Cubed score in the same evaluation, even though we did not perform any optimizations in our system specific to the SemEval-2013 challenge.

Hence, to answer the first question of the hypothesis: To induce a word sense inventory for a large part of the vocabulary, we used a scalable Distributional Similarity computation algorithm. To obtain word senses, the resulting word similarity graph is clustered individually for each target word.

Reliable disambiguation of word instances in context (the second research question) is achieved by using context clues of cluster words as representative clues for each word sense.

Limitations

Notable limitations of our system are the following: First, the results with respect to the WSD setting of the SemEval-2013 WSI task, in which induced senses are mapped to WordNet senses, are only in the midfield among other participants. A potential reason for this is that we did not cluster instance contexts directly, as other higher-performing systems did. However, our induced senses are therefore not specific to these instances, and can be re-used in other contexts.

Also, while our system performs the sense induction step in an unsupervised manner, it nonetheless depends on supervised language processing components, namely a POS tagger, a lemmatizer and a dependency parser. While it is possible to replace these components by their unsupervised counterparts [Riedl et al., 2014], this was not conducted in the scope of this thesis.

10.2 Future Work

Building on the notion of having a fixed word-sense inventory produced by the WSI algorithm, future work should primarily focus on mapping the induced sense inventory to existing lexical resources such as WordNet or the SUMO ontology. This, for one, allows the re-use of unsupervised WSI systems in scenarios where well-defined senses from such resources are required. Even more importantly, it would also open up new possibilities to enhance existing sense inventories with useful statistical information that can be extracted automatically from large text corpora.

Another interesting question would be whether other or additional distributional features could improve performance of our WSI system. As both the Wikipedia-based, as well as the SemEval-2013 evaluation showed, the addition of dependency features for contextualization consistently improved WSD performance. This indicates that incorporating more distributional features may improve WSD further. There are two major possible additions: N-grams of various window sizes may complement the high accuracy of dependency features. Using larger values for n would also allow to add more disambiguation context than dependency features could provide. But also using smaller windows for co-occurrence features, which are the foundation of our contextualization algorithm, may improve accuracy due to the higher sensitivity to local context. Notably, [Pedersen, 2000] reported an improved accuracy for their Bayesian classifier using an ensemble of such co-occurrence features with windows of varying sizes.

Lastly, since the ultimate goal of WSI systems is to facilitate *unsupervised* means for WSD, a possible setting of the proposed system should be evaluated that neither makes use of complex POS taggers, nor of dependency parsers. Since these are usually trained on hand-labeled data, removing the need for these would yield a true, fully unsupervised WSI system that could be applied even when no such advanced components are available.

10.3 Summary

In this thesis, we introduced an unsupervised system for Word Sense Disambiguation (WSD) that utilizes the notion of Distributional Semantics to induce senses from large text corpora. In contrast to most state-of-the-art WSI systems, our system provides a fixed sense inventory that can be used to label previously unseen instances. Disambiguation of words in context was done by using context features of words from each sense cluster.

To optimize the quality of the Distributional Thesaurus (DT) used by our system, we introduced a pseudoword-based evaluation to measure the contribution of noise in computed word similarities.

We also introduced a minimally supervised, large-scale evaluation method for WSID systems based on Wikipedia's link structure. The same evaluation was used to tune system parameters, without the need for any hand-annotated test data or manual performance evaluation. This automatic evaluation method is especially interesting as, in contrast to other recent WSI evaluations, it provides a large amount of sense-labeled contexts (several hundreds in some cases) for a large amount of words. Namely, we provided 100 sense-tagged, high-quality contexts for 100 nouns, which is nearly a magnitude larger than e.g. compared to the SemEval-2013 WSI task, which provided 22-100 contexts for only 20 nouns. Yet building our evaluation dataset involved only an hour of manual inspection.

Lastly, we showed that our system performs competitively with regard to task 13 of the SemEval-2013 challenge, a recent, well-known WSI evaluation.

Glossary

AI Artificial Intelligence. 7

CBC Clustering by Committee. 34

CW Chinese Whispers. 31

DT Distributional Thesaurus. 47

EL Entity Linking. 67

HDFS Hadoop Distributed File System. 70

HDP Hierarchical Dirichlet Process. 26

IR Information Retrieval. 7

kNN k-Nearest Neighbor. 15

LDA Latent Dirichlet allocation. 26

LDOCE Longman Dictionary of Contemporary English. 15

LL Log-likelihood ratio. 23

LMI Lexicographer's Mutual Information. 23

MaxEnt Maximum-entropy. 14, 15

MCL Markov Chain Clustering. 29–31, 77

MFS Most frequent sense. 14

ML Machine Learning. 11

MT Machine Translation. 7

NE Named Entity. 67

NPMI Normalized Pointwise Mutual Information. 35

PMI Pointwise Mutual Information. 34

SemCor Semantic Concordance. 15

SVD Singular Value Decomposition. 34

UIMA Unstructured Information Management Architecture. 70

WCSS within-cluster sum of squares. 25

WNDCG weighted variant of Normalized Discounted Cumulative Gain. 53

WSD Word Sense Disambiguation. 7, 8, 11, 14, 24

WSI Word Sense Induction. 8, 11, 27, 47

List of Figures

List of Tables

Bibliography

[Agirre and Edmonds, 2007] Agirre, E. and Edmonds, P. G. (2007). *Word Sense Disambiguation: Algorithms and Applications*. Springer-Verlag.

[Agirre et al., 2006] Agirre, E., Martínez, D., Lacalle, O. L. D., and Soroa, A. (2006). 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, USA. Association for Computational Linguistics.

[Agirre and Soroa, 2007] Agirre, E. and Soroa, A. (2007). Semeval-2007 Task 02: Evaluating Word Sense Induction and Discrimination Systems. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, SemEval '07, pages 7–12, Prague, Czech Republic. Association for Computational Linguistics.

[Amigó et al., 2009] Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, F. (2009). A Comparison of Extrinsic Clustering Evaluation Metrics Based on Formal Constraints. *Information Retrieval*, 12(4):461–486.

[Auer et al., 2007] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. (2007). DBpedia: A nucleus for a Web of open data. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference*, volume 4825 LNCS, pages 722–735, Busan, Korea.

[Baldwin et al., 1998] Baldwin, A. B. B., Bagga, A., and Baldwin, B. (1998). Algorithms for scoring coreference chain. In *First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566, Pennsylvania, USA.

[Baskaya et al., 2013] Baskaya, O., Sert, E., Cirik, V., and Yuret, D. (2013). AI-KU: Using Substitute Vectors and Co-Occurrence Modeling for Word Sense Induction and Disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 300–306, Atlanta, Georgia, USA. Association for Computational Linguistics.

[Biemann, 2006] Biemann, C. (2006). Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 73–80, New York City, USA.

[Biemann, 2007] Biemann, C. (2007). *Unsupervised and Knowledge-free Natural Language Processing in the Structure Discovery Paradigm*. PhD thesis, Universität Leipzig.

[Biemann, 2010] Biemann, C. (2010). Co-Occurrence Cluster Features for Lexical Substitutions in Context. In *Proceedings of the 5th Workshop on TextGraphs in conjunction with ACL*, pages 55–59, Uppsala, Sweden. Association for Computational Linguistics.

[Biemann, 2012] Biemann, C. (2012). Turk Bootstrap Word Sense Inventory 2.0: A Large-Scale Resource for Lexical Substitution. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 4038–4042, Istanbul, Turkey.

[Biemann and Riedl, 2013] Biemann, C. and Riedl, M. (2013). Text: Now in 2D! a framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.

[Bizer, 2014] Bizer, C. (2014). DBpedia - A Large-scale , Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web*, 1:1–5.

[Bollacker et al., 2008] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD 08 Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, Vancouver, Canada.

[Bordag, 2006] Bordag, S. (2006). Word Sense Induction : Triplet-Based Clustering and Automatic Evaluation. In *11th Conference of the European Chapter of the ACL*, pages 137–144, Trento, Italy.

[Bordag, 2008] Bordag, S. (2008). A Comparison of Co-occurrence and Similarity Measures As Simulations of Context. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*, volume 4919 LNCS, pages 52–63, Haifa, Israel. Springer-Verlag.

[Bruce and Wiebe, 1994] Bruce, R. and Wiebe, J. (1994). Word-Sense Disambiguation Using Decomposable Models. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, page 8, Las Cruces, New Mexico. Association for Computational Linguistics.

[Bruce and Wiebe, 1999] Bruce, R. and Wiebe, J. (1999). Decomposable Modeling in Natural Language Processing. *Computational Linguistics*, 25(2):195–207.

[Chklovski and Mihalcea, 2002] Chklovski, T. and Mihalcea, R. (2002). Building a Sense Tagged Corpus with Open Mind Word Expert. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, pages 116–122, Philadelphia, PA. Association for Computational Linguistics.

[Church and Hanks, 1990] Church, K. W. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

[Curran, 2002] Curran, J. R. (2002). Ensemble methods for automatic thesaurus extraction. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*, volume 10, pages 222–229, Philadelphia, PA. Association for Computational Linguistics.

[Curran, 2003] Curran, J. R. (2003). *From Distributional to Semantic Similarity*. PhD thesis, University of Edinburgh.

[Daelemans et al., 1998] Daelemans, W., Bosch, A. V. D., and Zavrel, J. (1998). Forgetting Exceptions is Harmful in Language Learning. *Machine Learning*, 41:31.

[Dean and Ghemawat, 2004] Dean, J. and Ghemawat, S. (2004). MapReduce: Simplied Data Processing on Large Clusters. In *Proceedings of 6th Symposium on Operating Systems Design and Implementation*, pages 137–149. San Francisco, CA.

[Dorow, 2007] Dorow, B. (2007). *A Graph Model for Words and their Meanings*. PhD thesis, Universität Stuttgart.

[Dunning, 1993] Dunning, T. (1993). Accurate Methods for the Statistics of Surprise and Coincidence. *Computational Linguistics*, 19:61–74.

[Evert, 2005] Evert, S. (2005). *The Statistics of Word Cooccurrences Word Pairs and Collocations*. PhD thesis, Universität Stuttgart.

[Fellbaum, 1998] Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*, volume 71 of *Language, Speech, and Communication*. MIT Press.

[Firth, 1957] Firth, J. R. (1957). A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, 1952-59:1–32.

[Florian et al., 2002] Florian, R., Cucerzan, S., Schafer, C., and Yarowsky, D. (2002). Combining Classifiers for word sense disambiguation. *Natural Language Engineering*, 8(4):327–341.

[Gale et al., 1992] Gale, W., Church, K., and Yarowsky, D. (1992). One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*, pages 233–237, Harriman, New York. Association for Computational Linguistics.

[Gençay and Fagan, 2011] Gençay, R. and Fagan, S. (2011). *An Introduction to Textual Econometrics*, pages 133–154. Taylor & Francis, Boca Raton, US.

[Gliozzo et al., 2013] Gliozzo, A., Biemann, C., Riedl, M., Coppola, B., Glass, M. R., Hatem, M., and Heights, Y. (2013). JoBimText Visualizer : A Graph-based Approach to Contextualizing Distributional Similarity. In *Proceedings of TextGraphs-8 Graph-based Methods for Natural Language Processing*, volume 1, pages 6–10, Seatlle, Washington.

[Harris, 1954] Harris, Z. S. (1954). Distributional structure. *Word*, 10(23):146–162.

[Hope and Keller, 2013a] Hope, D. and Keller, B. (2013a). MaxMax: A Graph-based Soft Clustering Algorithm Applied to Word Sense Induction. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, pages 368–381, Samos, Greece. Springer-Verlag.

[Hope and Keller, 2013b] Hope, D. and Keller, B. (2013b). UoS: A Graph-Based System for Graded Word Sense Induction. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, number 1, pages 689–694, Atlanta, Georgia, USA.

[Hovy et al., 2013] Hovy, E., Navigli, R., and Ponzetto, S. P. (2013). Collaboratively built semi-structured content and Artificial Intelligence: The story so far. *Artificial Intelligence*, 194:2–27.

[Ide and Suderman, 2004] Ide, N. and Suderman, K. (2004). The American National Corpus First Release. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, pages 1681–1684, Lisbon, Portugal.

[Järvelin and Kekäläinen, 2002] Järvelin, K. and Kekäläinen, J. (2002). Cumulated Gain-based Evaluation of IR Techniques. *ACM Transactions on Information Systems*, 20(4):422–446.

[Jurgens and Klapaftis, 2013] Jurgens, D. and Klapaftis, I. (2013). Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 290–299, Atlanta, Georgia, USA. Association for Computational Linguistics.

[Jurgens and Stevens, 2011] Jurgens, D. and Stevens, K. (2011). Measuring the Impact of Sense Similarity on Word Sense Induction. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, pages 113–123, Edinburgh, Scotland. Association for Computational Linguistics.

[Kilgarriff et al., 2000] Kilgarriff, A., Kilgarriff, A., Rosenzweig, J., and Rosenzweig, J. (2000). English Senseval: Report and Results. In *In Proceedings of the 2nd International Conference on Language Resources and Evaluation*, Athens, Grece.

[Kilgarriff et al., 2004] Kilgarriff, A., Rychlý, P., Smrz, P., and Tugwell, D. (2004). The Sketch Engine. *Linguistics*, 1:105–116.

[Klein et al., 2002] Klein, D., Toutanova, K., Ilhan, H. T., Kamvar, S. D., and Manning, C. D. (2002). Combining Heterogeneous Classifiers for Word-Sense Disambiguation. In *Proceedings of the ACL-02 Workshop on Word Sense Disambiguation: Recent Successes and Future Directions*, volume 8, pages 74–80, Philadelphia, PA. Association for Computational Linguistics.

[Kučera and Francis, 1967] Kučera, H. and Francis, W. N. (1967). *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI.

[Lau et al., 2013] Lau, J. H., Cook, P., and Baldwin, T. (2013). unimelb: Topic Modelling-based Word Sense Induction. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, volume 2, pages 307–311, Atlanta, Georgia, USA.

[Lau et al., 2012] Lau, J. H., Cook, P., McCarthy, D., Newman, D., Baldwin, T., and Computing, L. (2012). Word Sense Induction for Novel Sense Detection. In *Proceedings of the 13th Conference of the European Chapter of the ACL (EACL 2012)*, pages 591–601, Avignon, France. Association for Computational Linguistics.

[Leacock et al., 1998] Leacock, C., Miller, G. a., and Chodorow, M. (1998). Using Corpus Statistics and WordNet Relations for Sense Identification. *Computational Linguistics*, 24(1):147–165.

[Leacock et al., 1993] Leacock, C., Towell, G., and Voorhees, E. (1993). Corpus-based Statistical Sense Resolution. In *Proceedings of HLT '93 Proceedings of the workshop on Human Language Technology*, pages 260–265, Princeton, New Jersey. Association for Computational Linguistics.

[Lee and Ng, 2002] Lee, Y. K. and Ng, H. T. (2002). An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*, volume 10, pages 41–48, Philadelphia, PA. Association for Computational Linguistics.

[Lenat, 1995] Lenat, D. B. (1995). CYC: A Large-scale Investment in Knowledge Infrastructure. *Communications of the ACM*, 38(11):33–38.

[Lesk, 1986] Lesk, M. (1986). Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, Toronto, Ontario, Canada. ACM.

[Lin, 1998] Lin, D. (1998). Automatic retrieval and clustering of similar words. In *ACL '98 Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 768–774, Montreal, Quebec, Canada. Association for Computational Linguistics.

[Lin and Dyer, 2010] Lin, J. and Dyer, C. (2010). Data-Intensive Text Processing with MapReduce. In *Synthesis Lectures on Human Language Technologies*, volume 3, pages 1–177, Boulder, Colorado. Association for Computational Linguistics.

[Mallery, 1988] Mallery, J. C. (1988). *Thinking About Foreign Policy: Finding an Appropriate Role for Artificially Intelligent Computers*. PhD thesis, MIT, Cambridge, MA.

[Manandhar, 2010] Manandhar, S. (2010). SemEval-2010 Task 14 : Word Sense Induction & Disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval 2010)*, pages 63–68, Uppsala, Sweden. Association for Computational Linguistics.

[Matuszek et al., 2006] Matuszek, C., Cabral, J., Witbrock, M., and Deoliveira, J. (2006). An introduction to the syntax and content of Cyc. In *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*, volume 3864, pages 44–49, Stanford, CA.

[Miller and Charles, 1991] Miller, G. a. and Charles, W. G. (1991). Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

[Miller et al., 1993] Miller, G. a., Leacock, C., Tengi, R., and Bunker, R. T. (1993). A Semantic Concordance. In *Proceedings of the Workshop on Human Language Technology - HLT '93*, pages 303–308, Princeton, New Jersey. Association for Computational Linguistics.

[Mooney, 1996] Mooney, R. J. (1996). Comparative Experiments on Disambiguating Word Senses: An Illustration of the Role of Bias in Machine Learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 82–91, Philadelphia, PA.

[Moro et al., 2014] Moro, A., Raganato, A., Navigli, R., Informatica, D., and Elena, V. R. (2014). Entity Linking meets Word Sense Disambiguation : a Unified Approach. *Transactions of the Association for Computational Linguistics*, 2:231–244.

[Navigli, 2009] Navigli, R. (2009). Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

[Navigli and Ponzetto, 2012] Navigli, R. and Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

[Ng, 1997a] Ng, H. (1997a). Getting serious about word sense disambiguation. In *Tagging Text with Lexical Semantics: Why, What, and How? Workshop*, Washington, D.C.

[Ng, 1997b] Ng, H. T. (1997b). Exemplar-Based Word Sense Disambiguation: Some Recent Improvements. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 208–213, Providence, USA.

[Ng and Lee, 1996] Ng, H. T. and Lee, H. B. (1996). Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-Based Approach. In *Proceedings of the 34th Meeting of the Association for Computational Linguistics (ACL-96)*, pages 40–47, Santa Cruz, CA. Association for Computational Linguistics.

[Nivre et al., 2006] Nivre, J., Hall, J., and Nilsson, J. (2006). MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, volume 6, pages 2216–2219, Genoa, Italy. European Language Resources Association (ELRA).

[Pantel and Lin, 2002] Pantel, P. and Lin, D. (2002). Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, volume 41, pages 613–619, New York City, USA. ACM Press.

[Pedersen, 2000] Pedersen, T. (2000). A Simple Approach to Building Ensembles of Naive Bayesian Classifiers for Word Sense Disambiguation. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference (NAACL 2000)*, pages 63–69, Seattle, Washington.

[Pedersen, 2007] Pedersen, T. (2007). *Learning Probabilistic Models of Word Sense Disambiguation*. PhD thesis, Southern Methodist University, Dallas, TX.

[Pianta et al., 2002] Pianta, E., Bentivogli, L., and Girardi, C. (2002). MultiWordNet: developing an aligned multilingual database. In *Proceedings of the First International Conference on Global WordNet*, pages 21–25, Mysore, India.

[Pradhan et al., 2007a] Pradhan, S., Loper, E., Dligach, D., and Palmer, M. (2007a). SemEval-2007 Task 17: English Lexical Sample, SRL and All Words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic.

[Pradhan et al., 2007b] Pradhan, S. S., Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2007b). OntoNotes: A unified relational semantic representation. In *ICSC 2007 International Conference on Semantic Computing*, pages 517–524, Irvine, CA.

[Pradhan and Xue, 2009] Pradhan, S. S. and Xue, N. (2009). OntoNotes: The 90% Solution. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 57–60, New York City, USA.

[Purandare and Pedersen, 2004] Purandare, A. and Pedersen, T. (2004). Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces. In *Proceedings of the Conference on Computational Natural Language Learning*, pages 41–48, Boston, MA, USA.

[Rada et al., 1989] Rada, R., Mili, H., Bicknell, E., and Blettner, M. (1989). Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(1):17–30.

[Riedl et al., 2014] Riedl, M., Alles, I., and Biemann, C. (2014). Combining Supervised and Unsupervised Parsing for Distributional Similarity. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1435–1446, Dublin, Ireland.

[Sanderson, 1994] Sanderson, M. (1994). Word sense disambiguation and information retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, Dublin, Ireland. Springer-Verlag.

[Schütze, 1992] Schütze, H. (1992). Dimensions of meaning. In *Proceedings of the 1992 ACM/IEEE conference on Supercomputing*, Minneapolis, Minnesota, USA. IEEE Computer Society Press.

[Schütze, 1998] Schütze, H. (1998). Automatic Word Sense Discrimination. *Computational Linguistics*, 24(1):97–123.

[Schütze and Pedersen, 1995] Schütze, H. and Pedersen, J. O. (1995). Information Retrieval Based on Word Senses. In *Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, NV USA.

[Shafer et al., 2010] Shafer, J., Rixner, S., and Cox, A. L. (2010). The Hadoop distributed filesystem: Balancing portability and performance. In *ISPASS 2010 - IEEE International Symposium on Performance Analysis of Systems and Software*, pages 122–133, White Plains, NY.

[Steinbach et al., 2000] Steinbach, M., Karypis, G., Kumar, V., and Others (2000). A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526, Boston, MA, USA.

[Suchanek et al., 2007] Suchanek, F. M., Kasneci, G., and Weikum, G. (2007). Yago: A Core of Semantic Knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, Banff, Alberta, Canada. ACM.

[Teh et al., 2006] Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical Dirichlet Processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

[Towell and Voorhees, 1998] Towell, G. and Voorhees, E. M. (1998). Disambiguating highly ambiguous words. *Computational Linguistics*, 24(1):125–145.

[Tratz et al., 2007] Tratz, S., Sanfilippo, A., Gregory, M., Chappell, A., and Whitney, P. (2007). PNNL : A Supervised Maximum Entropy Approach to Word Sense Disambiguation. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 264–267, Prague, Czech Republic.

[van Dongen, 2000] van Dongen, S. (2000). *Graph clustering*. PhD thesis, University of Utrecht.

[Van Rijsbergen, 1979] Van Rijsbergen, C. J. (1979). *Information Retrieval*, volume 30. Butterworth.

[Vasilescu et al., 2004] Vasilescu, F., Langlais, P., and Lapalme, G. (2004). Evaluating Variants of the Lesk Approach for Disambiguating Words. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, pages 633–636, Lisbon, Portugal.

[Véronis, 1998] Véronis, J. (1998). A study of polysemy judgements and inter-annotator agreement. In *Programme and advanced papers of the Senseval workshop*, Herstmonceux Castle (England).

[Véronis, 2004] Véronis, J. (2004). HyperLex: Lexical cartography for information retrieval. *Computer Speech and Language*, 18:223–252.

[Vinh et al., 2010] Vinh, N., Epps, J., and Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854.

[Wagner, 2006] Wagner, C. (2006). Breaking the Knowledge Acquisition Bottleneck Through Conversational Knowledge Management. *Information Resources Management Journal*, 19(1):70–83.

[Wee, 2010] Wee, H. L. (2010). Word Sense Prediction Using Decision Trees. Technical report, Department of Computer Science, National University of Singapore.

[Widdows and Dorow, 2002] Widdows, D. and Dorow, B. (2002). A graph model for unsupervised lexical acquisition. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Taipei, Taiwan.

[Yarowsky, 1993] Yarowsky, D. (1993). One Sense Per Collocation. In *Proceedings of the workshop on Human Language Technology - HLT '93*, pages 266–271, Princeton, New Jersey.

[Yarowsky, 1995] Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196, Cambridge, Massachusetts.

[Yuret, 2012] Yuret, D. (2012). FASTSUBS: An efficient and exact procedure for finding the most likely lexical substitutes based on an n-gram language model. *IEEE Signal Processing Letters*, 19(11):725–728.

[Zaharia et al., 2010] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark : Cluster Computing with Working Sets. In *HotCloud'10 Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, page 10.

[Zhao and Karypis, 2001] Zhao, Y. and Karypis, G. (2001). Criterion Functions for Document Clustering: Experiments and Analysis. Technical report, Department of Computer Science and Engineering, University of Minnesota, Minneapolis.

[Zhao and Karypis, 2002] Zhao, Y. and Karypis, G. (2002). Comparison of Agglomerative and Partitional Document Clustering Algorithms. Technical report, Department of Computer Science and Engineering, University of Minnesota, Minneapolis.