Linguistic Category Induction and Tagging Using the Paradigmatic Context Representations with Substitute Words

by

Mehmet Ali Yatbaz

Dissertation submitted to the
Department of Computer Engineering
for fulfillment of the requirements for the degree of

Doctor of Philosophy

at

Koç University
Thesis Supervisor: Deniz Yuret

February, 2014

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a doctoral dissertation by

Mehmet Ali Yatbaz

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Thesis Committee:

_____

Assoc. Prof. Deniz Yuret

_____

Asst. Prof. T. Metin Sezgin

_____

Assoc. Prof. S. Serdar Kozat

_____

Asst. Prof. Gülşen Cebiroğlu Eryiğit

_____

Prof. Aylin Küntay

Date: _____

ABSTRACT

This thesis introduces a new paradigmatic representation of word contexts. Paradigmatic representations of word context are constructed from the potential substitutes of a word, in contrast to syntagmatic representations, which are constructed from the properties of neighboring words. The potential substitutes are calculated by using a statistical language model that is trained on raw text without any annotation or supervision. Thus, each context is represented as a distribution of substitute words. This thesis introduces models with different properties that can incorporate the new paradigmatic representation, and discusses the applications of these models to the *tagging task* in natural language processing (NLP).

In a standard NLP tagging task, the goal is to build a model in which the input is a sequence of *observed words*, and the output, depending on the level of supervision, is a sequence of *cluster-ids* or predefined *tags*. We define 5 different models with different properties and supervision requirements. The first model ignores the identity of the word, and clusters the substitute distributions without requiring supervision at any level. The second model represents the co-occurrences of words with their substitute words, and thus incorporates the word identity and context information at the same time. To construct the co-occurrence representation, this model discretizes the substitute distribution. The third model uses probabilistic voting to estimate the distribution of tags in a given context. Unlike the first and second models, this model requires the availability of a *word-tag dictionary* which can provide all possible tags of each given word. The fourth model proposes two extensions to the standard HMM-based tagging models in which both the word identity

and the dependence between consecutive tags are taken into consideration. The last one introduces a generative probabilistic model, the noisy channel model, for the tagging tasks in which the word-tag frequencies are available. In this model, each context C is modeled as a distinct channel through which the speaker intends to transmit a particular tag T using a possibly ambiguous word W. To reconstruct the intended message (T), the hearer uses the distribution of possible tags in the given context $\Pr(T|C)$ and the possible words that can express each tag $\Pr(W|T)$.

The models are applied and analyzed on NLP tagging tasks with different characteristics. The first two models are tested on unsupervised part-of-speech (POS) induction in which the objective is to cluster syntactically similar words into the same group. The probabilistic voting model is tested on the morphological disambiguation of Turkish, with the objective of disambiguating the correct morphological parse of a word, given the available parses. The HMM-based model is applied to the part-of-speech tagging of English, with the objective of determining the correct POS tag of a word, given the available tags. Finally, the last model is tested on the word-sense disambiguation of English, with the objective of determining the correct sense of a word, given the word-sense frequencies.

ÖZETÇE

Bu tez kelime bağlamlarını temsil etmek için yeni bir düşey bağıntı tanımlamaktadır. Bir kelimenin düşey bağıntısı kelimenin bağlamında değiştirim sonucu onun yerine gelebilen olası kelimelerin oluşturduğu bağıntıdır. Öte yandan yatay bağıntı bir kelimenin öncesindeki ya da sonrasındaki kelimeler arasında kurulan bağıntıdır. Bir kelimenin yerini alabilecek olası kelimeler işlenmemiş veri üzerinde eğitilmiş bir istatistiksel dil modeli ile hesaplanmaktadır. Sonuç olarak kelime bağlamları, o bağlamda görülebilecek olası kelime dağılımları ile temsil edilmektedir. Bu tez bahsi geçen yeni düşey bağıntıyı kullanabilen farklı doğal dil işleme modelleri tanımlamakta ve bu modellerin doğal dil işlemede kullanılan farklı dizisel etiketleme problemleri üzerindeki uygulamalarını göstermektedir.

Doğal dil işleme problemlerindeki dizisel etiketlemenin temel amacı verilen bir kelime dizisine birebir denk gelen dizisel etiketileri bulamaktadır. Bu nedenle modeller girdi olarak kelime dizisi almakta ve çıktı olarak her kelimeye bir etiket gelecek şekilde bir etiket dizisi vermektedir. Öğreticisiz modellerde çıktı dizisi her kelimeye ait küme isimleri iken öğreticili modellerde çıktı dizisi her kelimeye ait önceden tanımlanmış etiketlerdir. Bu tezde 5 farklı model tanımlanmaktadır. İlk model öğreticisiz bir modeldir ve olası kelime dağılımlarını kullanarak kelimeleri kümelemeyi amaçlamaktadır. İkinci model verilen bir kelime ile o kelimeye ait olası kelimelerin birlikte görülme sıklıklarını modelliyen öğreticisiz bir modeldir. Üçünci model kelimenin yerini alabilecek kelimeleri kullanarak olasılıksal oylama yapan bir modeldir. Bu model ilk iki modelin aksine, her kelimenin olası etiketlerine ihtiyaç duyan öğreticili bir modeldir. Dördüncü model dizisel etiketleme probleminde sıklıkla kullanılan saklı Markof modelleriyle birlikte kullanılabilen 2 yöntem

önermektedir. Bir önceki model gibi bu model de her kelimeye ait olası etiketlere ihtiyaç duyar. Tezdeki son model gürültülü kanal modelidir ve bu model gürültülü kanal ve alınan mesajı kullanarak esas gönderilmek istenen mesajı bulmayı amaçlar. Bu modelde her bağlam bir kanal, her kelime alınan mesaj ve kelimeye ait etiket ise gönderilmek istenen esas mesajdır.

Tezin son kısmında yukarıda bahsi geçen modeller farklı özelliklerdeki etikeleme problemlerine uygulanmıştır. İlk iki model öğreticisiz sözcük türü bulma problemine uygulanmıştır. Olasılıksal oylama modeli ise Türkçe ekbiçim belirsizliği giderme problemi üzerinde denenmiştir. Saklı Markof modeline dayanan yöntemler ise öğreticili sözcük türü bulma problemine uygulanmıştır. Son olarak gürültülü kanal modeli kelime anlam belirsizliği giderme problemi üzerinde denenmiştir.

ACKNOWLEDGMENTS

I would like to thank to Leyla Yatbaz, Naci Yatbaz and my advisor, Deniz Yuret. I'am fortunate enough to have these three people in my life. Not a single sentence, paragraph, thesis or life-time can express my feelings about them.

I would like to thank to Mustafa Arif Karaman and Demet Çek for bringing joy and their support into my life.

I would like to thank T. Metin Sezgin, S. Serdar Kozat, Gülşen Cebiroğlu Eryiğit and Aylin Küntay for being in my thesis defense committee and for their constructive comments.

Finally, I would like to thank all of my family members and friends for supporting me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

The amount of online raw (unstructured) text is rapidly increasing as a result of extensive usage and coverage of the Internet. We will propose an unsupervised learning framework for NLP tagging tasks[1], which, under some constraints, will allow us to incorporate vast amounts of raw data into the learning phase by means of a paradigmatic representation and a statistical language model.

Natural language problems such as word-sense disambiguation, part-of-speech (POS) induction, POS tagging, morphological disambiguation, or named-entity recognition have three common properties: (1) there exists an observed *word sequence*, (2) a hidden *tag sequence* is paired with that observed text, and (3) the objective is to determine the hidden tag sequence values that match the given observation. Therefore, we group the NLP problems with these three properties under a general framework which is called the *tagging task*. Thus any problem with the above-mentioned structure can be reduced to this general framework and solved by the procedures that will be described in the thesis.

Supervised methods in the NLP literature require both the input word sequence and corresponding tag sequence for training. However, it takes far more effort to manually annotate or organize this vast amount of data by expert human annotators than to directly use the data. The main drawback of the supervised approach is the difficulty of acquiring sufficiently large amounts of training data, also known as the *knowledge acquisition bottleneck*. For example, Yarkowsky and Florian (2002) report that each successive doubling

[1]These tasks are also known as disambiguation or induction tasks depending on the availability of the tagged corpus.

of the training data for word-sense disambiguation problem, i.e. selecting the correct sense of a word from the possible senses, leads to only a 3–4% error reduction within their experimental range. Banko and Brill (2001) experiment with the problem of selection among confusable words, and show that the learning curves do not converge even after a billion words of training data. They suggest active-learning, unsupervised learning or lower levels of supervision to take advantage of large datasets when labeling is expensive. Yuret (2004) observes that in a supervised Naive Bayes WSD system trained on SemCor, approximately half of the test instances do not contain any of the contextual features (e.g. neighboring content words or local collocation patterns) observed in the training data. SemCor is the largest publicly available corpus of sense-tagged text, and has only about a quarter million sense-tagged words, while the largest English raw text data available has about 10^{12} words.

Moreover, not all of the languages have the variety of tagged resources that English has. For example, the largest available corpus of Turkish morphological disambiguation problem, i.e. selecting the correct affix parse of a word, is a corpus of 1 million semi-automatically tagged words, and because of the semi-automatic tagging this training corpus itself has inconsistencies (Hakkani-Tür et al., 2002). On the other hand, the largest untagged Turkish web corpus consists of 440 million Turkish words derived from a variety of domains (Sak et al., 2008), while the supervised corpus contains 1 million words from a specific news domain.

1.1 Relationships Between Linguistic Units

Relationships between linguistic units can be classified into two types: syntagmatic (concerning positioning), and paradigmatic (concerning substitution). Syntagmatic relations determine which units can combine to create larger groups and paradigmatic relations determine which units can be substituted for one another. Figure 1.1 illustrates the paradigmatic vs syntagmatic axes for words in a simple sentence and their possible substitutes.

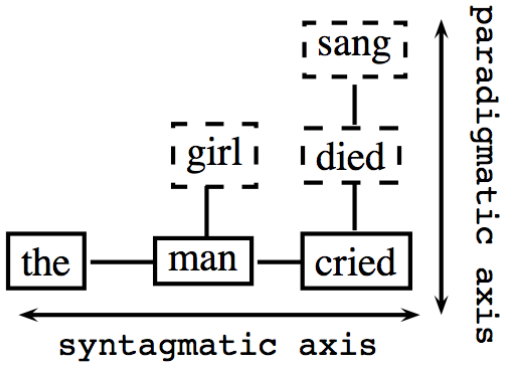


Figure 1.1: Syntagmatic vs. paradigmatic axes for words in a simple sentence (Chandler, 2007).

In this thesis, we represent the paradigmatic axis directly by building *substitute distributions* for each word position in the text using an n-gram statistical language model. The domain of a substitute distribution represent words in the vocabulary, and the magnitudes represent the probability of occurrence in the given position. Note that the substitute distribution for a word position (e.g. the second word in Fig. 1.1) is a function of the context only (i.e. "the ___ cried"), and does not depend on the word that does actually appear there (i.e. "man") given the context. Thus substitute vectors represent *individual word contexts*, not word types. We refer to the use of features based on substitute distributions as *paradigmatic representations of word context* and the use of features based on neighboring words as *syntagmatic representation of word context*.

The two examples below illustrate the advantages of the paradigmatic representation in uncovering similarities where no overt similarity that can be captured by a syntagmatic representation exists. The word *director* from the first sentence and the word *chief* from the second one have no common neighbors in their 4-gram neighborhood. The paradigmatic representation captures the similarity of these words by suggesting the same top substitutes for both (the numbers in parentheses give substitute probabilities):

(1) *"Pierre Vinken, 61 years old, will join the board as a nonexecutive* **director** *Nov. 29."*
director: chairman (.8242), director (.0127), directors (.0127) . . .

(2) *". . . Joseph Corr was succeeded by Frank Lorenzo ,* **chief** *of parent Texas Air ."*
chief: chairman (.09945), president (.0031), directors (.0012) . . .

The high probability substitutes reflect both semantic and syntactic properties of the context. Top substitutes for "director" and "chief" are not only nouns, but specifically nouns compatible with the semantic context. Top substitutes for the word "the" in the first example consist of words that can act as determiners: its (.9011), the (.0981), a (.0006), ...

1.2 Scope

The goal of this thesis is to incorporate paradigmatic context representations, constructed by using the raw text, into the natural language processing (NLP) tagging tasks such as word-sense disambiguation, part-of-speech tagging, morphological disambiguation, and part-of-speech induction. We assume that words in a similar context have similar properties, and are hence interchangeable without affecting the meaning or the structure of the original sentences. We use a statistical language model (SLM) to construct substitute distribution in a given context, and use these distributions to improve the accuracy of tagging tasks. In order to observe the effectiveness of our framework on special cases of the tagging tasks, we conducted experiments on certain well known NLP problems, as follows:

1. **Word-sense disambiguation of English:** The senses of the substitute words are used to determine the correct sense of the target ambiguous word. Substitute words can be used to determine the natural clustering of the word senses.

2. **Part-of-speech tagging:** The correct POS tag of the target word is determined using the likely substitutes. Alternatively, substitute words are used to prune the word-tag dictionary. This implicitly decreases the ambiguity level of the dictionary, and therefore improves the estimation quality of the probabilistic models in certain cases, such as expectation maximization.

3. **Morphological disambiguation:** The correct morphological parse of the target word is determined using the likely substitutes. This problem is a more complex version of the English POS tagging problem because the number of unique tags are much higher then the tags in a typical English corpus (Yuret and Ture, 2006). This high number of tags causes data sparseness.

4. **Part-of-speech induction:** In contrast to part-of-speech tagging and morphological disambiguation, this problem induces word categories without using any tagged data. It clusters words according to their contexts, and the number of clusters can be set to any arbitrary number or determined by the data itself.

1.3 Overview

This thesis presents a framework that enables the usage of raw text together with a statistical language model to improve the performance of tagging tasks. We will define the basics of the tagging tasks, and present concrete results of this framework on some well-known tagging tasks. These problems are presented in respective chapters, together with the review of the relevant literature, as follows:

Chapter 2 explains the usage of the raw text together with a statistical language model. It presents the algorithm to output the likely substitutes of a target word in a context with the help of a statistical language model constructed from the raw text data. This replacement algorithm plays the central role in the methods proposed in the following chapters.

Chapter 3 defines the different types of contexts and reviews the literature of context representations.

Chapter 4 introduces 4 tagging models that can incorporate substitute distributions. This chapter details the assumptions, weaknesses, strengths and required level of supervision of each model. The rest of the thesis presents selected applications of the models that are defined in this chapter.

Chapter 5 defines the part-of-speech (POS) disambiguation problem and introduces several applications of the substitute words in POS disambiguation problem.

Chapter 6 defines the morphological disambiguation problem and introduces an application of the substitute words in the morphological disambiguation problem.

Chapter 7 defines the word-sense disambiguation (WSD) problem and introduces the noisy channel model that determines the correct sense of an English word in a given context with the help of substitute words.

Chapter 8 formulates the unsupervised POS induction problem, reviews the literature, and describes a POS induction system that models the co-occurrence of words and their substitutes to construct word or word-instance clusters.

1.4 Contribution

This thesis introduces a new context representation by using the substitute word distributions that are calculated according to a statistical language model and presents the possible ways of incorporating these distributions into the well known NLP tagging tasks. This thesis makes the following research contributions:

Representation

- It introduces a new paradigmatic context representation by using the substitute words. This new representation enables the integration of unlabeled raw text into the NLP tagging tasks.

- The substitute distributions are constructed by using an n-gram statistical language model. Compared to syntagmatic representations they do not suffer as much from the data sparsity as the context size, n, becomes larger.

- Calculation of substitute distributions does not require any tagged corpus therefore can be applied to any language.

- Unlike syntagmatic representations it can capture the semantic or syntactic similarity between word-instances even when they appear in totally different contexts.

- Substitute distribution reflects both semantic and syntactic properties of the context. Therefore, it can be used both with semantic and syntagmatic tagging tasks.

Models and Applications

- I introduce a clustering model that can construct both word-based and instance-based clusters of substitute distributions. The model is tested and analyzed on the unsupervised part-of-speech induction task of English.

- I demonstrate that representing contexts with paradigmatic representations and modeling co-occurrences of word and context types give superior results in unsupervised part-of-speech induction task when compared to its syntagmatic counterpart.

- I extend the co-occurrence modeling framework to incorporate morphological and orthographic features and test the co-occurrence model with features on unsupervised part-of-speech induction task of 15 languages.

- I introduce an instance based POS induction system that can handle ambiguous words and is competitive with the word-based systems in overall accuracy.

- I present a probabilistic voting model that estimates the tag distribution of a specific context by using the corresponding substitute distribution. This model is tested and analyzed on the Turkish morphological disambiguation task.

- I introduce two methods that create artificial sentences using the substitute distributions and incorporate them into the HMM based probabilistic tagging models. Both of the methods are tested and analyzed on the part-of-speech disambiguation task of English.

- I introduce a noisy channel model that is a probabilistic generative model and seamlessly integrates substitute distributions into the model building process. The model is tested and analyzed on the word-sense disambiguation task of English.

Chapter 2

CALCULATING SUBSTITUTE DISTRIBUTIONS

In this thesis, we represent the paradigmatic axis directly by building the *substitute distribution* for each word position in the text. Note that the substitute distribution is a function of the context only and is *conditionally* independent of the target word in a given context. Substitute distributions represent *individual word contexts*, not words. Table 2.1 shows the most likely substitutes of words in a sample sentence calculated with a 4-gram language model. This chapter provides the details of calculating the substitute distribution of a given context.

will:	*will* (0.9985), *would* (0.0007), *to* (0.0006), *also* (0.0001), . . .
join:	*join* (0.6528), *leave* (0.2140), *oversee* (0.0559), *head* (0.0262), *rejoin* (0.0074), . . .
the:	*its* (0.9011), *the* (0.0981), *a* (0.0006), . . .
board:	*board* (0.4288), *company* (0.2584), *firm* (0.2024), *bank* (0.0731), *strike* (0.0030), . . .

Table 2.1: The substitute word distributions (with probabilities in parentheses) for some of the positions in the example sentence *"Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29."* according to a 4-gram language model.

2.1 Statistical Language Modeling

A statistical language model (SLM) is a probability distribution over a set of strings where the probability of each string is estimated by using a large amount of raw text of the target

language. The estimation procedure for a given string can become cumbersome since (1) not all strings are observed in the raw text and (2) the strings can be arbitrarily long. Thus, the probability of a given string $S = w_1 w_2 w_3, \ldots, w_m$ that maximizes the likelihood can be approximated by

$$P(S) = \prod_{i=1}^{m} P(w_i | w_{i-1}, \ldots, w_1) \tag{2.1}$$

$$\propto \prod_{i=1}^{m} P(w_i | w_{i-1}, \ldots, w_{i-n+1}) \tag{2.2}$$

$$\propto \prod_{i=1}^{m} \frac{C(w_i, w_{i-1}, \ldots, w_{i-n+1})}{C(w_{i-1}, \ldots, w_{i-n+1})} \tag{2.3}$$

where $n-1$ is the number of words that the target word is conditioned by, and $C(S)$ is the frequency of the string S in the corpus. Eq. 2.2 can be derived from Eq. 2.1 by the Markov assumption (i.e., each word depends only on the previous $n-1$ words). Eq. 2.2 can be estimated by various smoothing techniques, whose detailed analysis together with the review of SLM, can be found in (Chen and Goodman, 1999; Rosenfeld, 2000; Goodman, 2001). Eq. 2.3 presents the maximum-likelihood estimation of Eq 2.2.

2.2 Language Model Quality

In this thesis the quality of a language model is measured by *perplexity*, which is defined to be the geometric average of inverse probabilities of the words in the test corpus (T), and is explicitly formulated as

$$perplexity(T) = \left(\prod_{i=1}^{N} \frac{1}{P(w_i | w_{i-1}, \ldots, w_{i-n+1})} \right)^{\frac{1}{n}} \tag{2.4}$$

where w_i represents the i^{th} word and N is the total number of words in T. Each word in Equation 2.4 depends on the previous $n-1$ words, where n is the n-gram order of the language model.

The log_2 of perplexity is *entropy* which is simply the average number of bits per word required to encode the test corpus. Thus the least ambiguous language model[1] on a given test corpus will have the lowest entropy and perplexity. Perplexity calculations do not require any annotated corpus or supervision; therefore, one can tune the parameters of a language model for a development corpus with the help of Eq.2.4.

2.3 The Substitute Distribution of a Word Context

It is best to use both the left and the right context when estimating the probabilities for potential lexical substitutes. For example, in *"He lived in San Francisco suburbs."*, the token *San* would be difficult to guess from the left context but it is ascertained easily from the right context. We define c as the $(2n-1)$-word window centered on the target word, w_0, position: $w_{-n+1} \dots w_0 \dots w_{n-1}$. The probability of a substitute word w in a given context c can be estimated as:

$$P(w_0 = w|c_w) \propto P(w_{-n+1} \dots w_0 \dots w_{n-1}) \quad (2.5)$$

$$= P(w_{-n+1})P(w_{-n+2}|w_{-n+1}) \dots P(w_{n-1}|w_{-n+1}^{n-2}) \quad (2.6)$$

$$\approx P(w_0|w_{-n+1}^{-1})P(w_1|w_{-n+2}^{0}) \dots P(w_{n-1}|w_0^{n-2}) \quad (2.7)$$

where w_i^j represents the sequence of words $w_i w_{i+1} \dots w_j$. In Equation 2.5, $P(w|c_w)$ is proportional to $P(w_{-n+1} \dots w_0 \dots w_{n-1})$ because the words of the context are fixed. Since in Equation 2.6 the terms without w_0 are identical for each substitute, they have been dropped in Equation 2.7. Finally, because of the Markov property of n-gram language models, only the closest $n-1$ words are used in the experiments.

Certain terms near sentence boundaries were truncated in Equation 2.7. Specifically, shorter n-gram contexts were used at the beginning of the sentence, and the tokens beyond the end-of-sentence were dropped.

[1]Goodman (2001) refers to the least ambiguous model as the *True model*.

Estimating substitute words for each context is expensive due to the large number of vocabulary words that need to be considered as substitutes. For example, the Web 1T dataset (Brants and Franz, 2006a) contains 13.5 million unique words, and thus has 13.5 million candidate substitutes for every context. In order to keep computations feasible, we limit the set of candidate substitute words by replacing low-frequency vocabulary words with the unknown-word tag, <unk> . Moreover, most of the time, 90% of the total probability is distributed among the top 100 substitutes of a target word. To calculate the most likely k substitutes of a context c, a naive substitute algorithm would need to calculate $\Pr(w|c)$ for all w in the vocabulary. To take advantage of the skewed distribution structure of substitute words, we use the FASTSUBS algorithm (Yuret, 2012) which calculates $\Pr(w|c)$ for only the most likely candidates in c instead of for all of the words in the vocabulary. Thus it can generate the top k most likely substitutes of a given word much faster than the naive substitute algorithms. On a typical 2012 workstation, FASTSUBS accomplishes the top 100 substitute generation task for a 1M-word corpus in about 5 hours, while a naive algorithm that looks at the whole vocabulary would take more than 6 days. To calculate the final substitute vectors used in the rest of this study, the probability vectors for each position were normalized to add up to 1.0 .

2.4 Sampling from a Substitute distribution

Substitute vectors are continuous representation of contexts. However, it is also possible to construct discrete representation of contexts by using the substitute distributions. To discretize the continuous representation, one can sample random substitutes (with replacement) from a substitute word distribution, and then represent the corresponding context using these small number of substitutes as discrete contextual features. The sample space

of the substitute word distribution is the vocabulary of the language model[2]. In effect, we are using substitute word distributions and the sampled random substitutes as *contextual features* that represent the properties of a word's position.

[2]Sampled substitutes might include the unknown word tag <unk> that represents the words outside the fixed size vocabulary of the language model. For example, proper nouns typically have <unk> as a substitute.

Chapter 3

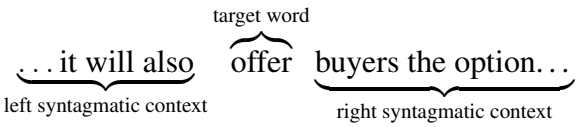
REPRESENTING WORD CONTEXT

This chapter demonstrates the various contextual representations of a word in tagging literature, and introduces the substitute words as an alternative to the current contextual representations. In this thesis, words in the vocabulary of a corpus are referred to as a *word*, and each occurrence of a word is referred to as a *word instance* or *instance*. The contextual representations can be categorized, depending on the way these representations incorporate the local context information of the target word or instance, into three groups: (1) syntagmatic representations, (2) Hidden Markov Models (HMM), and (3) paradigmatic representations.

3.1 Syntagmatic Representation

In the syntagmatic representation, the context, which is called the "frame," is defined with the neighboring words, typically co-occurrences with a single word on the left or a single word on the right word (e.g., **the** *dog* **is***;* **the** *cat* **is**) (Schütze and Pedersen, 1993; Redington et al., 1998; Mintz, 2003; St Clair et al., 2010; Lamar et al., 2010b; Maron et al., 2010). Turney and Pantel (2010) give a broad overview of syntagmatic approaches and their applications within the Vector Space Modeling framework. Depending on the way they incorporate co-occurrences, these models can perform hard (word-based) or soft (instance-based)

clustering.



Schütze and Pedersen (1993) represent the context of a word instance by concatenating its left and right *co-occurrence vectors*. These vectors are calculated for each word by using the left and the right neighbors of the word instances, thus characterizing the distribution of the left and right neighbors of the word. One constraint of this representation is that it represents words rather than word instances, and hence it performs word-based clustering.

Véronis (2004) constructs an undirected word co-occurrence graph of a given target word and its neighboring words. The nodes of the graph represent words, and an undirected edge between two nodes represents the co-occurrence of the corresponding two words in the context of the target word. Each edge is assigned a weight according to the co-occurrence frequency of the nodes (words). Although this model can handle larger context window sizes, it discards the order of the context words.

Mintz (2003) shows on a subset of the child-directed speech corpus (CHILDES) (MacWhinney, 2000) that the non-adjacent most frequent bi-gram frames are useful for language learners on the syntactic categorization of instances. For example, the instances that are observed at "_" in the frame "**the** _ **is**" are usually assigned to the same category. Using the "top-45 frequent frames" (the 45 most frequent frames), Mintz achieved an average of 98% unsupervised accuracy[1]. The main limitation of the top-45 frequent frames is that, because of the sparsity, they can analyze only 6% of the word instances on average. Another drawback is that word instances even with one common neighbor could not exchange information.

[1]Unsupervised accuracy is defined as the number of hits (when two intervening word instances observed in the frame are from the same category) divided by number of false alarms (when two intervening word instances observed in the frame are from different categories).

St Clair et al. (2010) extend the work of Mintz (2003) and introduce the flexible bi-gram frames which represent the context by using the left and the right bi-grams separately. As a result, instances with a common left or right bi-gram can exchange information and might be grouped together. For instance, two instances that are observed at "'_" in "**the** _ **is**" and "**a** _ **is**" can be assigned the same category because of the shared right bi-gram "**is**". Using a feed forward connectionist model, they showed that the flexible frames are statistically better than the frequent frames in terms of the supervised accuracy[2]. They also showed that representing instance contexts only with the left or the right bi-gram is statistically better than the frequent frames but worse than the flexible frames in terms of supervised accuracy. Neither Mintz (2003) nor St Clair (2010) report any results with contexts larger than a bi-gram, since, as the context is enriched, the re-occurrence frequency of a frame decreases, causing data sparsity (Manning and Schütze, 1999).

3.2 HMM

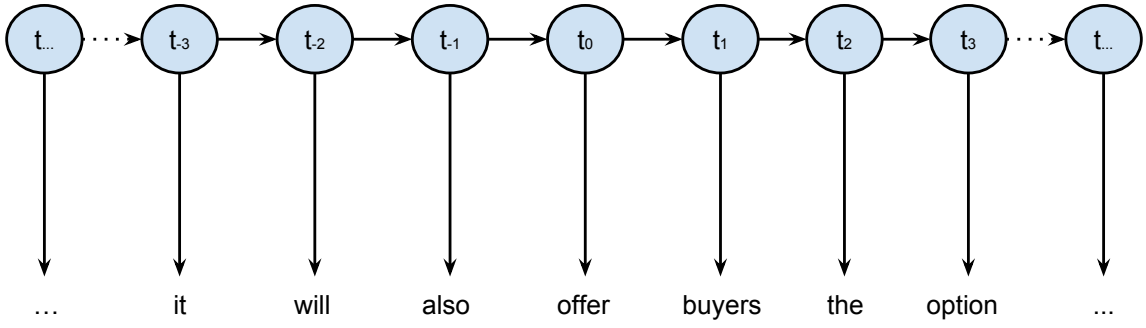


Figure 3.1: A bi-gram HMM-based context of an example sentence.

Prototypical HMMs in tagging literature use a bi-gram structure in which instances are

[2]In order to perform meaningful comparisons, they used all of the frequent frames instead of the top-45 ones.

generated by latent categories and which learns the latent category sequence to generate the given word sequence (see Figure 3.1) (Brown et al., 1992; Blunsom and Cohn, 2011; Goldwater and Griffiths, 2007b; Johnson, 2007a; Ganchev et al., 2010; Berg-Kirkpatrick and Klein, 2010; Lee et al., 2010). The tagging literature has focused on the first- and second-order HMMs since the higher-order HMMs have additional complicating factors[3] and requires more complex training procedures (Johnson, 2007a). Depending on the design and the training procedure, the HMMs can group words or word instances.

3.3 Paradigmatic Representation

In the paradigmatic representation, context is represented by the distribution of substitute words. Schütze (1995) incorporates paradigmatic information by concatenating the left co-occurrence and the right co-occurrence vectors of the right and the left token, respectively, and groups the tokens with similar vectors. The vectors from the neighbors include potential substitutes. Similarly, Schütze and Pedersen (1993) define the words that frequently co-occur together as the *syntagmatic associates* and words that have similar left and right neighbors as the *paradigmatic parallels*.

Sahlgren (2006) gives a detailed analysis of paradigmatic and syntagmatic relations in the context of word-space models used to represent the word meanings. Sahlgren's paradigmatic model represents words using co-occurrence counts of their frequent neighbors, in contrast to his syntagmatic model that represents words using counts of contexts (documents, sentences) they occur in. Our substitute vectors do not represent words at all, but they represent *contexts of word instances* determined from the probabilities of likely substitutes. Sahlgren finds that more nearest neighbors share the same part of speech in the word-spaces built by frequent neighbor vectors than in the word-spaces built by context

[3]The number of parameters in a prototypical HMM increases quadratically with the HMM order.

vectors.

$$\ldots \text{it will also} \overbrace{\text{offer}}^{\text{target word}} \text{buyers the option} \ldots$$

$$\text{give}(.7032)$$

$$\text{help}(.1203)$$

$$\text{attract}(.1081)$$

$$\vdots$$

In this thesis, the context of a given word is represented by a distribution of substitute words. The entries of the substitute distribution reflect how likely it is to observe each substitute in the context of the target word. The above example illustrates the likely substitutes of the target word *offer*. We calculate the most likely substitute words in a given context using a statistical language model. Our paradigmatic representation is related to the second-order co-occurrences used in (Schütze, 1995). Our method improves on his foundation in two aspects: (1) it uses a 4-gram language model rather than bi-gram statistics, (3) it includes the whole corpus vocabulary rather than the most frequent 250 words. More importantly, rather than simply concatenating the left and right context vectors of the target word, we use a statistical language model.

Chapter 4

MODELS

In this thesis, we focus on incorporating the paradigmatic representation into the models with different levels of supervision, and on applying these models to certain *tagging tasks* in natural language processing (NLP). In a standard NLP tagging task, the goal is to build a model in which the input is a sequence of *observed words*, and the output, depending on the level of supervision, is a sequence of *cluster-ids* or predefined *tags*.

Tagging

The definition of tags depends on the nature of the tagging task at hand. For example: if the task is word-sense-disambiguation (WSD), then the output (of the tagging task solver) would be the corresponding sense sequence of the input; if the task is part-of-speech (POS) disambiguation, then the output would be the corresponding POS tag sequence of the input; or if it the task is morphological disambiguation, then the output would be the corresponding morphological parse sequence of the input.

The models in this section can be placed, depending upon how much supervision is available, in several categories.

Supervised tagging

Supervised tagging models require an annotated training corpus in which word sequences and the corresponding tag sequences are both available. Thus the output tag sequence of a given test input will only contain the tags that are observed in the annotated training corpus. Supervised models usually perform better in comparison to unsupervised models. However, since the annotation process of raw text is generally expensive and error-prone, resource-poor languages lack annotated corpora. For example, two different occurrences of the word *the* in the same context (i.e., "*Meet _ Press*") are erroneously labeled as determiner (*DT*) and proper-noun (*NNP*) in the 1M-word Wall Street Journal Section of

the Penn Treebank (PTB) corpus. Most importantly, children do not require any annotated corpora to perform similar tasks during their language acquisition.

Unsupervised tagging

Unsupervised tagging models observe only the word sequences, without requiring any annotated corpora. These models have no information regarding the possible tag set or the number of available tags, and hence output only the cluster-id sequences of the input word sequence. The main advantage of these models is that they can be applied to any language. The number of clusters is not fixed, and clusterings with different granularities can be constructed. Finally, they perform constructive learning which, arguably, better resembles the language acquisition by children (Tomasello, 2009).

There are also models that do not require the tag sequences of observed input sequences, yet require some level of supervision for performing the desired NLP task[1]. The level of required supervision can be further assigned one of two categories according to the availability of (1) the word-tag dictionary and (2) the word-tag distribution. The word-tag dictionary has the mapping information between words and their possible tags, while the word-tag distribution provides the frequency information of a word with a particular tag, or vice versa. Similarly to the supervised models, the output tag sequences of these models can only contain the tags that are observed in the word-tag dictionary or distribution. Table 4.1 presents the word-tag dictionary entries and word-tag distributions of the words *of* and *a* according to the PTB corpus.

Notation

We denote a sequence of n words (or a sentence) by $w_0 \dots w_n$ (or w_0^n in an abbreviated form), its corresponding context sequence by $c_0 \dots c_n$ (c_0^n), and output tag or cluster-id sequence by $t_0 \dots t_n$ (t_0^n). The substitute distribution of the i^{th} word w_i in a sequence is denoted by s_i, and the possible tag set of w_i is denoted by T_{w_i} . In some sections, we use the non-indexed symbols w, c, t, and s to represent, respectively, the target word, its context,

[1]The NLP literature has been using different naming conventions for these models, such as "weakly supervised" or "minimally supervised". However, since none of the naming convention clearly specifies the nature of supervision, we explicitly state the level of supervision when referring to these models.

Word	Word-Tag Dictionary	Word-Tag Distribution
of	IN	IN(28334)
	RP	RP(2)
	RB	RB(2)
a	DT	DT(23647)
	SYM	SYM(11)
	JJ	JJ(2)
	LS	LS(2)
	NNP	NNP(2)
	IN	IN(1)

Table 4.1: The word-tag dictionary entries and distributions of the words *of* and *a* in the PTB corpus for POS tagging.

tag (cluster-id), and substitute distribution. Random variables that sample w_i, c_i, and t_i are denoted by W, C, and T. The set of possible substitutes of a word in a given context is represented by S_c. The definition of the tag set depends on the context of the task to which the model is being applied. For example, in the context of WSD, t and T_w stand for the sense and possible senses of w, respectively.

Organization

Section 4.1 and 4.2 describe the possible applications of unsupervised models that cluster words or instances according to their substitute distributions (Section 4.1), or model co-occurrences with their random substitutes that are sampled from the substitute distributions (Section 4.2). Section 4.3 presents a probabilistic voting algorithm that assigns tags to contexts instead of instances, while Section 4.4 introduces two methods that incorporate the substitute distribution into the standard HMM-based probabilistic models. Models in Section 4.3 and 4.4 require the availability of a word-tag dictionary. Section 4.5 presents a

noisy channel model that uses the word-tag distributions to figure out the most likely tag of a given word.

4.1 Model 1: Clustering of the Substitute Distributions

Clustering-based tagging models are members of unsupervised models, and hence they require only the input sequences in order to output the corresponding cluster-id sequences.

Model 1 assumes that w_0^n and t_0^n are independent, given c_0^n. Each context c_i is represented by the substitute word distribution s_i at that position, and tags are assigned to the corresponding substitute distributions. Thus, word-instances with similar substitute distributions are assigned to the same clusters. The similarity between two substitute distributions can be defined by using any kind of function, as long as it satisfies the conditions of non-negativity, identity, symmetry, and triangular inequality.

The main limitation of this model is that the identity or the features of target words are ignored during tagging, because of the independence of w and t for a given c. Another important limitation is that tags t_i and t_{i+1} of the consecutive words w_i and w_{i+1} are independent of each other for the given contexts c_i and c_{i+1}.

The substitute distributions are high-dimensional vectors, and they are problematic with many learning algorithms because of high computational costs and the curse of dimensionality. However, these drawbacks can be overcome by applying a dimensionality reduction algorithm prior to clustering. Finally, the choice of clustering algorithm is highly specific to the tagging task, and needs to be made carefully.

The unsupervised model in the next section removes the independence assumption of w and t for given c by modeling the co-occurrence of the target word with its substitutes and features.

4.2 Model 2: Co-occurrence Modeling

Model 2 constructs Euclidean embeddings for words and their contextual features representing their co-occurrence statistics, and clusters these embeddings to induce word categories. In this section, we combine the substitute distributions of the word context with features of target word within the co-occurrence data embedding (CODE) framework (Globerson et al., 2007; Maron et al., 2010). In the previous section, the clustering model assumes the independence of w and t for a given c, and hence it does not incorporate the target word identity or features into the tagging task. On the other hand, Model 2 relaxes the independence assumption by modeling the joint probability of w and c according to the co-occurrence of w with c.

Model 2 models the pairwise joint distributions between target words and their contextual, morphological, and orthographic features by embedding the frequently co-occurring pairs closer in Euclidean space. In other words, words, substitutes and features that are frequently observed as pairs in the co-occurrence data will have close embeddings while pairs not observed together will have embeddings that are far apart from each other. The final step of the co-occurrence modeling consists of clustering the embeddings in order to assign cluster-ids to words or word-instances. In this behavior, co-occurrence modeling seems very similar to Model 1. However, the main difference is that in co-occurrence modeling both w and c are involved in the construction of embeddings, while in Model 1 the substitute distributions are independent of w for a given c. Similar to Model 1, co-occurrence models require only raw text in order to output the cluster-id sequence of a given input sequence. One limitation of Model 2 is that it assumes the tags of the consecutive words to be independent of each other for given contexts.

As an example, the co-occurrence data in Figure 4.1 consists of word-substitute pairs such as (*W:director*, *S:chairman*) and (*W:chief*, *S:chairman*). Model 2 therefore forces the embeddings of *W:director* and *W:chief* to be close to the embedding of *S:chairman*.

Word	Substitute
⋮	⋮
W:director	S:chairman
W:chief	S:chairman
⋮	⋮
W:Pierre	S:John
W:Frank	S:John
⋮	⋮

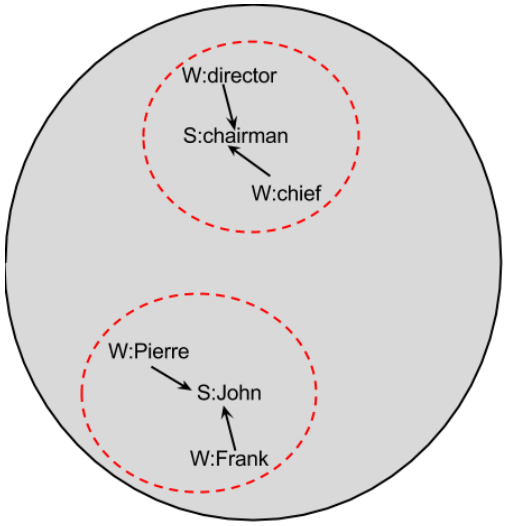


Figure 4.1: The table on the left is the sample input co-occurrence data, and the figure on the right is the final embeddings of the words and substitutes that are observed in this sample co-occurrence data after embedding algorithm converges. To distinguish between the target words and substitute words, we use the prefix *W:* and *S:*, respectively.

Similarly to the former case, the embeddings of *W:Pierre* and *W:Frank* will be close to the embedding of *S:John*, because they are frequently co-occurring pairs. As a result, the final embeddings of *W:director* and *W:chief* will be close to each other, as they share the common substitute *S:chairman*, but will be apart from *W:Pierre* and *W:Frank*, as these lack a common substitute. (Similarly, the embeddings of *W:Pierre* and *W:Frank* will be close to each other because of the substitute *S:John*). After Model 2 constructs embeddings on the sphere, we apply a clustering algorithm on these embeddings to induce word categories.

Section 4.2.1 describe in detail the representation of words and their substitutes as co-occurrence data. Section 4.2.2 describes the CODE embedding algorithm and details the model likelihood and its training procedure in our setup. The spherical optimization (S-CODE) described in (Maron et al., 2010) is used for efficiency and is detailed in Sec-

tion 4.2.2. Finally, Section 4.2.3 describes the different ways in which words and substitutes can be clustered to produce word-based and instance-based clusters.

4.2.1 Co-occurrence Data

To capture the relation between each word and its context, we construct a co-occurrence representation by pairing the words with randomly sampled substitutes. The calculation of substitute distributions and random substitute sampling are detailed in Chapter 2.

Table 4.2 shows an example sentence with random substitutes of each of its words and their pairwise co-occurrence representation input to the co-occurrence embedding algorithm. It is possible (and advantageous) to sample more than one substitute and generate multiple pairs for the same word-context pair as seen in Table 4.2. A target word might appear both as a word and a random substitute. Therefore, to clarify this ambiguity, we prepend "W:" and "S:" to words and substitutes, respectively, in the co-occurrence data.

Word	**Random Substitutes**
Pierre	*Mr.* / *Pierre* / *John*
Vinken	*<unk>* / *Beregovoy* / *Cardin*
,	*,* / *,* / *,*
61	*48* / *52* / *41*
years	*years* / *years* / *years*
old	*old* / *old* / *old*
,	*,* / *,* / *,*
will	*will* / *will* / *will*
join	*head* / *join* / *leave*
the	*its* / *its* / *the*
board	*board* / *company* / *firm*
as	*as* / *as* / *as*
a	*a* / *a* / *a*
nonexecutive	*nonexecutive* / *non-executive* / *nonexecutive*
director	*chairman* / *chairman* / *director*
Nov.	*April* / *May* / *of*
29	*16* / *29* / *9*
.	*.* / *.* / *.*

Word	**Substitute**
W:Pierre	*S:Mr.*
W:Pierre	*S:Pierre*
W:Pierre	*S:John*
W:Vinken	*S:<unk>*
W:Vinken	*S:Beregovoy*
W:Vinken	*S:Cardin*
⋮	⋮
W:join	*S:head*
W:join	*S:join*
W:join	*S:leave*
W:the	*S:its*
W:the	*S:its*
W:the	*S:the*
⋮	⋮
W:director	*S:chairman*
W:director	*S:chairman*
W:director	*S:director*
⋮	⋮

Table 4.2: The table on the left shows three possible substitutes sampled with replacement for each position in an example sentence based on a 4-gram language model. The table on the right is the pairwise co-occurrence data fed to S-CODE derived from these samples. The prefixes "W:" and "S:" are used to distinguish target words and substitutes.

The next section describes CODE and S-CODE which take the pairwise co-occurrence data as input, and calculate the Euclidean embeddings of the words and their substitutes on an n-dimensional unit sphere.

4.2.2 *The CODE Model*

In this section, we review the unsupervised method that we use to model co-occurrence statistics: the Co-occurrence Data Embedding (CODE) (Globerson et al., 2007) method and its spherical extension (S-CODE) introduced by (Maron et al., 2010).

Let W and C be two categorical variables with finite cardinalities $|W|$ and $|C|$. We observe a set of pairs $\{w_i, c_i\}_{i=1}^n$ drawn IID from the joint distribution of W and C. The basic idea behind CODE and related methods is to represent (embed) each value of W and each value of C as points in a common Euclidean space $\mathbf{R}^d$ such that the values that frequently co-occur lie close to each other. There are several ways to formalize the relationship between the distances and co-occurrence statistics. In this thesis, we use the following:

$$p(w, c) = \frac{1}{Z}\bar{p}(w)\bar{p}(c)e^{-d^2_{w,c}} \tag{4.1}$$

where $d^2_{w,c}$ is the squared distance between the embeddings of w and c, $\bar{p}(w)$ and $\bar{p}(c)$ are empirical probabilities, and $Z = \sum_{w,c}\bar{p}(w)\bar{p}(c)e^{-d^2_{w,c}}$ is a normalization term. If we use the notation ϕ_w for the point corresponding to w, and ψ_c for the point corresponding to c, then $d^2_{w,c} = \|\phi_w - \psi_c\|^2$. The log-likelihood of a given embedding $\ell(\phi, \psi)$ can be expressed as:

$$\begin{aligned}
\ell(\phi, \psi) &= \sum_{w,c}\bar{p}(w, c)\log p(w, c) \\
&= \sum_{w,c}\bar{p}(w, c)(-\log Z + \log\bar{p}(w)\bar{p}(c) - d^2_{w,c}) \\
&= -\log Z + const - \sum_{w,c}\bar{p}(w, c)d^2_{w,c}
\end{aligned} \tag{4.2}$$

The likelihood is not convex in ϕ and ψ. We use gradient ascent to find an approximate solution for a set of ϕ_w, ψ_c that maximize the likelihood. The gradient of the $d^2_{w,c}$ term pulls neighbors closer in proportion to the empirical joint probability:

$$\frac{\partial}{\partial\phi_w}\sum_{w,c} -\bar{p}(w, c)d^2_{w,c} = \sum_{y} 2\bar{p}(w, c)(\psi_c - \phi_w) \tag{4.3}$$

The gradient of the Z term pushes neighbors apart in proportion to the estimated joint probability:

$$\frac{\partial}{\partial \phi_x}(-\log Z) = \sum_y 2p(w,c)(\phi_w - \psi_c) \tag{4.4}$$

Thus the net effect is to pull pairs together if their estimated probability is less than the empirical probability, and to push them apart otherwise. The gradients with respect to ψ_c behave similarly. S-CODE (Maron et al., 2010) additionally restricts all ϕ_w and ψ_c to lie on the unit sphere. With this restriction, Z stays around a fixed value during gradient ascent. This allows S-CODE to substitute an approximate constant $\tilde{Z}$ in gradient calculations for the real Z for computational efficiency. In our experiments, we used S-CODE with its sampling-based stochastic gradient ascent algorithm and smoothly decreasing learning rate.

S-CODE with More than Two Variables

In order to accommodate multiple feature types, the S-CODE model in the previous section needs to be extended to handle more than two variables. Section 6.2 of Globerson et. al (2007) suggest the following likelihood function:

$$\ell(\phi, \quad \psi^{(1)}, \ldots, \psi^{(K)}) = \bar{p}(w,c)\log p(w,c) + \sum_i^K \sum_{w,f^{(i)}} \bar{p}(w,f^{(i)})\log p(w,f^{(i)}) \tag{4.5}$$

where $\bar{p}(w,c)$ is the empirical joint distribution of context C with W, and $F^{(1)}, \ldots, F^{(K)}$ are K extra different variables whose empirical joint distributions with W, namely, $\bar{p}(w,f^{(1)}) \ldots \bar{p}(w,f^{(K)})$, are known. Eq. 4.5 then represents a set of CODE models $p(w,f^{(k)})$ in which each $F^{(k)}$ has an embedding $\psi_f^{(k)}$ but all models share the same embedding ϕ_w.

We adopt the above likelihood function. Now, let W represent a word, C represent a context (i.e., random substitute), and $F^{(1)}, \ldots, F^{(K)}$ stand for morphological and orthographic features of the word. So each co-occurrence is a (K+1)-tuple $(W, C, F^{(1)}, \ldots F^{(K)})$. With this setup, the training procedure needs to be modified a little: instead of sampling the

(word (w) , context (c)) pair, we sample the (word (w) , context (c) , features ($f_1, \ldots, f_K$)) tuple, and feed it to the gradient ascent algorithm. The gradient search algorithm updates the embeddings according to $p(w, c)$ and $p(w, f^{(i)})$, where $i = 1 \ldots k$. No updates are performed between c and $f^{(i)}$s, since they do not have any co-occurrence statistics, and w is the only shared variable.

4.2.3 Clustering Embeddings

At this stage, each word instance and its r substitutes, where r is the number of substitutes per instance sampled to represent their contexts, are mapped to real vector embeddings on an n-dimensional sphere[2]. We apply the instance-weighted k-means clustering algorithm to three different representations derived from these embeddings, each with its own advantages and disadvantages:

Word embeddings (W): We cluster the word embeddings. Each word has a single embedding, and gets assigned to a single cluster (which we will refer to as the one-tag-per-word assumption). Thus clustering words resulting from this representation employ the one-tag-per-word assumption from the beginning, and cannot handle ambiguous words with multiple parts of speech.

Average of substitute embeddings ($\bar{\text{S}}$): We construct a vector representation for each word-instance with the average of its r substitute embeddings, where r is the number of sampled substitutes per instance. First, we normalize these average vectors to Euclidean unit length then cluster the instances obtained from these averaged and normalized vectors, thus assigning each instance to a cluster. For example, the target word *W:Pierre* in Table 4.2 will be represented with the average of the embeddings of *S:Mr.*, *S:Pierre* and *S:John*, while

[2]In fact, many words that appear in the text also appear as substitutes, and thus have two embeddings.

all other instances of *Pierre* in different contexts are represented with the average of their own substitute embeddings.

The concatenation of the word embedding and the average of its substitute embeddings ($\mathbf{W} \oplus \bar{\mathbf{S}}$): We construct a vector representation for each word instance by concatenating its word embedding and the average of its r substitute embeddings. This results in a $2n$-dimensional vector representing an instance. Prior to clustering, we normalize these $2n$-dimensional vectors to Euclidean unit length. Clustering these $2n$-dimensional normalized vectors assigns each instance to a cluster. For instance, the target word *Pierre* in Table 4.2 will be represented by the concatenation of the embedding of *W:Pierre* and the average of the embeddings of *S:Mr.*, *S:Pierre* and *S:John*.

To sum up, the first setting applies the one-tag-per-word assumption from the beginning, and clusters words instead of word-instances. The second setting clusters word-contexts (as represented by the average of its substitutes), and is able to categorize the individual word-instances. However, it ignores the identity of the target word. The third setting also clusters word-instances, but incorporates the word identity by concatenating the word and the average of the corresponding substitute embeddings.

The clustering and the co-occurrence models require only the input word sequences to output the corresponding cluster-id sequences. The remaining models in this section require different levels of supervision to output the tag sequence of a given input sequence. The next section introduces a probabilistic voting model that requires the word-tag dictionary to be available in order for it to output the possible tag sequence of a given input.

4.3 Model 3: Probabilistic Voting Model

In this section, we model the tag distribution in a given context, and assign the most likely tag to the context instead of the target word-instance. The distribution of t in c can be

defined in terms of the substitutes and their possible tags in a word-tag dictionary or distribution. The model is defined as

$$\underset{t \in T}{\operatorname{argmax}} \Pr(t|c) = \sum_{s \in S} \Pr(t|s,c) \Pr(s|c) \tag{4.6}$$

$$= \sum_{s \in S} \Pr(t|s) \Pr(s|c) \tag{4.7}$$

where s is a substitute word in the substitute distribution S of the context c. $\Pr(s|c)$ is the probability of observing s in c which is the entry corresponding to s in S. Eq.(4.6) is simplified to Eq.(4.7) by making the assumption that the tag of a substitute word and the context are independent of each other for a given substitute word s.

The model distributes $\Pr(s|c)$ among the possible tags of s in a word-tag dictionary, taking into consideration $\Pr(t|s)$ which is the probability of observing s with tag t. The estimation of $\Pr(t|s)$ depends on the level of supervision. For example, if the word-tag dictionary is available, then $\Pr(t|s)$ could be uniform over the possible tags of s, or if the word-tag distribution is available, then $\Pr(t|s)$ would consist of the observation frequencies of s with t. One could only have the substitutes but not the corresponding $\Pr(s|c)$ or simply wants to assign equal weights to each substitute instead of weighting them with $\Pr(s|c)$. In this situation $\Pr(s|c)$ could be uniform over the possible substitutes in a given context.

As is the case with the previous models, this model also assumes that the tags of two consecutive words are independent of each other for any fixed contexts of the words. Model 3 also assumes that tag distribution is independent of the target word given the context of the target word. The models in the following section relaxes this assumption by integrating the substitute distributions into the HMM-based models.

4.4 Model 4: Constraining HMM-Based Models

None of the previous models captures the relationship between consecutive tags because of the independence assumption of consecutive tags in fixed contexts. However, one can take advantage of consecutive tags to constrain the output sequence. For example, in a POS disambiguation task, determiners (DT) are usually followed by a noun (NN). In this section, we introduce two different ways of incorporating substitute words into the HMM-based probabilistic models in which consecutive tag sequences depend on each other while consecutive words are independent of each other for any given tags.

The HMM-based probabilistic models have been used to solve NLP tagging tasks (Merialdo, 1994; Goldwater and Griffiths, 2007a) with different levels of supervision. The prototypical n-tag HMM model maximizes the likelihood of the corpus $w_1 \dots w_N$, expressed as

$$P(w_1|t_1) \prod_{i=2}^{N} P(w_i|t_i) P(t_i|t_{i-1}, \dots, t_{i-n+1}) \tag{4.8}$$

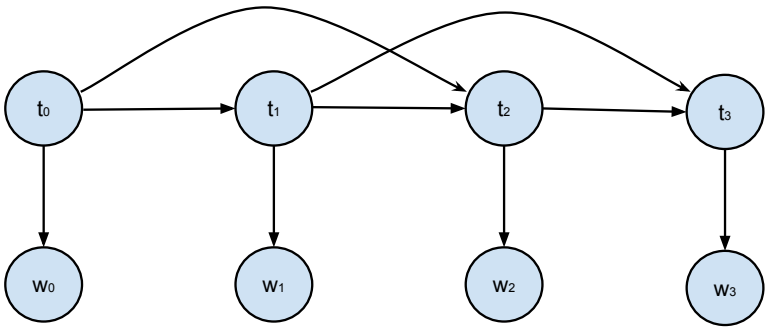


Figure 4.2: Graphical structure of a standard second-order HMM tagger on an example 4-word sequence.

where w_i are the word tokens, and t_i are their (hidden) tags. The HMM-based approaches generally first learn the parameters relating the hidden structure to the observed sequence of variables $\Pr(w_i|t_i)$, and then the new hidden structure from the previous $n-1$ hidden

structures $\Pr(t_i|t_{i-1}, \ldots, t_{i-n+1})$. Finally, they identify the most probable values of the hidden structure for a given observed sequence using the Viterbi search algorithm (Viterbi, 1967). Figure 4.2 illustrates a standard bi-gram HMM model in which the hidden units that generate tags are represented by h_0^4, and the observed word sequence is represented by w_0^4.

The HMM-based models differ in the way the model parameters are estimated. For example, the HMM-EM models that are trained with expectation maximization estimate the parameters by using the maximum likelihood estimation (MLE), maximum a posteriori (MAP)-based models define a prior distribution over the parameters and find the parameter values to maximize the posterior distribution given data, and Bayesian models integrate over the posterior of the parameters to incorporate all possible parameter settings into the estimation process.

In this section, we focus on HMM-EM since it is the simplest HMM-based approach, and has a traditional place in the NLP literature of unsupervised tagging (Merialdo, 1994). We assume that the word-tag distribution is available. However, models in this chapter are not limited to HMM-EM, and can be extended to HMM-MAP and HMM-Bayesian.

In POS disambiguation, the observed variables are a sequence of words $(w_0, w_1, \ldots, w_{n-1}, w_n)$, and the hidden variables are a sequence of POS tags $(t_0, t_1, \ldots, t_{n-1}, t_n)$. The HMM parameters θ can be estimated by using Baum-Welch EM algorithm on an unlabeled training corpus D (Baum, 1972). The tag sequence that maximizes $\Pr(t|w_0, \ldots, w_n, \hat{\theta})$ can be identified by the Viterbi search algorithm.

Johnson (2007b) showed that HMM-EM has a tendency of assigning equal number of words to each hidden state, thus resulting in poor tagging performance on tasks with skewed word-tag distributions. Mitzenmacher (2004) argued that a Bayesian method with sparse priors over the tags may perform better than an HMM estimated with EM on a problem with skewed word-tag distributions.

Ravi and Knight (2009a) defined the *observed grammar size* to be the number of distinct tag bi-grams observed in the output sequences of a given input corpus. They showed that

the observed grammar size of the HMM-EM model in a POS disambiguation task is larger than the actual observed grammar size[3]. To fix this, they constrain the number of nonzero transition probabilities ($\Pr(t_i|t_{i-1})$) of a bi-gram HMM model using integer programming (IP).

To take advantage of substitute distributions in an HMM setting, we consider generating artificial sentences where one of the words of the original sentence is replaced with a likely substitute. We assume the hidden tag sequence that generates the original sentence should also generate the artificial sentences. Motivated by this idea, we propose two methods that incorporate substitute words to improve the HMM-EM performance. The first method improves the performance of HMM-EM by reducing the noise and rare tags in the word-tag dictionary. The second one does not modify the word-tag dictionary or the EM training phase, but constrains the Viterbi search algorithm by providing artificially created new sentences that are derived from the target sentence. As a result, the first method constrains the HMM-EM to learn smaller models in terms of the grammar size, while the second one constrains the search space of Viterbi even when the HMM-EM grammar size is larger than the actual grammar size.

Currency gyrations can **whipsaw(VB/NN)** the funds .
Currency gyrations can **withdraw(VB)** the funds .
Currency gyrations can **restore(VB)** the funds .
Currency gyrations can **modify(VB)** the funds .
Currency gyrations can **justify(VB)** the funds .
Currency gyrations can **regulate(VB)** the funds .

Table 4.3: Sample artificial sentences generated for a test sentence from the Penn Treebank.

[3]The actual observed grammar size is calculated using an annotated PTB corpus. Annotations are only used for the model comparison, and are discarded during the learning process.

Table 4.3 presents an example from a POS tagging task in which the highly likely substitutes of the target word **whipsaw** are listed for a given sentence from the Penn Treebank (PTB). In this example, each substitute is an unambiguous verb (*VB*), and hence the correct tag of **whipsaw** can be correctly disambiguated by the substitutes. The target word and its substitutes are both syntactically and semantically related to each other. Thus, our approach is not limited to the POS tagging task, but can be extended to other NLP tagging tasks such as WSD.

Section 4.4.1 presents the steps of the word-tag dictionary reduction method, and Section 4.4.2 presents the data-enhanced HMM-EM algorithm.

4.4.1 Method 1: Dictionary Reduction

This method, initial to the HMM-EM training, reduces the word-tag dictionary size by deleting the unlikely tags of each target word from the word-tag dictionary. To determine the unlikely tags of a word, it estimates the tag distribution of the word by averaging the tag-context distributions of its instances. The tag distribution of each instance is calculated by using Model 3. For example, the word *a* has 7 possible tags in its word-tag dictionary and two of its possible tags (i.e, *SYM* and *LS*)[4] are observed with very low probabilities in the tag distribution of the word *of*. Thus they are deleted from the word-tag dictionary. Instead of removing the unlikely tags, this method can be also used for estimating the word-tag distributions when the only available information is the word-tag dictionary.

Another interpretation of dictionary reduction is that votes of the substitute words are used to reduce the possible number of tags per word in the word-tag dictionary. This expedites the training phase of the HMM-EM algorithm by reducing the number of non-zero parameters. For all instances of the target word, the procedure counts the votes of possible tags and removes the tags with low votes from the word-tag dictionary entry of the target

[4] *SYM* and *LS* represent symbols and list item markers.

word.

4.4.2 Method 2: Data Enhanced Viterbi Search Algorithm

In this method, the Viterbi search algorithm constructs the most likely tag sequence of a given observed word sequence after the training phase of HMM. Because of the existence of ambiguous words, more than one tag sequence is possible for any given word sequence. To constrain all possible tag sequences of a given word sequence, we construct artificial sentences such that each of these sentences differ only in one word from the original sentences. We do this with the expectation that the tag sequence will be the same for both the original and the artificial sentence.

Figure 4.3 presents the difference between the standard HMM-EM and data enhanced HMM-EM. To keep things simple, we present the derivation of the Viterbi search on a first order HMM in which each hidden state is only conditioned by its preceding hidden state. The derivation for the second order HMM is similar.

Viterbi Search Algorithm

Let a_i denote the i^{th} artificial sequence generated from the original sequence $w_{00} \ldots w_{0n}$, where w_{ij} denotes the j^{th} word of a_i, h_j denotes the j^{th} hidden state, t_k denotes the k^{th} tag in the set of all possible tags T, and $\delta_j(t_k)$ denotes the probability of the best tag sequence up to the j^{th} word that is tagged with t_k. The number of artificial sequences is r, and the number of words in the original sequence is n. Using these symbols, the probability of the best path up to j^{th} word with tag t_k in a standard HMM can be written as

$$\delta_j(t_k) \quad = \quad max_{t_1,..,t_{j-1}} \Pr(h_1, .., h_{j-1}, h_j = t_j, w_{01}, .., w_{0j}|\theta) \tag{4.9}$$

Equation 4.9 can be solved recursively by using following equations:

$$\delta_1(t_k) = \Pr(h_1 = t_k|\theta)\Pr(w_{01}|h_1 = t_k, \theta) \quad (4.10)$$

$$\delta_{j+1}(t_k) = max_{t\in T}\delta_j(t)\Pr(h_{j+1} = t_k|h_j = t, \theta)\Pr(w_{0j+1}|h_{j+1} = t_k, \theta) \quad (4.11)$$

$$\hat{\delta} = max_{t\in T}\delta_n(t) \quad (4.12)$$

Equation 4.10 gives the probability of tagging the first word w_{01} with tag t_j, and Equation 4.11 defines the recursive relation between consecutive δs. Equation 4.12 gives the probability of the best path. The best path can be constructed during the recursive calculation of δs. We do not need to demonstrate the path construction part of the Viterbi search, since the data-enhanced HMM uses the same formulation.

In the case of data-enhanced HMM-EM, the optimum tag sequence $\mathbf{t_{opt}}$ of the original sequence, together with its artificial sentence set $a_1 \ldots a_r \in A_s$, is given by Equation 4.13. This equation can be solved by modifying Equations 4.10 and 4.12, as follows:

$$\mathbf{t_{opt}} = argmax_{t_0\ldots t_n}\Pr(t_0 \ldots t_n|\mathbf{s}, A_s, \hat{\theta}) \quad (4.13)$$

$$\delta_1(t_k) = \Pr(h_1 = t_k|\theta)\prod_{i=1}^{r}\Pr(w_{i1}|h_1 = t_k, \theta) \quad (4.14)$$

$$\delta_{j+1}(t_k) = max_{t\in T}\delta_j(t)\Pr(h_{j+1} = t_k|h_j = t, \theta)\prod_{i=1}^{r}\Pr(w_{ij+1}|h_{j+1} = t_k, \theta) \quad (4.15)$$

Equation 4.14 and 4.15 do the same calculation, except that they incorporate the replacement sentences into the model as observed variables. In the case of HMM-EM, the Viterbi search algorithm is applied jointly to the original sequence and its artificial sentences so as to get the most probable tag sequence for all.

In this section, we use the HMM-based model to find out the most likely tag sequence of a given input sequence when a word-tag dictionary is available. An n^{th} order HMM-EM models assumes that each tag depends on the previous $n-1$ tags and word-instances are independent of each other given their corresponding tags. Thus the models in this section explicitly incorporate into the learning process both the word identity and the dependencies

between the tags. The next model introduces a noisy channel model that requires the word-tag distribution in order to perform the tagging task.

4.5 Model 5: Noisy Channel Model

In this section, we introduce a noisy channel model that disambiguates the most likely tag sequence of a given word sequence. Unlike the previous models, this model requires the availability of the word-tag distribution. The main limitation of the model is that the tags of consecutive words are independent of each other, given the corresponding contexts.

The noisy channel model has been the foundation of standard models in speech recognition (Bahl et al., 1983) and machine translation (Brown et al., 1990). The noisy channel model can be used whenever a received signal does not uniquely identify the message being sent. Bayes' Law can be used to interpret the ambiguous signal, and identify the most probable intended message. In tagging tasks, we model each context as a distinct channel, where the intended message is a tag T, and the received signal is an ambiguous word W. The model assumes the independence of consecutive messages. In this section, we will describe how to model a given context C as a noisy channel, and, in particular, how to estimate the context-specific tag distribution by using the word-tag distribution.

Equation 4.16 expresses the probability of a tag T of word W in a given context C. This is the well-known Bayes' formula with an extra $\Pr(.|C)$ in each term to indicate the dependence on the context.

$$\Pr(T|W,C) = \frac{\Pr(W|T,C)\Pr(T|C)}{\Pr(W|C)} \tag{4.16}$$

To perform disambiguation, we need to find the tag T that maximizes the probability $\Pr(T|W,C)$. This is equivalent to the maximization of the product $\Pr(W|T,C)\Pr(T|C)$, because the denominator $\Pr(W|C)$ does not depend on T. To perform the maximization, the two distributions $\Pr(W|T,C)$ and $\Pr(T|C)$ need to be estimated for each context C.

How to estimate $\Pr(T|C)$, the distribution of word tags that can be expressed in the given context? The only supervision is the word-tag distribution, since we do not have access to any tagged data, and consequently we do not know what tags are likely to be expressed in any given context. Therefore, it is not possible to estimate $\Pr(T|C)$ directly.

What we do have is the word frequencies for each tag $\Pr(W|T)$, and the word frequencies for the given context $\Pr(W|C)$. We use the word-tag distribution to estimate $\Pr(W|T)$, and a statistical language model to estimate $\Pr(W|C)$, as detailed in Section 2. We make the independence assumption $\Pr(W|T,C) = \Pr(W|T)$, i.e. the distribution of words used to express a particular tag is the same for all contexts. Finally, the relationship between the three distributions, $\Pr(T|C)$, $\Pr(W|T,C)$, and $\Pr(W|C)$ is given by the total probability theorem:

$$\Pr(W|C) = \sum_{T} \Pr(T|C) \Pr(W|T,C) \tag{4.17}$$

We can now solve for $\Pr(T|C)$ using linear algebra. Let $\mathbf{WT}$ be a matrix, $\vec{t}$ and $\vec{w}$ two vectors such that:

$$\begin{aligned} \mathbf{WT}_{ij} &= \Pr(W = i|T = j) \\ \vec{t}_j &= \Pr(T = j|C = k) \\ \vec{w}_i &= \Pr(W = i|C = k) \end{aligned} \tag{4.18}$$

Using this new form, we can see that Equation 4.17 is equivalent to the linear equation $\vec{w} = \mathbf{WT} \times \vec{t}$, and $\vec{t}$ can be solved using a linear solver. Typically $\mathbf{WT}$ is a tall matrix, and the system has no exact solutions. One can use a linear constraint solver (e.g., interior point algorithm) or a pseudo-inverse algorithms (e.g., the Moore-Penrose pseudo-inverse) to compute an exact or approximate solution to:

$$\vec{t} = \mathbf{WT}^{+} \times \vec{w} \tag{4.19}$$

Another interpretation of this model is that it performs a voting process similar to the probabilistic voting in Section 4.3 to estimate the tag distribution of a given context. To

see this, consider the following example, let each word be unambiguous, and therefore assume that **WT** equals the identity matrix $\mathbf{I}_{|W|}$, of dimensionality $|W| \times |W|$. Under this assumption, the inverse of **WS** would also be equal to $\mathbf{I}_{|W|}$, and $\Pr(T|C)$ would be equal to $\Pr(W|C)$. This implies that the most likely tag in C is actually the tag of the unambiguous word with the highest $\Pr(W|C)$. Thus, the noisy channel model estimates $\Pr(T|C)$ by distributing $\Pr(W|C)$ to the tags with respect to the inverse of **WT**.

4.6 Conclusion

The first model ignores the identity of the word and clusters the substitute distributions without requiring any level of supervision. The second one models the co-occurrences of words with their substitute words, therefore, incorporates the word identity and the context information at the same time. To construct the co-occurrence representation this model discretizes the substitute distributions. The third model performs probabilistic voting and estimates the distribution of tags in a given context. Unlike the first and second model this model requires the availability of *word-tag dictionary* in which all possible tags of a given word is available. The fourth model proposes two extensions to the standard HMM-based tagging models in which both the word identity and the dependence between the consecutive tags are concerned. The last one introduces a generative probabilistic model, the noisy channel model, for the word sense disambiguation task in which the word-tag frequencies are available. In this model, each context C is modeled as a distinct channel through which the speaker intends to transmit a particular meaning S using a possibly ambiguous word W. To reconstruct the intended meaning the hearer uses the distribution of possible meanings in the given context $\Pr(S|C)$ and possible words that can express each meaning $\Pr(W|S)$. Section 5 applies Model 4 to POS disambiguation, Section 6 applies Model 3 to morphological disambiguation, Section 7 applies Model 5 to word-sense disambiguation, and, finally, Section 8 applies Model 1 and 2 on POS induction.

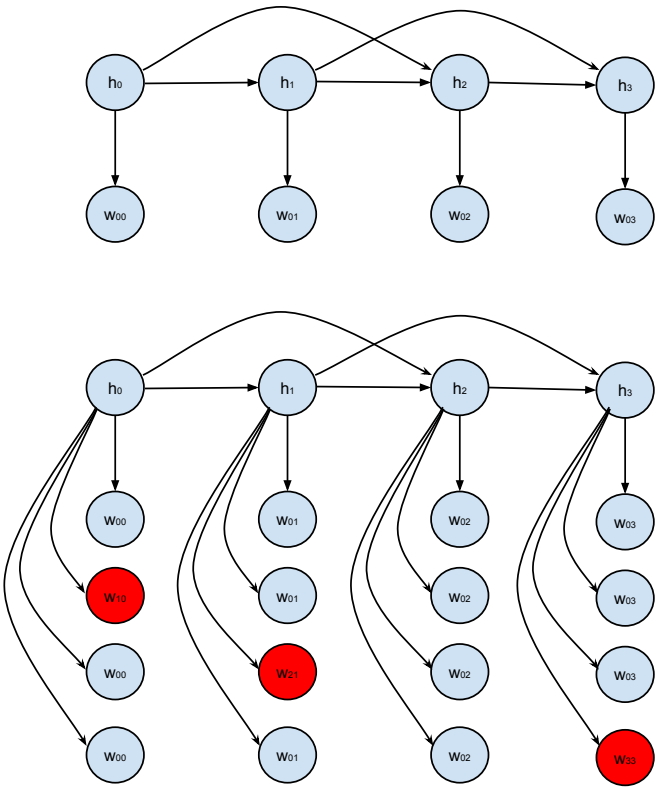


Figure 4.3: Graphical structure of a standard second-order HMM tagger (top) and data-enhanced HMM tagger (bottom) on a 4 word sentence. Red circles represent the substitute in an artificial sentence while the blue ones represent the original words.

Chapter 5

PART OF SPEECH DISAMBIGUATION

In this section, we apply the algorithms defined in Section 4.4 to the weakly-supervised part-of-speech (POS) disambiguation of English.[1] This task consists of predicting the correct POS tag of a word in a specified context, given an unlabeled corpus and a dictionary with possible word-POS-tag pairs. The performance of an unsupervised POS tagging system depends highly on the quality of the word-tag dictionary (Banko and Moore, 2004) or the constraints on the learning models (Johnson, 2007b).

The rest of this chapter is organized as follows: Section 5.1 reviews the related work on POS tagging. Section 5.2 defines the experimental settings that are used in this chapter. Section 5.3 defines supervised and unsupervised baselines. Section 5.4 presents the results of experiments with the dictionary reduction method (see Section 4.4.1) to improve the performance of the methods involving expectation maximization (EM). Section 5.5 presents the results of experiments with the data-enhanced Viterbi decoding method (see Section 4.4.2). Section 5.5.3 shows an application of the data-enhanced Viterbi decoding on out-of-vocabulary (OOV) words. Finally, Section 5.7 defines the road map for future work. This section also introduces a new method that incorporates substitute words in the disambiguation process during the estimation of the probabilistic model parameters, rather than just using them for Viterbi decoding or dictionary reduction.

---

[1] Throughout this chapter POS tagging is used interchangeably with POS disambiguation.

5.1 Related Work

Probabilistic models such as the hidden Markov model trained by expectation maximization (HMM-EM), maximum a posteriori (MAP) estimation, and Bayesian methods have been used to solve the unsupervised POS tagging problem (Merialdo, 1994; Goldwater and Griffiths, 2007a). All of these approaches first learn the parameters relating the hidden structure to the observed sequence of variables and then identify the most probable values of the hidden structure for a given observed sequence. They differ in the way they estimate the model parameters. HMM-EM estimates model parameters by using the maximum likelihood estimation (MLE), MAP defines a prior distribution over parameters and finds the parameter values that maximize the posterior distribution for a given data, and Bayesian methods integrate over the posterior of the parameters to incorporate all possible parameter settings into the estimation process. Some baseline results and performance reports from the literature are presented in Table 5.1.

Johnson (2007b) criticizes the standard HMM-EM approaches for their poor performance on unsupervised POS tagging and their tendency to assign equal number of words to each hidden state. Mintzenmacher (2004) further claims that words have skewed POS tag distributions, and a Bayesian method with sparse priors over the POS tags may perform better than HMM-EM. Goldwater and Griffiths (2007a) use a fully Bayesian HMM model that averages over all possible parameter values. Their model achieves 86.8% tagging accuracy with sparse POS priors, outperforming the standard second-order HMM-EM (3-gram tag model) with 74.50% accuracy on a 24K (PTB24K) word subset of the Penn Treebank corpus. Taking a different approach, Smith and Eisner (2005) use the conditional random fields estimated using contrastive estimation, which achieves 88.6% accuracy on the same PTB24K corpus, and thus outperforms the HMM-EM and Bayesian methods.

Despite the fact that HMM-EM has a poor reputation in the POS literature, Goldberg et al. (2008) have shown that with good initialization in conjunction with certain language-

Accuracy	System
64.2	Random baseline
74.4	Second-order HMM
81.8	Most frequent tag baseline
82.0	First-order HMM
86.8	Fully Bayesian approach with sparse priors (Goldwater and Griffiths, 2007a)
88.6	CRF/CE (Smith and Eisner, 2005)
91.4	EM-HMM with language specific information, good initialization and manual adjustments to standard dictionary (Goldberg et al., 2008)
91.8	Minimized models for EM-HMM with 100 random restarts (Ravi and Knight, 2009b).

Table 5.1: Tagging accuracy on a PTB24K-word corpus. All the systems—except (Goldwater and Griffiths, 2007a)—use the same 45-tag dictionary that is constructed from the Penn Treebank.

specific features and language-dependent constraints, HMM-EM achieves 91.4% accuracy. Aside from the language-specific information and the good initialization, they also employ some manual tuning to reduce the noise in the word-tag dictionary.

Ravi and Knight (2009b) focus on POS tag collection to find the smallest POS model that explains the data. They apply integer programming to construct a minimal bi-gram POS tag set, and use this set to constrain the training phase of the EM algorithm. The model trained by EM is used to reduce the dictionary, and these steps are iteratively repeated until no further improvement is observed. Their model achieves 91.6% accuracy on the PTB24K word corpus. (The accuracy increases to 91.8% with 100 random starts.) The main advantage of this model is the restriction of the tag set so that rare POS tags and the noise in the corpus do not influence the estimation process.

5.2 Experimental Settings

In this section, we present a number of experiments measuring the performance of *dictionary reduction* and *data-enhanced Viterbi decoding*, which have been defined in Section 4.4. As the models in this section are trained[2] and tested on the same unlabeled data, no out-of-vocabulary words are involved.

5.2.1 Language Model

To obtain accurate domain independent probability estimates, we used the Web 1T data set (Brants and Franz, 2006b) that contains the counts of word sequences of length up to five in a 10^{12}-word corpus derived from publicly accessible Web pages. The SRILM toolkit is used to train a 5-gram language model (Stolcke, 2002a). The language model parameters are optimized by using a randomly selected PTB24K word corpus from Penn Treebank. In order to efficiently apply the language model to a given test corpus, the vocabulary is limited to the words seen in the test corpus.

5.2.2 Dataset

In the rest of this chapter, we limit ourselves to the 4 corpora consisting of the first 12K (PTB12K), 24K (PTB24K), 48K (PTB48K), and 96K (PTB96K) words of the 1M-word Wall Street Journal Section of the Penn Treebank (PTB) corpus. To be consistent with the POS literature, the tag dictionary is constructed by listing all of the observed tags for each word in PTB. Nearly 55% of the words in Penn Treebank corpus are ambiguous, and the average number of tags is 2.3.

Table 5.2 shows the POS speech groups and their distributions in the PTB24K word corpus. We report the model accuracy on several POS groups. Our motivation is to deter-

[2]The GMTK tool is used to train the HMM-EM model on an unlabeled corpus (Bilmes and Zweig, 2002).

Groups	Member POS tags	Count	%
Noun	NN, NNP, NNS, NNPS	7511	31.30
Verb	VBD, VB, VBZ, VBN, VBG, VBP	3285	13.69
Adj	JJ, JJR, JJS	1718	7.16
Adv	RB, RBR	742	3.09
Pronoun	CD, PRP, PRP$	1397	5.82
Content	Noun, Verb, Adj, Adv, Pronoun	14653	61.05
Function	Other	9347	38.95
Total	All 45 POS tags	24K	100.00

Table 5.2: Group names, members, number, and percentage of words according to their gold POS tags.

mine the accuracy of the HMM-EM model on the subgroups before and after implementing the dictionary reduction procedure.

5.3 Baseline

Table 5.3 presents some standard baselines for comparison. We define a random and a supervised baseline on the PTB24K corpus. The random baseline is calculated by randomly picking one of the tags of each word. This baseline also represents the amount of ambiguity in the corpus. The supervised baseline consists simply of the most frequent POS tag of each word, using the 1M-word Penn Treebank corpus as the training corpus. (The first 24K words of the corpus are not included in the 1M-word training corpus.) If the target word does not exist in the training set, then the supervised baseline randomly picks one of the possible tags of the missing word.

The first- and second-order HMMs can be treated as the unsupervised baselines. These

	Noun	Verb	Adj	Adv	Pronoun	Content	Function	Total(%)
Random Baseline	76.98	53.87	68.46	72.98	87.64	71.59	52.64	64.21
3-gram HMM	77.43	68.16	78.06	73.32	94.85	76.88	70.45	74.38
Supervised Baseline	85.29	59.24	64.52	59.03	84.57	75.62	91.37	81.75
2-gram HMM	92.22	83.84	85.22	83.96	95.56	89.42	70.49	82.05

Table 5.3: Percentages of words tagged correctly by different models using the standard dictionary.

unsupervised baselines are calculated by training the uniformly initialized first- and second-order HMMs on the target corpus without any smoothing. All the initial parameters of HMM-EM are uniformly initialized so as to observe only the effect of the artificial sentences on the performance of HMM-EM.

The success of the supervised baseline on the *Noun*, *Pronoun*, and *Function word* groups indicates that tag distributions of the words in these groups are highly skewed towards one of the available tags. Compared to the above groups, the supervised baseline performs poorly on *Verb*, *Adj*, and *Adv*. This is a result of the less skewed POS tag behavior of these tags.

The second-order HMM is commonly taken to be the baseline in the POS tagging literature. However, as is clear from Table 5.3, this model can be outperformed by an unsupervised first-order HMM or a simple supervised baseline like majority voting. It is worth noting that although the first-order HMM and the supervised baseline have similar overall accuracies, the first-order HMM is better on the content words while the supervised baseline is better on *Function words*. This is to be expected, since EM tends to assign words uniformly to the available POS tags. Thus, EM cannot capture the skewed behavior of *Function words*. Moreover, the amount of skewness affects the accuracy of EM in such

a manner that the performance gain over the supervised baseline on *Verb*, *Adj*, and *Adv* is around 20–25% while the performance gain on *Noun* and *Pronoun* is around 6–7%.

5.4 Experiment: Dictionary Reduction

EM tends to assign equal number of words to each POS tag because it cannot capture the sparse structure of word distributions. If the word-tag dictionary is noisy, then a large number of function words are tagged with very rare POS tags. Table 5.4 illustrates this abuse of rare tags, also seen in (Ravi and Knight, 2009b). In this set of experiments we assume that we only have the top k substitutes and have no information regarding their probabilities in the corresponding context.

To remove rare tags from the word-tag dictionary, we apply the following steps:

Algorithm

1. Choose the top k most likely unambiguous substitutes in the target word context.

2. Substitutes must be observed in the word-tag dictionary.

3. Count the tags of the top k substitutes for all target word instances.

4. Remove the tags that are not observed as the tag of substitutes in any of the target word instances.

The first rule decreases the level of ambiguity by selecting only the unambiguous substitutes from the word-tag dictionary. The second rule makes sure that the unambiguous substitutes do occur in the word-tag dictionary. The counts of substitute POS tags and the deleted rare POS tags for two erroneous function words are shown in Table 5.4. The experiments in this section focus on: (1) the analysis of dictionary reduction and (2) the number of top substitutes used for each ambiguous word.

Word	Tag dictionary	Gold tagging	EM tagging	Substitutes POS counts
of	{RB, RP, IN}	IN(632) RP(0) RB(0)	IN(0) RP(632) RB(0)	IN(2377) **RP(0)** RB(850)
a	{LS, SYM, NNP, FW, JJ, IN, DT}	DT(458) IN(1) JJ(2) SYM(1) LS(0)	DT(0) IN(0) JJ(0) SYM(258) LS(230)	DT(513) IN(317) JJ(1329) **SYM(0)** **LS(0)**

Table 5.4: Deleted POS tags of the given words are shown in bold.

The results obtained with the dictionary that is reduced by using the top 5 likely unambiguous substitutes are presented in Table 5.5. Note that with the reduced dictionary the uniformly initialized first-order HMM-EM achieves 91.85% accuracy. We execute 100 random restarts of the EM algorithm and select the model with the highest corpus likelihood. Our model achieves 92.25% accuracy—so far the highest accuracy reported for the PTB24K corpus.

As Table 5.5 shows, the effect of dictionary reduction is more noticeable on the content words than on the function words. This happens mainly because function words are frequently tagged with one of their tags. The same explanation can also be given for the high accuracy of the majority voting-based supervised baseline on function words.

The reduced dictionary (RD) removes the problematic rare POS tags of the words, thus improving significantly the accuracy on the content and function words as compared to the HMM models trained on the original dictionary.

POS Groups	2-gram HMM Accuracy (%)	2-gram HMM RD Accuracy (%)
Noun	92.22	94.01
Verb	83.84	84.90
Adj	85.22	89.52
Adv	83.96	85.18
Pronoun	95.56	95.92
Content	89.42	91.18
Function	70.49	92.92
All	82.05	91.85

Table 5.5: Percentages of correctly tagged words by different models with the modified dictionary. The dictionary size is reduced by using the top 5 substitutes of each target word.

5.4.1 *Number of Substitutes*

In this set of experiments, we try different numbers of artificial substitute words for each ambiguous word in a given sentence. We run our method on the PTB24K corpus with 1, 5, 10, 25, and 50 substitutes per ambiguous word. Table 5.6 displays the results. The performance of our method is dependent on the number of substitutes, the highest score being achieved when 5 substitutes are used. Incorporating the probability of the substitutes into the model rather than using a hard cutoff might offer a better solution.

Number of Substitutes	2-gram HMM RD Accuracy (%)
none	82.05
1	89.65
5	**91.85**
10	90.09
25	89.97
50	89.83

Table 5.6: Percentages of correctly tagged words by the models trained on the PTB24K corpus with different reduced dictionaries. The dictionary size is reduced by using different number substitutes.

5.4.2 *Amount of Data*

In this set of experiments we doubled the size of the data and trained HMM-EM models on a corpus that consists of the first 48K words of the Penn Treebank corpus. Our aim is to observe the effect of more data on our dictionary reduction procedure. Using the 5 replacements of each ambiguous word we reduce the dictionary and train a new HMM-EM

model using this dictionary. The additional data together with 100 random starts increases the model accuracy to 92.47% on the 48K corpus.

POS Groups	3-gram HMM RD Accuracy (%)	2-gram HMM RD Accuracy (%)
Noun	89.45	93.47
Verb	85.56	88.99
Adj	86.02	87.53
Adv	94.44	95.92
Pronoun	94.08	94.04
Content	88.91	91.97
Function	92.44	92.26
All	90.31	92.09

Table 5.7: Percentages of the correctly tagged words by the first and second order HMM-EM model trained on the 48K corpus with reduced dictionary. The dictionary size is reduced by using the top 5 replacements of each target word.

As we mentioned before, when the model is trained using the original dictionary, the performance gap between the first order HMM the second order HMM is around 8% as presented in Table 5.3. On the other hand, when we use the reduced dictionary together with more data the accuracy gap between the second order and the first order HMM-EM becomes less than 2% as shown in Table 5.7. This confirms the hypothesis that the low performance of the second order HMM is due to data sparsity in the 24K-word dataset, and better results may be achieved with the second order HMM in larger datasets.

5.4.3 17-Tag Set

To observe the effect our method on a model with coarse grained dictionary, we collapsed the 45–tagset treebank dictionary to a 17–tag set coarse dictionary (Smith and Eisner, 2005). The POS literature after the work of Smith and Eisner follows this tradition and also tests the models on this 17–tagset. Table 5.8 summarizes the previously reported results on coarse grained POS tagging. Our system achieves 92.9% accuracy where the oracle accuracy of 24K dataset with the reduced 17–tagset dictionary is 98.3% and the state-of-the-art system IP+EM scores 96.8%.

Model	Accuracy	Data Size
BHMM	87.3	24K
CE+spl	88.7	24K
RD	**92.9**	24K
LDA+AC	93.4	1M
InitEM-HMM	93.8	1M
IP+EM	96.8	24K

Table 5.8: Performance of different systems using the coarse grained dictionary.

The IP+EM system constructs a model that describes the data by using minimum number of bi-gram POS tags then uses this model to reduce the dictionary size (Ravi and Knight, 2009b). InitEM-HMM uses the language specific information together with good initialization and it achieves 93.8% accuracy on the 1M word treebank corpus. LDA+AC semi-supervised Bayesian model with strong ambiguity class component given the morphological features of words and scores 93.4% on PTB (Toutanova and Johnson, 2007). CE+spl is HMM model estimated by contrastive estimation method and achieves 88.7% accuracy (Smith and Eisner, 2005). Finally, BHMM is a fully Bayesian approach that uses sparse

POS priors and scores 87.3% (Goldwater and Griffiths, 2007a).

5.5 Experiment: Data-Enhanced Viterbi Search Algorithm

This method uses the statistical language model for generating artificial sentences to improve the disambiguation of ambiguous words. The improvement results from the amended Viterbi decoding algorithm described in Section 4.4.2. The steps of the data-enhanced HMM-EM can be summarized as follows:

Algorithm

1. Select the substitute words of the ambiguous words in $w_0 \dots w_n$, using the Criteria 1, 2, or 3.

2. Construct r artificial sentences by replacing w_i with the selected substitutes..

3. Finally, apply the Viterbi search algorithm to jointly predict the tag sequence of the original and the artificial sequences.

The experiments in this section focus on: (1) substitute selection criteria and (2) the number of the substitutes used for each ambiguous word.

The selected substitute words may not always have a common tag. If so, then the probability of the optimum tag sequence will be zero whenever a zero clique occurs, and the Viterbi search algorithm will not be able to find an optimum tag sequence. To prevent zero cliques in data-enhanced HMM-EM, we perform smoothing to assign non-zero probabilities to every entry of the $\Pr(w|t)$ matrix. To solve this problem, we assign a very small fraction ϵ of the probability to the zero entries of the $\Pr(w|t)$ matrix. The remaining part of the probability, namely $(1-\epsilon)$, is distributed among the non-zero entries of $\Pr(w|t)$ matrix in proportion to their probabilities. Assigning negligible probabilities to the zero entries of

the word given in the tag matrix is a smoothing operation that prevents the occurrence of zero cliques in the Viterbi algorithm.

5.5.1 Substitute Selection Criteria

The main idea of our model is that similar sentences should have the same tag sequence. Our method derives similar sentences from the target sentence by replacing each word with its likely substitutes. In order to observe the effect of selected substitutes, we define three different selection criteria: (1) select the words with at least one common tag, (2) select the words whose tag set is a subset of the target word tag set, and (3) select only the unambiguous words that have a common tag with the target word. The number of substitute words is fixed to 25 for the experiments in this section.

	12K	24K	48K	96K
Criterion 1	74.61	77.42	79.72	80.68
Criterion 2	75.17	77.42	81.02	80.99
Criterion 3	76.38	80.00	81.55	81.61
Unsup. Baseline	72.18	74.38	77.43	78.75
Random Baseline	66.57	67.03	67.07	66.99

Table 5.9: Results of our approach on different corpora with different settings. All the results are statistically significant and the 25 best substitute words for each ambiguous word are used in all the experiments.

As Table 5.9 shows, Criterion 3 (i.e. selecting the unambiguous words in the given context) outperforms the other criteria on all of the corpora. The main problem with Criterion 1 is that it accepts any word as long as that word has a common tag with the target word, and some of these words have more tags than the target ambiguous word. Criterion 2

performs better than Criterion 1 in almost all of the experiments. Thus, words with one tag (unambiguous), or fewer tags than the replaced word, are better substitutes than the words that have more tags than the replaced word.

5.5.2 Experiments on the Number of Substitutes

	Number of Substitutes				
Data	0	1	10	25	50
12K	72.19	73.85	74.67	74.61	74.40
24K	74.38	77.51	77.48	77.43	77.80
48K	77.43	79.78	80.06	79.72	79.40

Table 5.10: Results of our approach on different corpora with different number of substitute words per ambiguous word. Selection criterion 1 is used to obtain these results, and accuracies are reported as percentages.

In this set of experiments, we vary the number of substitute words for each ambiguous word in a given sentence. This directly affects the number of artificial sentences generated. We run our method on the corpora PTB12K, PTB24K, and PTB48K with 1, 10, 25, and 50 substitute words per ambiguous word. Table 5.10 shows the results of our experiments. The performance of our method does not change significantly as the number of substitutes increases.

5.5.3 Out-of-Vocabulary (OOV) Words

To observe the performance of our model on OOV words, we tested the data-enhanced Viterbi on a corpus that is different from the one used during the training of the model. The test corpus thus includes some words that are not observed in training data.

We do not perform any smoothing on the model parameter θ, that is, we do not assign non-zero OOV word probabilities. Hence θ does not contain any information related to OOV words. The main disadvantage of smoothing is that it assigns the same tag distribution to all OOV words. In our method, by replacing each OOV word in a sentence S with the most likely word that can be used in the same context, we get another sentence S' that consists only of in-vocabulary (IV) words. We repeat this step to construct more and more artificial sentences like S', and then proceed with the Viterbi algorithm.

We performed two experiments: (1) the possible tags of the OOV words are known, and (2) the possible tags of the OOV words are unknown. In the first case, the selected substitutes words and the OOV word have at least one common tag in the worst case. In the latter case, this is not guaranteed since the possible tags are unknown. In these experiments, we used the first PTB24K words of Penn Treebank as the training corpus and 5 random 12K words corpora from the Penn Treebank as the test corpora. Since the model trained on the PTB24K corpus has no information related to the OOV words, the Viterbi search algorithm assigns zero to all OOV word with certain tag probabilities, and thus leads to the generation of zero clique. The smoothing that we use assigns negligible probabilities to all zero entries of $P(word|tag)$. Therefore only the contextual information of hidden states is utilized in assigning the correct tag of the OOV word. The baseline model is trained on the PTB24K corpus. The performance of this model on the test corpus with OOV is summarized in Table 5.11 in the row labeled "Unsup. Baseline."

When the possible tags of the OOV word are given, our method improves HMM-EM. Its performance on both IV and OOV words is significantly better than the baseline score, which implies that artificial sentences without OOV words successfully represent the hidden tag sequence of S and improve the algorithm performance. Even when possible tags cannot be found for OOV word, our method still improves HMM-EM. While the improvement resulting from this method is significantly higher compared to the baseline system, it is significantly lower than in the previous case. The main reason for such a performance

Accuracy(%)	All words	IV	OOV
With tags	74.91	73.77	79.21
Without tags	66.36	72.53	40.06
Unsup. Baseline	57.5	66.57	18.84

Table 5.11: The performance of the data-enhanced Viterbi algorithm that uses the 25 most likely unambiguous substitutes for each ambiguous word. All of the results are averaged over 5 test corpora. The first two rows give the performance of the system with and without the possible tags of the OOV words, and the last row gives the performance of the baseline system. The average percentage of OOV words is 18.99%.

difference is that the most frequent words of SLM dominate the substitute word sets, and cannot be eliminated because we do not have the possible tags for the OOV words.

5.6 Conclusion

In this chapter we present an application a dictionary reduction data enhancement method that can be applied to HMM-based models. With the help of a SLM, our system created artificial sentences that are assumed to have the same POS tag sequence with the target sentence.

In dictionary reduction method, I use artificial sentences to reduce the size of the word–tag dictionary. To test our method we used HMM-EM as the unsupervised model. Our method significantly improves the prediction accuracy of the unsupervised first order HMM-EM system in all of the POS groups and achieves 92.25% and 92.47% word tagging accuracy on the 24K and 48K word corpora respectively. We also tested our model on a coarse grained dictionary with 17 tags and achieved an accuracy of 92.8%.

In data enhanced Viterbi, I fed them to the model based approach to jointly predict the

optimum tag sequence of the target and its artificial counterparts. To present our method we used HMM-EM as the model based approach. Our method significantly improves the prediction accuracies of the unsupervised HMM-EM system in all of the corpora and achieves %2.6 error accuracy gain in the worst case and %5.6 in the best case.

Finally, I also demonstrated the performance gain with data enhancement on OOV words whether the possible tags of OOV words are available or not. In both cases our method increases the accuracy of the baseline system substantially. Moreover the prediction accuracy of OOV words are comparable to the accuracy of IV words when the possible tags are available for the OOV word. Thus we concluded that our method successfully generates artificial sentences without OOV words for a given sentence with OOV words.

In this chapter, I show that unambiguous replacements of an ambiguous word can reduce the amount of the ambiguity thus replacement words might also be incorporated into the unsupervised disambiguation problems.

5.7 Future Work

In Section 4.4.1 and 4.4.2, the statistical language models (SLMs) have been used in an ad-hoc manner to generate and use likely substitutes, or have been incorporated into a probabilistic model to provide the $\Pr(\text{word}|\text{context})$ term. These models assign the same uniform weight to all likely substitutes, even though the language model ranks these substitutes with different probabilities.

Some ideas for future work as follows: We intend to use the statistical language model for generating artificial data to expedite the disambiguation process during the expectation maximization. Specifically, we assume that the same hidden tag sequence that has generated a particular test sentence can also generate artificial sentences where one of the words has been replaced with a likely substitute. Thus, words that are observed frequently in the context of an ambiguous target are incorporated into the disambiguation process. More-

over, every substitute is weighted according to the probability estimate assigned to it by the language model. As a result, the EM algorithm estimates the probabilities according to not only the original observation sequences but also the artificial observation created and weighted by the statistical language model.

By the very the nature of EM, the estimation process can converge to a local maximum, which might decrease the accuracy of disambiguation. On the other hand, non-parametric Bayesian approaches do not suffer from this convergence problem. Consequently, we intend to replace EM with non-parametric Bayesian methods.

Finally, the quality of the likely replacements and the probabilities assigned to them have a crucial role in the disambiguation process. We will conduct experiments to investigate the effect of in-domain and out-of-domain corpora on the selection of replacements.

Chapter 6

MORPHOLOGICAL DISAMBIGUATION

The terminology of morphological disambiguation can be applied to agglutinative languages as follows: In the latter, the equivalent of POS tagging is morphological disambiguation and the equivalent of the term *tag* is *parse*. The morphological disambiguation problem can therefore be defined as selecting the correct parse of a word in a given context from the possible candidate parses of the word. Our approach does not directly assign any parses to the target word. Instead, it uses the target word to limit the set of possible parses, and then assigns probabilities to these depending on the context. This approach has been previously applied to the word-sense disambiguation problem where the aim was to determine the sense of an ambiguous word in a given context (Pustejovsky et al., 2004).

The tags in English and the parses in an agglutinative language differ in a major respect. Unlike the tags in English, the number of theoretically possible parses in agglutinative languages can be infinite although the number of features is finite. Therefore, even in a training corpus of 1 million words, it is possible to observe thousands of different possible parses—a situation that leads to data sparseness. Finally, our model can be applied to any agglutinative language since it does not require any hand-crafted rules and does not depend on the knowledge of a native speaker.

To predict the correct parse of an ambiguous word, we proceed as follows: First, the possible parses are generated using a morphological analyzer. Then, using the language model together with the vocabulary of the corpus, a probabilistic model is applied to each ambiguous word. The resulting disambiguation accuracy for the ambiguous words is 64.5%, whereas 31.9% and 71.0% are the unsupervised and supervised baselines, re-

spectively.

Morphological disambiguation, an important step in a number of NLP tasks, is quite crucial for agglutinative languages, such as Turkish, Finnish, Hungarian, and Czech. For example, a morphological analyzer used in conjunction with a disambiguator can significantly reduce the perplexity of a Turkish language model (Yuret and Biçici, 2009).

Three possible morphological parses for the Turkish word "*masalı*" are shown below. The candidate parses are generated using a morphological analyzer. The first token of the

masal	+Noun+A3sg+Pnon+Acc	(= the story)
masal	+Noun+A3sg+P3sg+Nom	(= his story)
masa	+Noun+A3sg+Pnon+Nom^DG+Adj+With	(= with tables)

analyzer output is the root of the word while the rest is the parse of the word that consists of features concatenated to each other either by a "+" or "^*DG*". The first two lines output by the analyzer for "*masalı*" have the same root, *masal* (= story) but different parses, while the last line has a different root *masa* (= table) and parse. Feature groups that are separated by a derivation boundary (^*DG*) are called "inflection groups" (OflazerH et al., 2002). The first feature following the root or a ^*DG* represents the part-of-speech (POS) tag of the new derived word. A morphological disambiguation system should pick the correct parse of the word "*masalı*", given the context in which this word appears.

In this chapter, we present an application of the probabilistic voting model (described in detail in Section 4.3) for the morphological disambiguation task of Turkish. The main idea behind the model is that instead of assigning parses to words, it assigns parses to the contexts of the words. The probability of the morphological analysis in a given context is estimated by a language model that is trained on an unlabeled corpus. Therefore, the model does not require any predefined rule set, and can be applied to any language as long as a parse (tag) dictionary for each word and a corpus are available.

The rest of this chapter is organized as follows: First, Section 6.1 reviews the related work on morphological disambiguation. Section 6.2 specifies in detail the steps of the probabilistic voting algorithm. Section 6.3 gives the details of the tag-parse dictionary construction and a simplification procedure that jointly increase the chance of there being shared parses between the entries of the dictionary. Section 6.4 summarizes our results on morphological disambiguation for Turkish. Finally, Section 6.6 discusses the probabilistic voting model and outlines the road map of future work.

6.1 Related Work

Several studies undertaken in the past decade have contributed to progress in the unsupervised morphological disambiguation of morphologically rich languages .

In Hebrew, a context-free model was used to estimate the morpho-lexical probabilities of a given word from an untagged corpus (Levinger et al., 1995). Like Turkish, Hebrew is a morphologically rich language, and morphemes in Hebrew can combine into a single word in both agglutinative and fusional ways. Thus a Hebrew word can have various segmentations and multiple morphological analyses. The method referred to above is very similar to ours because both use substitute words to disambiguate the target word. Our method uses one set of substitute words from the vocabulary while the other method explicitly uses a predefined set of rules to select the set of similar words for each target word before disambiguation takes place. Another important difference is that this method does not use any contextual information during the disambiguation task.

A more recent study has shown that morpheme-based segmentation and tagging in Hebrew can be learned simultaneously by using a stochastic unsupervised learning with HMM (Adler and Elhadad, 2006). Their model first estimates the probabilities of each segmentation and their possible tags by using a variation of the Baum-Welch algorithm. Then an adaptation of the Viterbi algorithm is applied to get the most probable segmentation and

tagging sequence.

6.2 Algorithm

Section 4.3 has described the mathematical framework of the model applied to the tag disambiguation task when a word-tag dictionary is available. The probabilistic model of Section 4.3 decomposes the problem into the estimation of $\Pr(t|c)$ and $\Pr(t|s,c)$, where t, c, and s represent the tag, the context, and the substitute words in c, respectively. Hence, we estimate $\Pr(s|c)$ using a statistical language model. We make the following two assumptions when estimating $\Pr(t|s,c)$.

1. **Pruning Assumption:** Every w has a possible tag set T_w which is available from the word-tag dictionary. Instead of assigning non-zero probabilities to all possible tags, our model simply assumes that, in the context of w, the only possible tags are the ones that are contained in T_w. Therefore, tags that are not in T_w have zero probability.

2. **Uniformity Assumption:** We assume that, given a substitute word s and context c, the distribution of the tags is uniform on $T_w \cap T_s$.

$$\Pr(t|s,c) = \begin{cases} \frac{1}{|T_w \cap T_s|} & \text{if } t \in T_w \cap T_s, \\ 0 & \text{otherwise.} \end{cases} \tag{6.1}$$

Another interpretation of this model is that each substitute word votes for the possible tags of the target word. The weight of the vote of s is determined by $\Pr(s|c)$. If the substitute and the target word have more than one common tag, then $\Pr(s|c)$ is equally distributed among the common tags.

The algorithmic steps of the disambiguator are specified below. Throughout this section, w_i denotes the i^{th} word from the set of target words W, c_i denotes the context of i^{th}

target word, s_{ij} denotes the j^{th} substitute of w_i, S_i denotes all substitutes of w_i, and T_i denotes the set of possible tags of w_i.

For all $w_i \in W$, we perform the following steps:

Algorithm

1. Calculate $P(s_{ij}|c_i)$ for each substitute, using the estimation method described in Chapter 2.

2. Determine the intersection of T_{w_i} with each $s_{ij} \in S_i$ for all $s \in S_{w_i}$. $P(t|v_{ij})$ is equal to $\frac{1}{|T_{w_i} \cap T_{s_{ij}}|}$ because of the uniformity assumption.

3. Select $t \in T_i$ that maximizes $P(t|c_i)$.

6.3 Word-tag Dictionary Construction and Simplification

The estimation quality of $P(t|c_w)$ is highly dependent on the parse set T_w of the target word. Using a Turkish morphological analyzer (OflazerH et al., 2002), we get the possible parses of the target word and its substitutes, and construct the word-parse dictionary. The analyzer produces the parses of each word as shown in the second column of Table 6.1. Because of the agglutinative nature of Turkish, the parses are complex and it is hard to find substitute words with common parses.

	Original Parse	Simplified Parse
masal	+Noun+A3sg+Pnon+Acc	Pnon+Acc
masal	+Noun+A3sg+P3sg+Nom	P3sg+Nom
masa	+Noun+A3sg+Pnon+Nom^DG+Adj+With	With

Table 6.1: Parse simplification of the word "*masalı*".

If the number of substitute words that have common parses with the target word is small, then $P(t|c_w)$ will be estimated using very few substitute words. Thus, instead of using the parses directly, we construct a discriminative minimal feature set T_w^* from T_w by using only the final inflection groups (IG) for each parse. To construct T_w^*, our model selects the minimum number of rightmost features from each of the last IGs such that these rightmost features uniquely discriminate the corresponding parse from the other parses in T_w. Table 6.1 illustrates the simplifications of the parses of the word "*masalı*".

6.4 Experiments and Results

	Test Set	Tagged Training Set
Sentences	446	50673
Tokens	5365	948404
Ambiguous tokens	2437(45.4%)	399223(42.1%)
Average Parses	1.85	1.76

Table 6.2: Test and Tagged Training Data Statistics

In this section, we present a number of experiments to observe the effects of the model parameters on the algorithm performance. We define an unsupervised and a supervised baseline on the test set to compare with the results of our method. The unsupervised baseline is calculated by randomly picking one of the parses of each word in the test set. To calculate a supervised baseline, we use a tagged training set consisting of 1 million words of semi-automatically disambiguated Turkish news text. Some brief statistics relevant to the tagged training set and the test set are presented in Table 6.2. Using the training set, the supervised baseline simply does majority voting for each word. If the target word does not exist in the training set, then the supervised baseline randomly picks one of the possible

parses of the missing word. The unsupervised baseline disambiguates 31.9% of the ambiguous words correctly, while the supervised baseline correctly disambiguates 71.0% of them. All the accuracy scores reported in this section are only for ambiguous words. The experiments in this section can be categorized as language model-related and substitute word-related[1].

6.4.1 Language Model

The substitutes are calculated using a language model that is trained on the Turkish corpus (Sak et al., 2008), as described in Section 2. The data set contains about 440 million words, and 10% of this data is extracted and used as the test set to calculate the perplexity of the language models. The SRILM toolkit is used to train the 4-gram model with Kneser-Ney interpolated smoothing, n-gram orders, and training corpus sizes. The effect of the training corpus size and the n-gram order on the model are discussed in the next section.

6.4.2 Corpus size

We used three corpora of different sizes to train the 4-gram language model and observe the performance of our disambiguator. For our experiments, we randomly select 1% and 10% of the original training corpus described in Section 2. The performance of the disambiguator with different corpus sizes are summarized in Table 6.3.

As Table 6.3 shows, the performance worsens as the corpus size decreases. However, using as little as10% of the corpus, our disambiguator can still achieve results comparable to the model using the whole corpus (in terms of 95% confidence interval). This is not the case when we use only 1% of the corpus, since in this case the loss of performance compared to the model using the whole corpus becomes statistically significant. These experiments

[1]For the sake of simplicity, all the reported results in this section are obtained (unless otherwise stated) by using the most frequent 200K words of the vocabulary.

Corpus Size	Accuracy
4M	60.4
40M	63.1
400M	**64.5**

Table 6.3: The performance of the model using the parse simplification together with different corpus sizes. Statistically significant results are displayed in bold ($p < 0.05$).

indicate that the performance may be improved by using larger Turkish corpora.

We used the Good-Turing and the Kneser-Ney smoothing techniques to observe the effect of smoothing on the probability estimates of our disambiguator. However, we found that the model performance is not significantly affected by the choice of the smoothing method. Similarly, 2, 3, and 4-gram language models were trained, but these different models did not have any significant effect on the model performance.

6.4.3 Number of Substitute Words

Number of Substitute	Accuracy
Top 10	63.4
Top 100	**64.3**
Top 200	**64.4**
Top 2000	**64.5**

Table 6.4: The performance of the model with different number of substitutes. Statistically significant results are displayed in bold ($p < 0.05$).

In these experiments, we calculate $P(s|c)$ of each substitute word, select 10, 100, 200,

and 2000 substitutes that have the highest $P(s|c)$, and use only these words to estimate $P(s|c)$. Table 6.4 shows the performance for different numbers of substitutes chosen. Increasing the number of substitutes to more than 100 makes no significant difference. Thus, the computational efficiency of our model can be increased by using a possibly faster algorithm that heuristically finds the top k substitutes with the highest $P(s|c)$.

6.5 Conclusion

In this chapter, we have presented an application of the probabilistic voting model (see Section 4.3) on the morphological disambiguation task of Turkish. The main idea behind our model is instead of assigning parses to words, it assigns parses to the contexts of the words. The probability of the morphological analysis in a given context is estimated by a language model that is trained on an unlabeled corpus. Therefore, the model does not require any predefined rule set and it can be applied to any language as long as a parse (tag) dictionary for each word and a corpus are available. We were able to achieve 64.5% accuracy on ambiguous words using this model. This accuracy might be improved by relaxing the uniformity assumption of the target word parse distribution and letting it to converge to the actual probabilities by using better statistical inference methods.

6.6 Future Work on Morphological Disambiguation

The accuracy might be improved by relaxing the uniformity assumption of the target word's parse distribution, and letting it converge to the actual probabilities by means of better statistical inference methods.

Chapter 7

WORD-SENSE DISAMBIGUATION

Word-sense disambiguation (WSD) is the task of identifying the correct sense of an ambiguous word in a given context. An accurate word-sense disambiguation system would be of benefit to applications such as machine translation and information retrieval. The most successful WSD systems to date rely on supervised learning and are trained on sense-tagged corpora. In this chapter, we present an unsupervised word-sense disambiguation algorithm that can leverage untagged text and can perform at the level of the best supervised systems for the all-noun disambiguation task.

Our unsupervised system uses the Web1T dataset (Brants and Franz, 2006a) for unlabeled examples. This dataset contains counts from a 10^{12}-word corpus derived from publicly available web pages.

The proposed approach to word-sense disambiguation is a probabilistic generative model to seamlessly integrate unlabeled text data into the model building process. Our approach makes use of the noisy channel model (Shannon, 1948), which has been an essential tool in fields such as speech recognition and machine translation. In this study, we demonstrate that the noisy channel model can also be a key component in unsupervised word-sense disambiguation, provided that we can solve the context-dependent sense distribution problem. In Section 4.5, we show one way to estimate the context-dependent sense distribution without using any sense-tagged data. Section 7.2 outlines the complete unsupervised WSD algorithm using this model. We estimate the distribution of coarse-grained semantic classes rather than fine-grained senses. The solution uses the two distributions for which we do have data: the distribution of words used to express a given sense, and the distribution of

words that appear in a given context. The first can be estimated using WordNet sense frequencies, and the second can be estimated using an n-gram language model, as described in Section 7.3.

7.1 Related Work

For a general overview of different approaches to word-sense disambiguation, see (Navigli, 2009; Stevenson, 2003). The Senseval and SemEval workshops (Cotton et al., 2001; Mihalcea and Edmonds, 2004; Agirre et al., 2007) are good sources of recent work, and have been used in this chapter to benchmark our results.

Generative models that utilize the noisy channel framework have previously been used in speech recognition (Bahl et al., 1983), machine translation (Brown et al., 1990), question answering (Echihabi and Marcu, 2003), spelling correction (Brill and Moore, 2000), and document compression (Daume III and Marcu, 2002), among other applications. To our knowledge, our work is the first application of the noisy channel model to unsupervised word-sense disambiguation.

The use of statistical language models that rely on large corpora for word-sense disambiguation has been explored in (Yuret, 2007; Hawker, 2007). For the specific modeling techniques used in this paper, see (Yuret, 2008). For a more general review of statistical language modeling, see (Chen and Goodman, 1999; Rosenfeld, 2000; Goodman, 2001).

Grouping similar senses into semantic classes for word-sense disambiguation has been explored previously. Senses that are similar have been identified using WordNet relations (Peters et al., 1998; Crestan et al., 2001; Kohomban and Lee, 2005), discourse domains (Magnini et al., 2003), annotator disagreements (Chklovski and Mihalcea, 2003), and other lexical resources such as Roget (Yarowsky, 1992), LDOCE (Dolan, 1994), and ODE (Navigli, 2006).

Ciaramita and Altun (2006) build a supervised HMM tagger using "supersenses", which

are essentially the 25 WordNet noun categories that we have used in our first experiment in addition to the 15 verb categories similarly defined. They report a supersense precision of 67.60 for nouns and verbs of Senseval-3. Table 7.2 gives our supersense score of 78% for Senseval-3 nouns. However, the results are not directly comparable since they do not report the noun and verb scores separately, nor do they calculate the corresponding fine-grained score to compare with other Senseval-3 results.

Kohomban and Lee (2007) go beyond the WordNet categories to utilize lexicographer files and experiment with clustering techniques for constructing their semantic classes. Their classes make use of local features from sense-labeled data to optimize feature–class coherence rather than adhering to the WordNet hierarchy. Their supervised system achieves an accuracy of 74.7% on Senseval-2 nouns and 73.6% on Senseval-3 nouns.

The systems mentioned so far are supervised WSD systems. Agirre and Martinez (2004) explore the large-scale acquisition of sense-tagged examples from the Web, and use this corpus to train supervised, minimally-supervised (requiring sense bias information from hand-tagged corpora, similar to our system), and fully unsupervised WSD algorithms. Their results on the Senseval-2 lexical sample data compare favorably to other unsupervised systems. Martinez et al. (2008) report work with a similar set of systems that has been trained using automatically acquired corpora on Senseval-3 nouns. Their minimally supervised system achieves a 63.9% accuracy on polysemous nouns from Senseval-3 (corresponding to 71.86% on all nouns).

7.2 Algorithm

Section 4.5 described how to apply the noisy channel model for WSD in a single context. In this section, we present the steps that we follow in our experiments to simultaneously apply the noisy channel model to all the contexts in a given word-sense disambiguation task.

Algorithm 1

1. Let $\mathbf{W}$ be the vocabulary. In this study, we took the vocabulary to be the set of approximately 12,000 nouns in WordNet that have non-zero sense frequencies.

2. Let $\mathbf{T}$ be the set of senses or semantic classes to be used. In this study, we used various partitions of noun synsets as semantic classes.

3. Let $\mathbf{C}$ be the set of contexts (9-word windows for a 5-gram model) surrounding each target word in the given WSD task.

4. Compute the matrix $\mathbf{WC}$ where $\mathbf{WC}_{ik} = \Pr(W = i|C = k)$. Here i ranges over the vocabulary $\mathbf{W}$ and k ranges over the contexts $\mathbf{C}$. This matrix concatenates the (w) word distribution vectors from Eqn. 4.19 for each context. The entries of the matrix are computed using the n-gram language model described in Section 7.3.

5. Compute the matrix $\mathbf{WT}$ where $\mathbf{WT}_{ij} = \Pr(W = i|T = j)$. Here i ranges over the vocabulary $\mathbf{W}$ and j ranges over the semantic classes $\mathbf{T}$. The entries of the matrix are computed using the WordNet sense frequencies.

6. Compute the matrix $\mathbf{TC} = \mathbf{WT}^{+} \times \mathbf{WC}$ where $\mathbf{TC}_{jk} = \Pr(T = j|C = k)$. Here j ranges over the semantic classes $\mathbf{T}$ and k ranges over the contexts $\mathbf{C}$. This step computes the pseudoinverse solution, which was described in Section 4.5, simultaneously for all the contexts. The resulting $\mathbf{TC}$ matrix is a concatenation of the (t) solution vectors from Eqn. 4.19 for each context. $\mathbf{WT}^{+}$ is the pseudoinverse of the matrix $\mathbf{WT}$.

7. Compute the best semantic class for each WSD instance by using $\operatorname{argmax}_T \Pr(T|W, C) \propto \Pr(W|T) \Pr(T|C)$. Here $\Pr(T|C)$ comes from the column of the $\mathbf{TC}$ matrix that

corresponds to the context of the WSD instance, and $\Pr(W|T)$ comes from the row of the $\mathbf{WT}$ matrix that corresponds to the word to be disambiguated.

8. Compute the fine-grained answer for each WSD instance by taking the most frequent (lowest numbered) sense in the chosen semantic class.

9. Apply the *one sense per discourse* heuristic: if a word is found to have multiple senses in a document, then replace it with the majority answer.

Appendix A discusses possible scaling issues of Step 6 and offers alternative solutions. We use the pseudo-inverse solution in all our experiments because it can be computed fast and none of the alternatives we tried made a significant difference in WSD performance.

7.3 Estimation Procedure

In Section 4.5, we showed how the unsupervised WSD problem, expressed as a noisy channel model, can be decomposed into the estimation of two distributions: $\Pr(W|T)$ and $\Pr(W|C)$. In this section, we describe our estimation procedure for these two distributions.

To estimate $\Pr(W|T)$, namely, the distribution of words that can be used to express a given meaning, we used the WordNet sense frequencies.[1] We did not perform any smoothing for the zero counts, but used the maximum likelihood estimate: $\text{count}(W,T)/\text{count}(T)$. As described in later sections, we also experimented with grouping similar WordNet senses into semantic classes. In this case, T stands for the semantic class, and the counts from various senses of a word in the same semantic class are added together to estimate $\Pr(W|T)$.

To estimate the distribution of words in a given context, $\Pr(W|C)$, we used a 5-gram language model. Estimating $\Pr(W|C)$ for each context is expensive because the number of

[1]The sense frequencies were obtained from the `index.sense` file included in the WordNet distribution. We had to correct the counts of three words (person, group, and location) whose WordNet counts unfortunately include the corresponding named entities and are thus inflated.

words that need to be considered is large. The Web 1T dataset contains 13.5 million unique words, and WordNet defines about 150,000 lemmas. To make the computation feasible, we needed to limit the set of words for which $\Pr(W|C)$ had to be estimated. We limited our set to those WordNet lemmas which had the same part of speech as the target word. We further required the word to have a non-zero count in WordNet sense frequencies. The inflection and capitalization of each word W was automatically matched to the target word. As a result, we estimated $\Pr(W|C)$ for about 10,000 words for each noun context, and assumed that the other words had zero probability. The n-grams required for all the contexts were listed, and their counts were extracted from the Web 1T dataset in one pass. The $\Pr(W|C)$ was estimated for all the words and contexts on the basis of these counts. At the end, we used only the 100 most likely words in each context for efficiency, as the difference in results using the whole distribution was not significant. For more details on smoothing with a large language model, see (Yuret, 2008) (we did not see a significant difference in WSD performance based relative to the smoothing method used).

7.4 Semantic Classes

Our algorithm internally differentiates semantic classes rather than fine-grained senses. Using fine-grained senses in the noisy channel model would be computationally expensive, because the word-sense matrix would need to be inverted (see Eqn. 4.19). It is also unclear whether using fine-grained senses for model building will lead to better learning performance: the similarity between the distributions of related senses is ignored and the data becomes unnecessarily fragmented.

Even though we use coarse-grained semantic classes for model building, we use fine-grained senses for evaluation. During evaluation, the coarse-grained semantic classes predicted by the model are mapped to fine-grained senses by picking the lowest numbered

WordNet sense in the chosen semantic class.[2] This is necessary to perform a meaningful comparison with published results.

We take semantic classes to be groups of WordNet synsets defined using the hypernym hierarchy (see Section 7.1 for alternative definitions). Section 7.5 presents three WSD experiments using different sets of semantic classes at different levels of granularity. In each experiment, we designate a number of synsets that are high in the WordNet hypernym hierarchy as "head synsets", and use their descendants to form the separate semantic classes.

An arbitrary set of head synsets will not necessarily have mutually exclusive and collectively exhaustive descendants. To assign every synset to a unique semantic class, we impose an ordering on the semantic classes. Each synset is assigned only to the first semantic class of whose head it is a descendant according to this ordering. If there are synsets that are not descendants of any of the heads, they are collected into a separate semantic class created for that purpose.

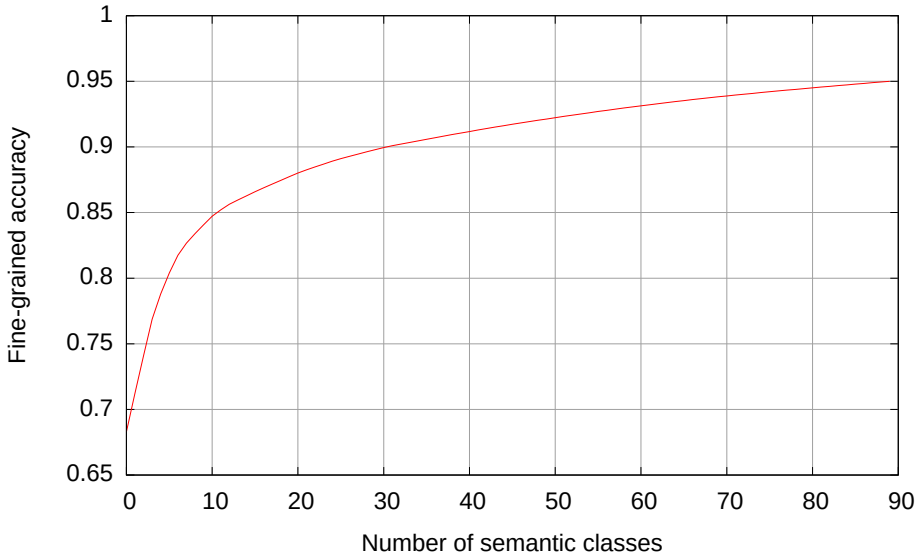
Using the coarse-grained semantic classes for prediction, Algorithm 1 will be unable to return the correct fine-grained sense when this is not the lowest-numbered sense in a semantic class. To quantify the restrictive effect of working with a small number of semantic classes, Figure 7.1 plots the number of semantic classes versus the best possible oracle accuracy for the nouns in the SemCor corpus. To compute the oracle accuracy, we assume that the program can find the correct semantic class for each instance, but has to pick the first sense in that class as the answer. To construct a given number of semantic classes, we used the following algorithm:

Algorithm 2

1. Initialize all synsets to be in a single "default" semantic class.

[2]The sense numbers are ordered by the frequency of occurrence in WordNet.

Figure 7.1: Upper bound on fine-grained accuracy for a given number of semantic classes



2. For each synset, compute the following score: the oracle accuracy achieved if that synset and all its descendants are split into a new semantic class.

3. Take the synset with the highest score, and split that synset and its descendants into a new semantic class.

4. Repeat steps 2 and 3 until the desired number of semantic classes is achieved.

The upper bound on fine-grained accuracy for a small number of semantic classes is surprisingly high. In particular, the best reported noun WSD accuracy (78%) is achievable if we can perfectly distinguish the five semantic classes from each other.

7.5 Three Experiments

We ran three experiments with the noisy channel model using different sets of semantic classes. The first experiment uses the 25 WordNet semantic categories for nouns, the second experiment looks at what happens when we group all the senses to just two or three semantic classes, and the final experiment optimizes the number of semantic classes using one dataset (which gives 135 classes) and reports the output using another dataset.

The noun instances from the last three SensEval/SemEval English all-words tasks are used for evaluation. We focus on the disambiguation of nouns for several reasons. Nouns constitute the largest portion of content words (48% of the content words in the Brown corpus (Kucera and Francis, 1967) are nouns). For many tasks and applications (e.g. web queries, (Jansen et al., 2000)) nouns are the most frequently encountered and important part of speech. Finally, WordNet has a more complete coverage of noun-semantic relations than of other parts of speech—an important consideration for our experiments with semantic classes.

As described in Section 7.2 we use the model to assign each ambiguous word to its most likely semantic class in all the experiments. The lowest-numbered sense in that class is taken as the fine-grained answer. Finally, we apply the *one sense per discourse* heuristic: if the same word has been assigned more than one sense within the same document, then we take a majority vote and use sense numbers to break the ties.

Table 7.1 gives some baselines for comparison. The performance of the best supervised and unsupervised systems on noun disambiguation for each dataset are given. The first sense baseline (FSB) is obtained by always picking the lowest-numbered sense for the word in the appropriate WordNet version. We prefer the FSB baseline over the commonly used most frequent sense baseline because the tie breaking is unambiguous for the former. All the results reported are for fine-grained sense disambiguation. The top 3 systems given in the table for each task are all supervised systems; the result for the best unsupervised

Task	WN	Nouns	FSB	1st	2nd	3rd	Unsup
senseval2	1.7	1067	71.9	78.0	74.5	70.0	61.8
senseval3	1.7.1	892	71.0	72.0	71.2	71.0	62.6
semeval07	2.1	159	64.2	68.6	66.7	66.7	63.5
total		2118	70.9	74.4	72.5	70.2	62.2

Table 7.1: Baselines for the three SensEval English all-words tasks; the WordNet version used; number of noun instances; percentage accuracy of the first sense baseline, the top three supervised systems, and the best unsupervised system. The last row gives the total score of the best systems on the three tasks.

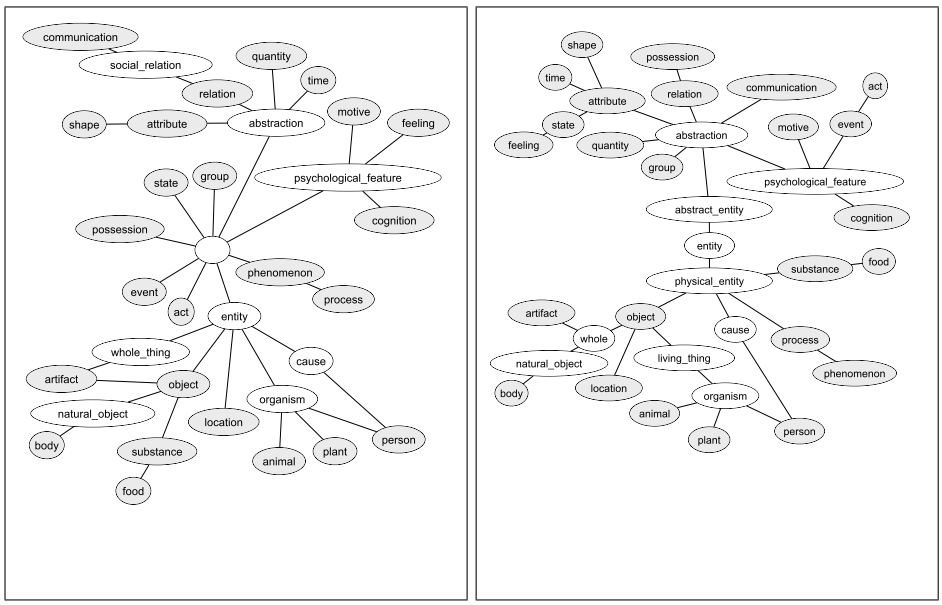
system is given in the last column. The reported unsupervised systems do use the sense ordering and frequency information from WordNet.

7.5.1 First experiment: the 25 WordNet categories

In previous work, the descendants of 25 special WordNet synsets (known as the "unique beginners") have been used as the coarse-grained semantic classes for nouns (Crestan et al., 2001; Kohomban and Lee, 2005). These unique beginners were used to organize the nouns into 25 lexicographer files, depending on their semantic category during WordNet development. Figure 7.2 shows the synsets at the top of the noun hierarchy in WordNet. The 25 unique beginners have been shaded, and the two figures show how the hierarchy evolved between the two WordNet versions used in this study.

We ran our initial experiments using these 25 WordNet categories as semantic classes. The distribution of words for each semantic class, $\Pr(W|T)$, is estimated from WordNet sense frequencies. The distribution of words for each context, $\Pr(W|C)$, is estimated using a 5-gram model derived from the Web 1T corpus. The system first finds the most likely

Figure 7.2: The top of the WordNet noun hypernym hierarchy for version 1.7 (left) and version 2.1 (right). The 25 WordNet noun categories are shaded.



semantic class using the noisy channel model, then picks the first sense in that class. Table 7.2 gives the results for the three datasets. These results are significantly better than the previously reported unsupervised results.

To illustrate which semantic classes are the most difficult to disambiguate, Table 7.3 gives the confusion matrix for the Senseval2 dataset. We can see that the frequently occurring concrete classes like person and body are disambiguated well. The most serious sources of error are the abstract classes like act, attribute, cognition and communication. These 25 classes may not be the ideal candidates for word-sense disambiguation. Even though they allow a sufficient degree of fine-grained distinction (Table 7.2 shows that we can get 85–90% if we could pick the right class every time), they seem too easy to confuse. In the next few experiments, we will use these observations to design better sets of semantic

Dataset	CorrClass	MaxScore	Score
senseval2	85.1	90.3	77.7
senseval3	78.0	88.7	70.1
semeval07	75.5	86.2	64.8
total	81.4	89.3	73.5

Table 7.2: The performance of the noisy channel model with the 25 semantic classes found in WordNet lexicographer files. The columns give the dataset, the percentage of times the model picks the correct semantic class, maximum possible fine-grained score if the model had always picked the correct class, and the actual score.

classes.

7.5.2 Second experiment: distinguishing mental and physical concepts

Figure 7.1 shows that the upper bound for fine-grained disambiguation is relatively high even for a very small number of semantic classes. In our next experiment we look at how well our approach can perform differentiating only two or three semantic classes.

Using Algorithm 2 on the appropriate version of SemCor, we pick the head synsets to be used for defining the semantic classes. Figure 7.2 shows that the top level of the hypernym hierarchy has changed significantly between the WordNet versions. Thus, different head synsets are chosen for different datasets. However the main distinction captured by our semantic classes seems to be between mental and physical concepts. Table 7.4 gives the results. The performance with a few semantic classes is comparable to the top supervised algorithms in each of the three datasets.

	ac	an	ar	at	bo	co	co	ev	fe	fo	gr	lo	mo	ob	pe	ph	po	pr	qu	re	sh	st	su	ti	F	A
act	58	0	4	7	0	7	2	3	2	0	5	0	0	0	0	0	1	4	1	1	0	2	0	0	9.1	59.8
animal	0	17	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2.1	77.3
artifact	0	0	66	2	0	0	6	5	0	0	5	1	0	1	0	0	0	0	0	0	3	1	0	0	8.4	73.3
attribute	3	0	0	19	0	3	0	0	0	0	0	1	0	1	2	0	2	1	0	0	1	3	0	0	3.4	52.8
body	0	0	0	0	123	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	11.6	99.2
cognition	6	0	1	2	0	82	5	1	0	0	0	2	1	1	1	0	1	0	5	1	0	5	0	0	10.7	71.9
communicat	2	0	1	0	0	2	29	1	0	0	0	2	5	0	0	1	0	0	0	1	0	0	0	2	4.3	63.0
event	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	2.0	90.5
feeling	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.4	100.
food	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1	100.
group	0	0	0	2	0	5	0	0	0	0	69	2	0	3	0	0	0	0	0	1	1	0	0	1	7.9	82.1
location	0	0	0	1	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	2.2	95.7
motive	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0.2	50.0
object	2	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	1	0.7	14.3
person	2	4	0	0	0	1	1	0	0	0	1	0	0	0	168	0	0	0	0	0	0	0	0	0	16.6	94.9
phenomenon	1	0	0	1	0	1	0	2	0	0	0	0	0	0	0	3	0	0	0	3	0	0	0	0	1.0	27.3
possession	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0.4	100.
process	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	12	0	0	0	1	0	0	1.4	80.0
quantity	0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	10	0	0	0	0	0	1.2	76.9
relation	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0.3	66.7
shape	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0.1	100.
state	1	0	1	5	0	1	1	2	0	0	1	0	0	0	1	0	0	0	0	0	0	98	0	0	10.4	88.3
substance	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0.9	100.
time	1	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	44	4.8	86.3

Table 7.3: Confusion matrix for Senseval2 data with the 25 WordNet noun classes. The rows are actual classes and the columns are predicted classes. Column names have been abbreviated to save space. The last two columns give the frequency of the class (F) and the accuracy of the class (A).

Dataset	Heads	CorrClass	MaxScore	Score
senseval2	entity/default	86.6	76.8	74.9
senseval3	entity/default	94.2	75.8	71.2
senseval3	object/entity/default	93.8	77.4	72.9
semeval07	psychological-feature/default	91.2	74.8	68.6

Table 7.4: The performance of the noisy channel model with two to three semantic classes. The columns give the dataset, the head synsets, the percentage of times the model picks the correct semantic class, maximum possible fine-grained score if the model had always picked the correct class, and the actual score.

7.5.3 Third experiment: tuning the number of classes

Increasing the number of semantic classes has two opposite effects on word-sense disambiguation performance. The higher the number, the finer distinctions we can make, and the maximum possible fine-grained accuracy goes up. However the more semantic classes we define, the more difficult it becomes to distinguish them from one another. For an empirical analysis of the effect of semantic class granularity on the fine-grained WSD accuracy, we generated different sets of semantic classes using the following algorithm:

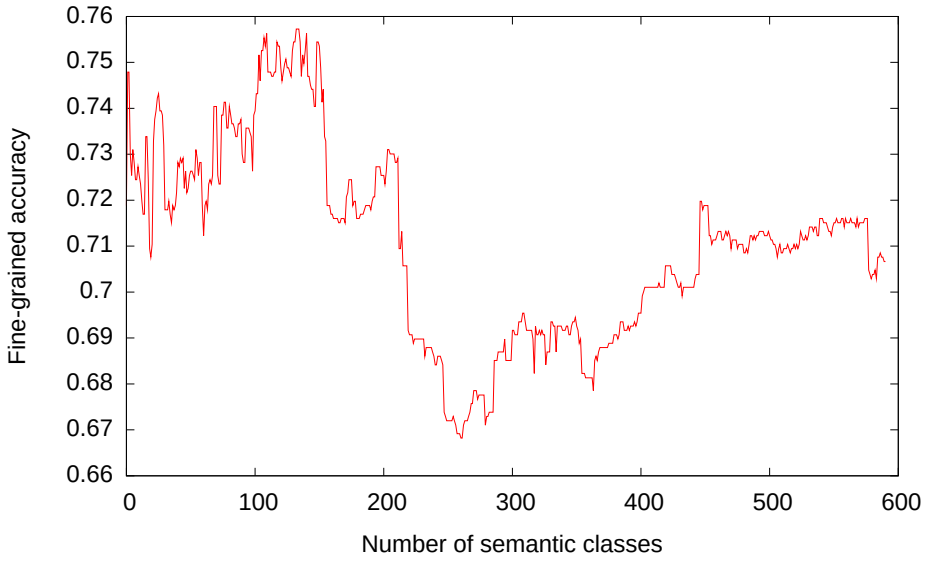
Algorithm 3

1. Sort all the synsets according to their "subtree frequency": i.e. the total frequency of its descendants in the hypernym tree.

2. Take the desired number of synsets with the highest subtree frequency and use them as head synsets, i.e. split their descendants into separate semantic classes.

Figure 7.3 shows the fine-grained accuracy we achieved on the Senseval2 dataset with up to 600 semantic classes, as generated by Algorithm 3. Note the differences from Fig-

Figure 7.3: The fine-grained accuracy on Senseval2 dataset for a given number of semantic classes



ure 7.1: (i) Figure 7.1 gives the best possible oracle accuracy, while Figure 7.3 gives the actual WSD accuracy. (ii) Algorithm 2 chooses the head synsets on the basis of their oracle score, while Algorithm 3 chooses them on the basis of their subtree frequency.

As we suspected, the relationship is not simple or monotonic. However, one can still identify distinct peaks at 3, 25, and 100–150 semantic classes. One hypothesis is that these peaks correspond to "natural classes" at different levels of granularity. Here are some example semantic classes from each peak:

3 classes	entity, abstraction
25 classes	action, state, content, location, attribute, ...
135 classes	food, day, container, home, word, business, feeling, material, job, man, ...

To test the out-of-sample effect of tuning the semantic classes determined by the peaks of Figure 7.3, we used the SemEval-2007 dataset as our test sample. When the 135 semantic

classes from the highest peak were used for the disambiguation of nouns in the SemEval-2007 dataset, an accuracy of 69.8% was achieved. This is higher than the accuracy of the best supervised system on this task (68.6%), although the difference is not statistically significant.

7.6 Conclusion

We have introduced a new generative probabilistic model based on the noisy channel framework for unsupervised word sense disambiguation. The main contribution of this model is the reduction of the word sense disambiguation problem to the estimation of two distributions: the distribution of words used to express a given sense, and the distribution of words that appear in a given context. In this framework, context similarity is determined by the distribution of words that can be placed in the given context. This replaces the ad hoc contextual feature design process by a statistical language model, allowing the advances in language modeling and the availability of large unlabeled corpora to have a direct impact on WSD performance. We have provided a detailed analysis of using coarse-grained semantic classes for fine-grained WSD. The noisy channel model is a good fit for class-based WSD, where the model decides on a coarse-grained semantic class instead of a fine-grained sense. The chosen semantic class is then mapped to a specific sense based on the WordNet ordering during evaluation. We show that the potential loss from using coarse-grained classes is limited, and state-of-the-art performance is possible using only a few semantic classes. We explore semantic classes at various levels of granularity and show that the relationship between granularity and fine-grained accuracy is complex, thus more work is needed to determine an ideal set of semantic classes. In several experiments we compare the performance of our unsupervised WSD system with the best systems from previous Senseval and SemEval workshops. We consistently outperform any previously reported unsupervised results and achieve comparable performance to the best supervised results.

7.7 Future Work

In order to determine the natural sense clusters, we will apply the non-parametric Bayesian approach to the raw text data. We have implemented the non-parametric hierarchical Dirichlet model to determine the latent sense clusters. However, three main challenges remain:

1. The size of the raw text should be decreased reasonably in order to apply the non-parametric methods. Otherwise the method would be impractical because of memory limitations.

2. The convergence of the algorithm is determined by the model hyper-parameters. Hence the suitable set of parameters needs to be determined for convergence to the reasonable latent sense clusters.

3. The latent sense clusters determined by the model should be mapped to the Wordnet clusters in order to compare our model with the relevant work in the literature.

Chapter 8

PART OF SPEECH INDUCTION

Unsupervised part-of-speech (POS) induction aims to cluster words into syntactic categories using unlabeled, plain text input. The problem of induction is important for studying under-resourced languages that lack labeled corpora and high quality dictionaries. It is also essential in modeling language acquisition by children because every child manages to induce syntactic categories without access to labeled sentences, labeled prototypes, or dictionary constraints (Ambridge and Lieven, 2011). Finally, categories induced from data may point to shortcomings or inconsistencies of hand-labeled categories, and help improve natural language processing systems when used as additional features.

A *Word-based* POS induction systems clusters all instances of a word into a single category. (We will refer to this behavior as the *one-tag-per-word assumption*.) An *Instance-based* systems clusters each occurrence of a word separately, and can thus handle ambiguous words.

Examples of word-based systems include the ones that represent each word by the vector of neighboring words (context vectors) and cluster them (Schütze, 1995; Lamar et al., 2010b,a), use a prototypical bi-tag HMM that assigns each word to a latent class (Brown et al., 1992; Clark, 2003), restrict a HMM-based Pitman-Yor process to perform one-tag-per-word inference (Blunsom and Cohn, 2011), define a word-based Bayesian multinomial mixture model (Christodoulopoulos et al., 2011), or construct word vector representations derived from co-occurrences with contextual features (Yatbaz et al., 2012).

The obvious limitation of the one-tag-per-word assumption is that instances of ambiguous words that have more than one POS role are grouped into the same class. For example,

the word *offer* is tagged as NN(399), VB(105) and VBP(34)[1] in its 538 occurrences in the human labeled Wall Street Journal (WSJ) Section of the Penn Treebank (PTB) corpus (Marcus et al., 1999). If all instances of *offer* are assigned to the most frequent tag NN, then 36% (139/538) of them will be erroneously labeled. In spite of this shortcoming, word-based POS induction systems generally do better than instance-based systems because the one-tag-per-word assumption is mostly valid: 93.69% of the word occurrences are tagged with their most frequent POS tag in the PTB (Toutanova et al., 2003).

In order to handle ambiguous words, models without a strict one-tag-per-word assumption need to group word *instances* into clusters according to their contexts. Some of these instance-based models exhibit the bias of assigning too few tags to words by using sparse priors in a Bayesian setting (Goldwater and Griffiths, 2007b; Johnson, 2007a; Gao and Johnson, 2008), or posterior regularization (Ganchev et al., 2010). Schütze and Pedersen (1993) represent the context of a word instance by concatenating context vectors of its left and right neighboring words, and then clustering the word instances. Berg-Kirkpatrick et al. (2010) use an EM algorithm in which they replace the multinomial components with miniature logistic regressions to achieve the highest instance-based accuracy on PTB. Christodoulopoulos et al. (2010) select prototypes of each cluster from the output of Brown (1992), and feed them to a HMM model that can handle prototypes as features (Haghighi and Klein, 2006). However, none of these models achieve results comparable to the best word-based systems.

There are two concerns inherent in all distributional methods: (i) words that are generally substitutable, like "the" and "its", are placed in separate categories (DT and PRP$) by the gold standard, (ii) words that are generally not substitutable, like "do" and "put", are placed in the same category (VB). Freudenthal et al. (2005) point out that categories with

[1]NN, VB, and VBP are three POS tags from the Penn Treebank corpus. They correspond to singular noun, verb in base form, and non-3^{rd}person singular verb in present tense, respectively. The numbers in parentheses are the frequencies.

unsubstitutable words fail the standard linguistic definition of a syntactic category, and that children do not seem to make errors of substituting such words in utterances (e.g. *"What do you want?"* vs. **"What put you want?"*). Whether gold-standard part-of-speech tags or distributional categories are better suited to applications like parsing or machine translation can be best decided by extrinsic evaluation. In this study, we obtain our results by comparing induced tags to gold-standard part-of-speech tags, and leave their extrinsic evaluation for future work.

The rest of this section is organized into subsections as follows: First, Section 8.1 reviews the related work of POS induction. Then, Section 8.2 and 8.3 define the experimental settings and tag perplexity, respectively. Section 8.4 presents the experiments of clustering framework (see Section 4.1). Finally, Section 8.5 presents the experiments of co-occurence modeling (see Section 4.2).

8.1 Related Work

There are several good reviews of algorithms for unsupervised part-of-speech induction (Gao and Johnson, 2008; Christodoulopoulos et al., 2010) and models of syntactic category acquisition (Ambridge and Lieven, 2011).

This work is to be distinguished from the supervised part-of-speech disambiguation systems which use labeled training data (Toutanova et al., 2003), unsupervised disambiguation systems which use a dictionary of possible tags for each word (Yatbaz and Yuret, 2010), and prototype-driven systems which use a small set of prototypes for each class (Haghighi and Klein, 2006). The problem of induction is important for studying under-resourced languages that lack labeled corpora and high quality dictionaries. It is also essential in modeling language acquisition by children because every child manages to induce syntactic categories without access to labeled sentences, labeled prototypes, or dictionary constraints.

Models of unsupervised part-of-speech induction fall into two broad groups depending

upon the information they utilize: Distributional models use only word types and their context statistics. Word-feature models incorporate additional morphological and orthographic features.

8.1.1 Distributional models

Distributional models can be further categorized into three subgroups depending on their learning algorithm. The first subgroup represents each word type/token by its context vector, and then clusters these vectors accordingly (Schütze, 1995). Work in modeling the syntactic category acquisition by children has generally followed this clustering approach (Redington et al., 1998; Mintz, 2003). The second subgroup consists of probabilistic models that use the Hidden Markov Model (HMM) framework (Brown et al., 1992). A third group of algorithms constructs a low-dimensional representation of the empirical co-occurrence statistics of word types (Globerson et al., 2007) (covered in more detail in Section 4.2.2).

Clustering Clustering-based methods represent the context by the neighboring words, typically a single word on the left and a single word on the right. (This context is referred to as "frame", e.g., **the** *dog* **is***;* **the** *cat* **is**.) They cluster word types rather than word tokens according to the frames they occupy. Thus, they employ the one-tag-per-word assumption from the start (with the exception of (Mintz, 2003; St Clair et al., 2010) and some methods in (Schütze, 1995)). These methods may suffer from data sparsity caused by infrequent words and infrequent contexts. The solutions suggested to override this data sparsity either restrict the set of words and set of contexts needed to be clustered to the most frequently observed, or use dimensionality reduction. By defining context similarity as a function of the number of common frames, Redington et al. (1998) bypass the data sparsity problem but achieve lower scores than the best performing systems. Mintz (2003) uses only the most frequent 45 frames to cluster tokens, and achieves 98% unsupervised accuracy on

the tokens observed in the most frequent 45 frames. Similar to Mintz's work, St Clair et al. (2010) show that systems that model the left and right frames of tokens separately perform better than the frequent frames, both in terms of token clustering accuracy and the token coverage. Biemann (2006) contructs a graph-based view of the most frequent 10,000 words using contexts formed from the most frequent 150-200 words and clustering the tokens. Schütze (1995) and Lamar et al. (2010b) employ SVD to enhance similarity between less frequently observed word types and contexts. Lamar et al. (2010a) represent each context by the currently assigned left and right tag (which eliminates data sparsity) and cluster word types using a soft k-means style iterative algorithm. They report .708 many-to-one accuracy on the PTB—the best clustering result to date.

HMMs The prototypical bi-tag HMM model maximizes the likelihood of the corpus $w_1 \dots w_n$, with the likelihood defined as $P(w_1|c_1) \prod_{i=2}^{n} P(w_i|c_i)P(c_i|c_{i-1})$, where w_i are the word tokens and c_i are their (hidden) tags. One problem with such a model is its tendency to distribute probabilities equally, and the resulting inability to model highly skewed word-tag distributions observed in hand-labeled data (Johnson, 2007a). To favor sparse word-tag distributions, one can enforce a strict one-tag-per-word solution (type clustering) (Brown et al., 1992; Clark, 2003), use sparse priors in a Bayesian setting (Goldwater and Griffiths, 2007b; Johnson, 2007a), or use posterior regularization (Ganchev et al., 2010). Each of these techniques provide significant improvements over the standard HMM model. For example, Gao and Johnson (2008) show that sparse priors can gain 4% (.62 to .66 on the PTB) in cross-validated many-to-one accuracy. However Christodoulopoulos et al. (2010) show that the older one-tag-per-word models, such as (Brown et al., 1992), outperform the more sophisticated sparse prior and posterior regularization methods both in speed and accuracy (the Brown model gets .68 many-to-one accuracy on the PTB). Given that 93.69% of the word occurrences in human-labeled data are tagged with their most frequent part-of-speech tags (Toutanova et al., 2003), this is probably not surprising; one-tag-per-word is a

fairly good first approximation for induction.

8.1.2 Word-feature models

One problem with the algorithms in the previous section is the poverty of their input features. Of the syntactic, semantic, and morphological information underlying syntactic categories according to the linguists' claims, only a limited amount of syntactic information is represented in the input of context vectors or bi-tag HMMs. Experiments incorporating morphological and orthographic features into HMM-based models demonstrate significant improvements. (Clark, 2003; Berg-Kirkpatrick and Klein, 2010; Blunsom and Cohn, 2011) incorporate similar orthographic features and report improvements of 3, 7, and 10%, respectively, over the baseline Brown model.

(Clark, 2003; Blunsom and Cohn, 2011) cluster types by incorporating similar orthographic features, and report improvements of 3 and 10% respectively over the baseline Brown model. Berg-Kirkpatrick et al. incorporate orthographic features into EM algorithm where they replace the multinomial components with miniature logistic regressions and cluster tokens, to report improvement over the Brown model by 7%.

Christodoulopoulos et al. (2010) use prototype-based features as described in (Haghighi and Klein, 2006) with automatically induced prototypes, and report an 8% improvement over the baseline Brown model by clustering tokens. Abend et al. (2010) train a prototype-driven model with morphological features by first clustering the high frequent types as the landmarks and then assigning the remaining types to these landmark clusters. Christodoulopoulos et al. (2011) define a type-based Bayesian multinomial mixture model in which each word instance is generated from the corresponding word type mixture component, and word contexts are represented as features. They achieve a .728 MTO score by extending their model with additional morphological and alignment features gathered from parallel corpora. To our knowledge, nobody has yet tried to incorporate phonological or prosodic

features in a computational model for syntactic category acquisition.

8.1.3 Paradigmatic representations

Yatbaz et al. (2012) explore the paradigmatic representation of word contexts by modeling the co-occurrence of words and their substitutes within the CODE framework. Their experiments on the PTB types shows that the paradigmatic representation improves the state-of-the-art MTO and V-measure (VM) accuracies of both distributional models and models with additional word features. Our work builds on that preliminary work by (1) exploring induction of part-of-speechs at token level (in addition to type level), (2) improving the model for using additional features, and (3) experimenting with additional languages.

8.2 Evaluation

We report many-to-one (MTO) and V-measure (VM) scores for our experiments as suggested in (Christodoulopoulos et al., 2010). The many-to-one (MTO) evaluation maps each cluster to its most frequent gold tag and reports the percentage of correctly tagged instances. The MTO score naturally gets higher with increasing number of clusters, but it is an intuitive metric when comparing results with the same number of clusters.

The V-measure (VM) (Rosenberg and Hirschberg, 2007) is an information-theoretic metric that reports the harmonic mean of homogeneity (each cluster should contain instances of only a single class) and completeness (all instances of a class should be members of the same cluster). In Section 8.5.8, we argue that homogeneity is perhaps more important in part-of-speech induction, and suggest an MTO scheme with a fixed number of clusters as a more intuitive metric.

Figure 8.1 illustrates the cases that causes high and low homogeneity and completeness scores on two example clusters. Induced cluster c_1 on the top only consists of word instances that are gold-tagged as noun (*NN*), and hence the homogeneity of the cluster is

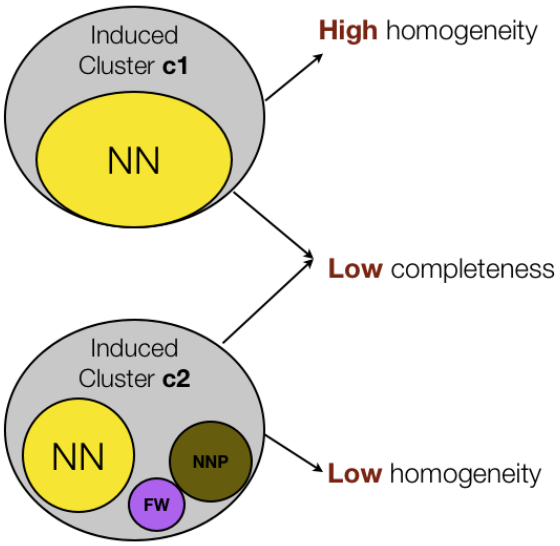


Figure 8.1: Illustration of the low and high homogeneity and completeness scores.

high. On the other hand, induced cluster c_2 consists of word instances with noun (*NN*), proper-noun (*NNP*), and foreign-word (*FW*) gold-tags, and hence has a low homogeneity score compared to c_1. Word instances with *NN* gold-tag are split into c_1 and c_2 which causes a low completeness accuracy.

8.3 Tag Perplexity

To measure the level of ambiguity of our models, we propose the tag perplexity of a word as a measure of its degree of ambiguity defined as follows:

$$GP(w) = 2^{-\sum_{i=1}^{N} \Pr(t|w) \log_2 \Pr(t|w)}$$

Here N is the number of words in the corpus, w is the target word, t is a cluster-id or a tag, and $\Pr(t|w)$ is the probability that the word w has been assigned the tag t. A word with N equally probable tags would have a GP of N and an ambiguous word has a GP of 1.

8.4 Experiments: Substitute Distribution Clustering

In this study, we predict the part of speech of a word in a given context based on its substitute distribution. This section details the choice of the data set, the vocabulary, and the estimation of substitute probabilities.

The choice of the clustering algorithm highly depends on the structure of the unsupervised tagging task. Therefore, instead of specifying each step of the algorithm in detail, we try to construct a general clustering framework that can be used on different kinds of induction tasks. The steps below can be modified or skipped to get better results on tasks other than the POS induction.

Algorithm

Optional. Take the average of the substitute vector instances of each word to construct word-based vectors.

1. Find the best similarity metric between two substitute distributions.

2. Apply dimensionality reduction to find low-dimensional representations.

3. Apply the clustering algorithm on low-dimensional vectors.

The optional step aims to constructs word-based representation of substitute representations which, by definition, assumes that all instances of a word will be in the same category.

The rest of this section is organized into subsections as follows: Section 8.4.1 details the test corpus and experimental settings used. Section 8.4.2 compares various similarity metrics. Section 8.4.3 summarizes the effect of dimensionality reduction of substitute distributions, and Section 8.4.4 compares the performance of clustering algorithms on low-dimensional substitute distributions. Section 8.4.5 suggests two method to construct word-based clusters, and Section 8.4.6 applies the findings of the previous sections to the whole

PTB corpus. Finally, Section 8.4.7 analyzes the clustering output with the highest MTO accuracy.

8.4.1 Experimental Settings

The first 24,020 tokens of the Penn Treebank (Marcus et al., 1999) Wall Street Journal Section 00 was used as the test corpus. The treebank uses 45 part-of-speech tags which is the set we used as the gold standard for comparison in our experiments. To compute substitute probabilities, we trained a language model using approximately 126 million tokens of the Wall Street Journal data (1987-1994) extracted from the CSR-III Text (Graff et al., 1995) (we excluded the test corpus). We used SRILM (Stolcke, 2002b) to build a 4-gram language model with Kneser-Ney discounting. Words that were observed less than 500 times in the LM training data were replaced by UNK tags, giving us a vocabulary size of 12,672. The perplexity of the 4-gram language model on the test corpus was 55.4, which is quite low as a result of using in-domain data and a small vocabulary.

We computed the 12,672-dimensional substitute distributions at each of the 24,020 positions in the test corpus, as specified in Section 2.3.

8.4.2 Distance Metrics

This step aims to determine the best similarity metric to judge vectors that belong to the same category ("similar") and vectors that belong to different categories ("distant"). The distance metrics we have considered in this thesis are listed in Table 8.1.

We represent each context with a sparse high-dimensional probability vector, called the substitute vector, as described in Section 2.3 section. To judge the merit of each distance metric, we obtained supervised baseline scores using leave-one-out cross validation and the weighted k-nearest-neighbor algorithm[2] on the gold tags of the PTB24K. The results are

[2]Neighbors were weighted using 1/distance. $k = 30$ was chosen empirically.

Cosine($\mathbf{p}, \mathbf{q}$)	=	$< \mathbf{p}, \mathbf{q} > /(\|\mathbf{p}\|_2 \|\mathbf{q}\|_2)$
Euclid($\mathbf{p}, \mathbf{q}$)	=	$\|\mathbf{p} - \mathbf{q}\|_2$
Manhattan($\mathbf{p}, \mathbf{q}$)	=	$\|\mathbf{p} - \mathbf{q}\|_1$
Maximum($\mathbf{p}, \mathbf{q}$)	=	$\|\mathbf{p} - \mathbf{q}\|_\infty$
KL2($\mathbf{p}, \mathbf{q}$)	=	$\sum_i p_i ln(p_i/q_i) + q_i ln(q_i/p_i)$
JS($\mathbf{p}, \mathbf{q}$)	=	$\sum_i p_i ln(p_i/m_i) + q_i ln(q_i/m_i)$
		where $m_i = 0.5(p_i + q_i)$

Table 8.1: Similarity metrics. JS is the Jensen-Shannon divergence, and KL2 is a symmetric implementation of Kullback-Leibler divergence. Bold lower case letters represent vectors.

listed in Table 8.2, and are sorted by score.

The entries with the log-prefix indicate a metric applied to the log of the probability vectors. Distance metrics on log-probability vectors performed poorly compared to their regular counterparts, indicating that differences in low-probability words are relatively unimportant and high-probability substitutes determine the syntactic category. The surprisingly good result achieved by the simple Maximum metric (which identifies the dimension with the largest difference between two vectors) also support this conclusion. The maximum score of .68 can be taken as a rough upper bound for an unsupervised learner using this space on the PTB24K corpus, because 32% of the instances are assigned to the wrong part of speech by the majority of their closest neighbors. We will discuss alternative ways to push this upper bound higher by enforcing a one-tag-per-word rule in Section 8.4.4.

8.4.3 Dimensionality Reduction

In this section, we investigate if there is a low-dimensional representation of the substitute vectors which still preserves the neighborhood information necessary to learn syntactic categories. We report experimental results on principal components analysis (PCA), Isomap

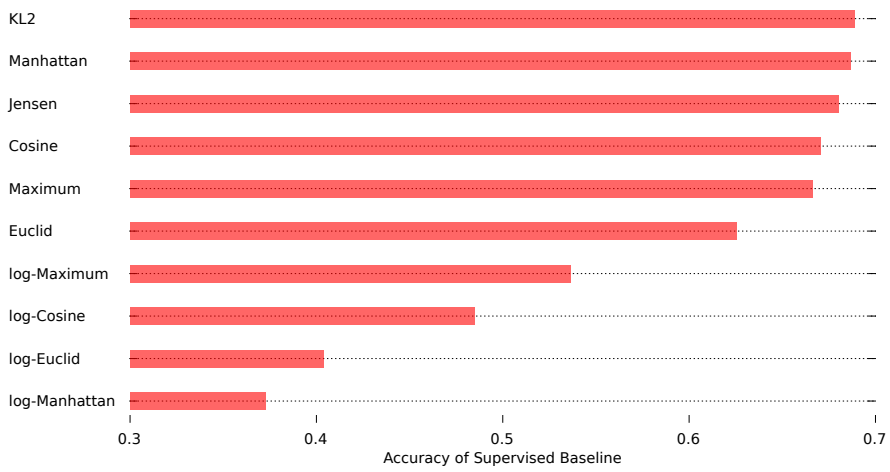


Figure 8.2: Supervised baseline scores with different distance metrics. Log-metric indicates that the metric is applied to the log of the probability vectors.

(Tenenbaum et al., 2000), locally linear embedding (LLE) (Roweis and Saul, 2000), and Laplacian eigenmaps (Belkin and Niyogi, 2003). Each dimensionality reduction algorithm tries to preserve certain aspects of the original vectors. PCA is a linear method that minimizes reconstruction error. Isomap tries to preserve distances as measured along a low-dimensional sub-manifold, assuming that the input vectors were sampled from the neighborhood of such a manifold. LLE most faithfully preserves the local linear structure of nearby input vectors. Laplacian eigenmaps most faithfully preserve proximity relations, mapping nearby inputs to nearby outputs.

We wanted to see how accuracy (determined by the k-nearest-neighbor supervised baseline, as in the previous section) changes as a function of the number of dimensions for each dimensionality reduction algorithm. For algorithms that require a distance matrix rather than raw input vectors, we used the Jensen-Shannon divergence which was judged the best by the experiments of the previous section. For graph-based methods we built neighbor-

hood graphs using 100 nearest neighbors. The low-dimensional output vectors were compared using the cosine distance metric for the supervised k-nearest-neighbor algorithm. Figure 8.3 plots the supervised baseline accuracy vs. the number of dimensions for each algorithm.

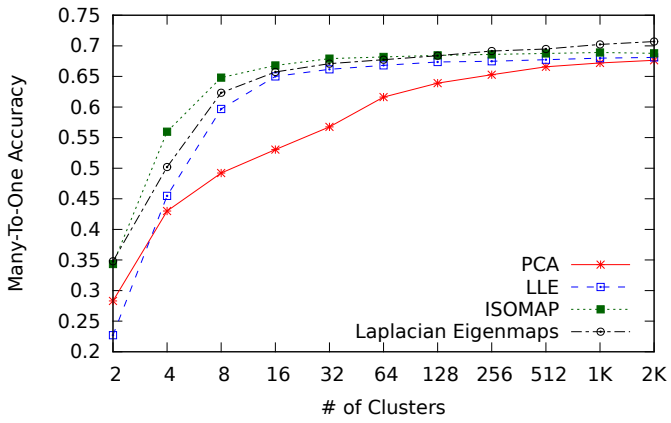


Figure 8.3: Supervised knn baselines for the four dimensionality reduction algorithms.

All of the graph-based algorithms (Isomap, LLE, and Laplacian eigenmaps) outperform PCA. They stay within 5% of their peak accuracy with as few as 16 dimensions. In fact Laplacian eigenmaps outperform the baseline with the original 12,672-dimensional vectors (68.95%) when allowed to retain more than about 250 dimensions. Spectral clustering uses the same transformation as the Laplacian eigenmaps algorithm, and we compare its performance to other clustering algorithms in the next section.

8.4.4 Clustering

In this section, we briefly describe and compare three clustering algorithms on the PTB24K. Hierarchical agglomerative clustering with complete linkage (HAC) starts with each instance in its own cluster, and iteratively combines the two closest groups (measured by

their most distant points) at each step (Manning et al., 2008). k-medoids minimizes sum of pairwise distances between each data-point and the exemplar at the center of its cluster (Kaufman and Rousseeuw, 2005). Spectral clustering uses the eigenvalues of the graph Laplacian $L = D^{-1/2}WD^{-1/2}$ to reduce the number of dimensions (similar to Laplacian eigenmaps), and uses simple k-means clustering on the resulting representation (Ng et al., 2002). All three algorithms accept the distance matrix consistng of the KL2 distance entries.

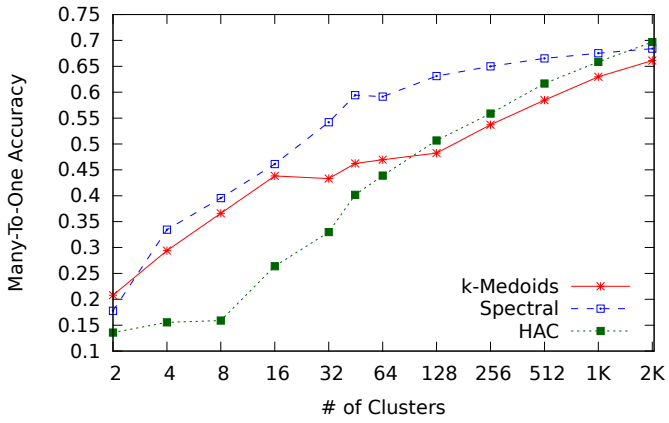


Figure 8.4: Many-to-one score for three clustering algorithms on the 45-tag PTB24K word corpus.

Figure 8.4 plots the many-to-one score versus the number of clusters for the three algorithms on the PTB24K. The many-to-one score naturally increases as we approach the one cluster per word limit. However, the evolution of the curves is quite informative. At the high end (more than 2000 clusters), HAC performs best with its conservative clusters, but its performance degrades fast as we reduce the number of clusters, because it cannot reverse the accumulated errors. At the low end (less than 16 clusters), k-medoids and spectral have similar performance. However, for the region of interest (between 16 to 2000 clusters), spectral clustering is clearly superior, with .5841 MTO accuracy at 45 clusters.

8.4.5 Word-base Clustering

We noted that the 45-cluster spectral clustering result described in the previous section assigned many more tags to each word than the gold standard. The tag perplexity of the gold standard 45-tag PTB24K word test corpus is 1.09, whereas the tag perplexity of the spectral clustering result is 2.76.

We experimented with two methods for reducing the number of tags assigned to each word: collapsing and word penalties. Collapsing enforces the one-tag-per-word constraint by re-tagging the corpus, whereas word penalties encourage it by increasing the distance between instances with different target words.

To collapse a given tag assignment for a corpus, we re-tag each word with its most frequent tag in the original assignment (we break ties randomly). This forcefully reduces the tag perplexity to 1, and removes any ambiguity. Collapsing improves the many-to-one accuracy by more than 10%, from .5841% to .6882%.

Interestingly, when we try to enforce the one-tag-per-word restriction before clustering (by giving the average substitute vector for each word type to spectral clustering), the results get worse (58.02% many-to-one accuracy). The information in individual instances seems to be necessary for good clusters to arise.

Word penalties include information about the target word in the distance metric. The substitute vectors and their associated KL2 distance metric carry no information about the target word, only its context. We used the following distance metric which increases the distance between instances with different target words:

$$D(i,j) = KL2(\vec{s_i}, \vec{s_j}) + \delta I(w_i \neq w_j)$$

Here $\vec{s_i}$ and $\vec{s_j}$ are the substitute vectors that correspond to the i^{th} and j^{th} instances, w_i and w_j are the i^{th} and j^{th} word-instances, δ is the regularization parameter, and I is the indicator function that gives 1 if the two words are different and 0 if they are the same.

Increasing δ decreases the tag perplexity, but the accuracy change is non-monotonic. At $\delta = 1$, the tag perplexity is 1.91, and the many-to-one accuracy increases from .5841% to 64.35%. This demonstrates that we can significantly increase the accuracy by including more information on the target word, yet without employing the full one-tag-per-word constraint.

8.4.6 Substitute Vector Clustering on the PTB

To demonstrate that the clustering algorithm can scale, we performed spectral clustering on the full PTB. We calculated the substitute vectors of each position in the PTB by using an efficient algorithm described in Section 8.4.1[3]. We used the Manhattan distance[4] to compute the vector similarities and achieve .5473 MTO accuracy.

To increase the sparsity of the solution we enforced a one-tag-per-word restriction by re-tagging each word with its most frequent tag in the original assignment (ties were broken randomly). One-tag-per-word re-tagging improves the MTO accuracy from .5473 to .6834.

The re-tagging experiment shows that the one-tag-per-word restriction improves the model performance in terms of the MTO accuracy by explicitly incorporating the word identity information into the model. In the next section, we suggest an alternative method that models the co-occurrence of words and their contexts (the latter represented by substitutes), and improves the results on the PTB.

8.4.7 Discussion

Figure 8.5 is the Hinton diagram showing the relationship between the most frequent tags and clusters found by the collapsed algorithm (.6834% many-to-one accuracy) on the PTB.

[3]The FASTSUBS algorithm of Section 2.3 calculates only the top 100 substitutes and their unnormalized probabilities, thus resulting in sparse substitute vectors. The new model is computationally more efficient than the one in this section, and it is more appropriate for large datasets like the PTB.

[4]As KL2 is undefined on sparse vectors, we use Manhattan whose performance is close to KL2 on both the PTB24K and PTB.

In this section we present a qualitative comparison of gold standard tags and discovered clusters.

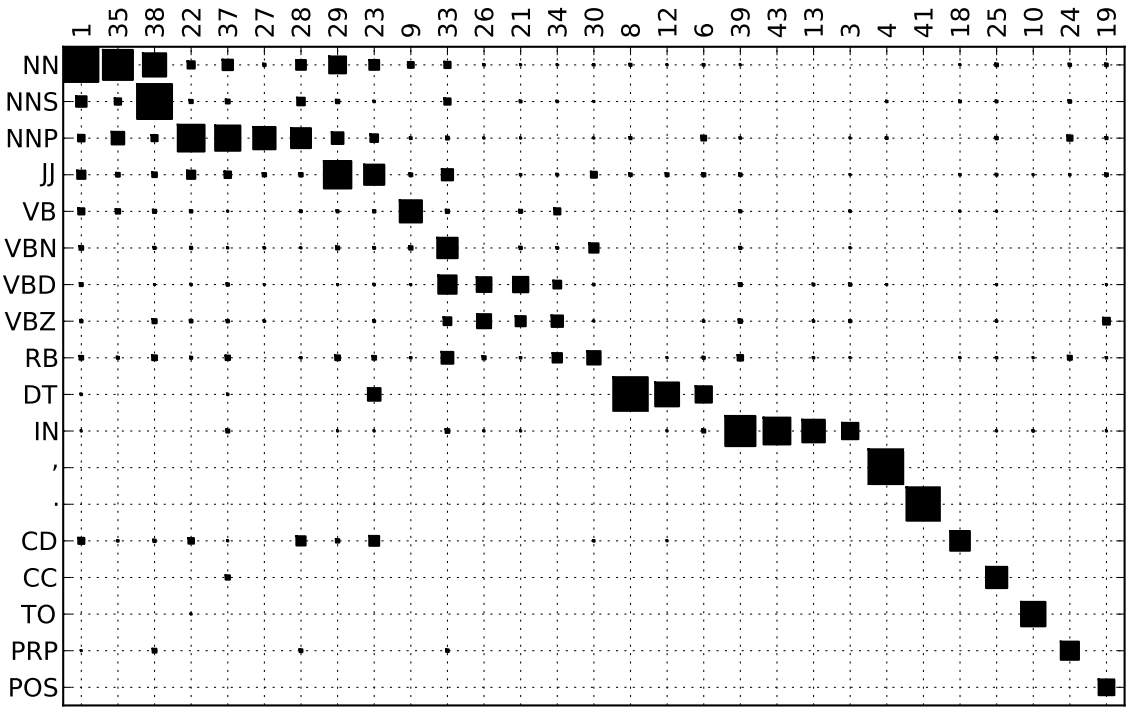


Figure 8.5: Hinton diagram of the most frequent tags (rows) and clusters (columns). Area of each square is proportional to the joint probability of the given tag and cluster.

Nouns and adjectives: Most nouns (NN*) are split between the clusters represented by the first seven columns of the Hinton graph, but not in the way Penn Treebank splits them. For example, the titles like *Mr.*, *Mrs.*, *Dr.* etc. are placed together in cluster 27 which does not exist as a separate class in the gold tags. Cluster 29 is the largest adjective (JJ) cluster; however, it also has noun members, probably resulting from the difficulty of separating noun-noun compounds and adjective modification.

Verbs and adverbs: Clusters 9 and 33 contain general verbs (VB*), but the verbs "be" (26), "say" (21), and "have" (34) have been split into their own clusters (whose count is shown within parentheses), presumably because they are not generally substitutable with the rest. Adverb (RB) is an amorphous class and the algorithm seems to have difficulty isolating it in a cluster.

Determiners and prepositions: We see a fairly clean separation of determiners (DT) and prepositions (IN) from other parts of speech, although each has been subdivided into further groups by the algorithm. For example, cluster 39 contains general prepositions but "of" (43), "in" (13), and "for" (3) are split into their own clusters. Determiners "the" (8), "a" (12), and capitalized "The"/"A" (6) are also split into their own clusters.

Closed-class items: Most closed-class items are cleanly separated into their own clusters, as shown in the lower right hand corner of the diagram.

8.5 Experiments: Co-occurrence Modeling

In this chapter, we model the co-occurrence of words and their contextual, orthographic, and morphological features in a high-dimensional Euclidean space that relates their joint probability to distance. Using these embeddings, we are able to build both word-based and instance-based clusters. The word-based model achieves the state-of-the-art results on 17 out of 19 corpora in 15 languages and comparable results on the 2 remaining, corpora. Although the instance-based model achieves lower scores than the word-based model, it handles the ambiguous words well, and achieves comparable or better performance than the best published systems on 15 of 19 corpora.

The rest of this section is organized into subsections as follows: Section 8.5.1 details the test corpus and experimental settings. Section 8.5.2 presents the performance of our word-based model and the sensitivity analysis of model parameters. Section 8.5.3 compares our

paradigmatic representation-based word model with its word-based syntagmatic counterpart. Section 8.5.4 explores morphological and orthographic features as additional sources of information for POS induction of words: its word- and instance-based system are the state-of-the-art in the field of POS induction. Section 8.5.5 presents the performance of our instance-based models and the sensitivity analysis of each model. Section 8.5.6 compares the performance of word-based and instance-based systems on ambiguous words. Finally, Section 8.5.7 extends the language and corpus coverage by applying the best performing model to 19 corpora in 15 languages.

8.5.1 Experimental Settings

To make a meaningful comparison with the previous works, the Wall Street Journal Section of the Penn Treebank (PTB) (Marcus et al., 1999) was used as the test corpus (1,173,766 tokens, 49,206 types) to be induced. The treebank uses 45 part-of-speech tags which is the set we used as the gold standard for comparison in our experiments.

To compute substitutes in a given context, we trained a language model using the ukWaC corpus ($\approx$ 2 billion tokens) constructed by crawling the ".uk" Internet domain (Ferraresi et al., 2008)[5]. We used SRILM (Stolcke, 2002b) to build a 4-gram language model with interpolated Kneser-Ney discounting. Words that were observed less than 2 times in the language model training data were replaced by <unk> tags, which gave us a vocabulary size of 4,254,946. The perplexity of the 4-gram language model on the PTB is 303, and the unknown word rate is 0.008. For computational efficiency, only the top 100 substitutes and their probabilities were computed for each position in the PTB, using the FASTSUBS algorithm (Yuret, 2012).

The experiments were run using the following default settings (unless otherwise stated): (i) each word was kept with its original capitalization, (ii) 90 substitutes were sampled per

[5]We use the Penn Treebank Tokenizer to make the training data compatible with PTB.

instance (iii) the learning rate parameters for S-CODE were set to $\varphi_0 = 50$, $\eta_0 = 0.2$, (iv) S-CODE convergence threshold, the log-likelihood difference between two consecutive iterations, was set to 0.001 , (v) the S-CODE dimensions and $\tilde{Z}$ were set to 25 and 0.166, respectively, (vi) a modified k-means algorithm with smart initialization was used (Arthur and Vassilvitskii, 2007), and (vi) the number of k-means restarts was set to 128 to improve clustering and reduce variance.

Each experiment was repeated 10 times with different random seeds, and the results are reported with standard errors in parentheses or error bars in graphs. Table 8.2 and 8.6 summarize all the results reported in this section and the ones we cite from the literature.

8.5.2 Word-based System

We cluster the S-CODE embedding of the target word ϕ_w using the k-means algorithm. The cluster-id for each ϕ_w becomes the predicted category of the corresponding word and all of its instances. Using the default settings, the many-to-one accuracy on the PTB is .7667 (.0056) and the V-measure is .6819 (.0029). These are the highest accuracies among the distributional models (see Table 8.2).

Distributional Models	MTO	VM
Lamar et al. (2010a)	.708	-
Brown et al. (1992)*	.678	.630
Goldwater et al. (2007b)*	.632	.562
Ganchev et al. (2010)*	.625	.548
Maron et al. (2010)	.688 (.0016)	-
Bigrams (Sec. 8.5.3)	.7319 (.0088)	.6554 (.0039)
Substitutes(Word-based) (Sec. 8.5.2)	.7667 (.0056)	.6819 (.0029)
Models with Additional Features	**MTO**	**VM**
Clark (2003)*	.712	.655
Christodoulopoulos et al. (2011)	.728	.661
Berg-Kirkpatrick et al. (2010)	.755	-
Christodoulopoulos et al. (2010)	.761	.688
Blunsom and Cohn (2011)	.775	.697
Word-based+Features (Sec. 8.5.4)	.8045(.0051)	.7228 (.0038)
Instance-based+Features (Sec. 8.5.5)	.7952 (.0030)	.6908 (.0027)

Table 8.2: Summary of results in terms of the MTO and VM scores. Standard errors, when available, are given in parentheses. Starred entries have been reported in the review paper (Christodoulopoulos et al., 2010). Distributional models use only the identity of the target word and its context. The models with features incorporate orthographic and morphological features. Instance-based models and the significantly best results are shown in bold.

To analyze the sensitivity of results to our specific parameter settings, we ran a number of experiments where each parameter was varied over a range of values.

The first plot of Figure 8.6 illustrates that the random-substitute result is fairly robust as long as the training algorithm can observe more than a few random substitutes per word.

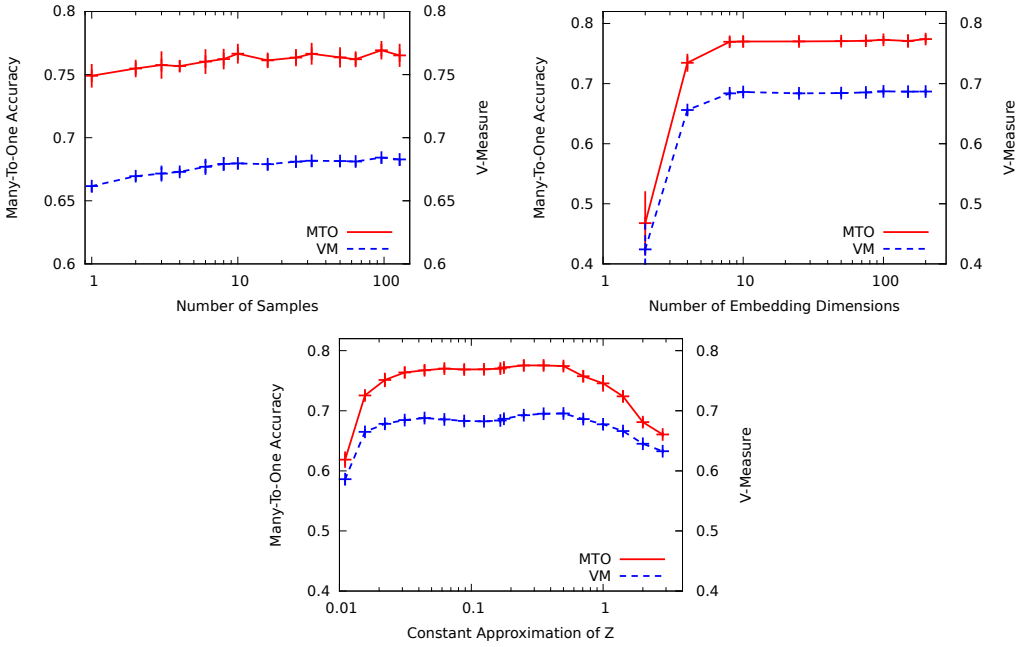


Figure 8.6: Sensitivity of instance-based POS induction performance on the PTB to (a) the number of sampled substitutes, (b) the number of embedding dimensions, (c) the constant approximation to the normalization constant $\tilde{Z}$.

The second plot of Figure 8.6 shows that at least 10 embedding dimensions are necessary to get within 1% of the best result, but there is no significant gain from using more than 25 dimensions. Finally, the third plot of Figure 8.6 shows that the constant $\tilde{Z}$ approximation can be varied within two orders of magnitude without a significant performance drop in the many-to-one score. For uniformly distributed points on a 25-dimensional sphere, the expected $Z \approx 0.146$. In the experiments in which we evaluated Z, we found the real Z to always be in the 0.140-0.170 range. When the constant $\tilde{Z}$, estimate is too small the attraction in Eq. 4.3 dominates the repulsion in Eq. 4.4, and all points tend to converge to the same location. When $\tilde{Z}$ is too high, it prevents meaningful clusters from coalescing.

We find our model with random substitute to be fairly robust with respect to different

parameter settings, and the resulting many-to-one score is significantly better than the state-of-the-art distributional models.

8.5.3 Paradigmatic vs Syntagmatic Representations of Word Context

To get a direct comparison between the paradigmatic and syntagmatic context representations, we feed 4 different co-occurrences into the S-CODE algorithm. The first model accepts (word (W) , right bigram (C)) pairs as the input, the second model accepts (word (W) , left bigram(C)) pairs as the input, the third model accepts (word (W) , concatenation of the left and right bigrams (C)) pairs (Mintz, 2003) as the input, and the final model accepts (words (W) , left bigram(C_1) and right bigram (C_2) tuples) (St Clair et al., 2010) as the input to the S-CODE. Finally, we cluster the word type embeddings (W) with k-means algorithm using the default settings.

To replicate the work of Maron et al. (2010), we use the model that uses (word (W) , right bigram (C)) pairs as the input. At the end, each word w in the vocabulary ends up with two points on the sphere: a ϕ_w point representing the behavior of w as the left word of a bigram, and a ψ_w point representing it as the right word. The two vectors for w are concatenated to create a 50-dimensional representation at the end. These 50-dimensional vectors are clustered using the k-means algorithm. Maron et al. (2010) report many-to-one scores of .6880 (.0016) for 45 clusters and .7150 (.0060) for 50 clusters (on the PTB). Using our default settings, the bigram model achieves .7319 (.0088) MTO and .6554 (.0039) VM accuracies. Table 8.3 summarizes all the results, and shows that the MTO and VM accuracies of the paradigmatic model are significantly higher than those of the syntagmatic models.

Input	MTO	VM
W (word) - C (right bigram)	.6625 (.0115)	.5809 (.0066)
W (word) - C (left bigram)	.6604 (.0054)	.5983 (.0028)
W (word) - C (left and right bigram concatenation)	.7268 (.0091)	.6416 (.0052)
W (word) - C_1, C_2 (left and right bigrams)	.7173 (.0061)	.6381 (.0032)
Maron et al. (2010)(replication)	.7319 (.0088)	.6554 (.0039)
W (word) - C (random substitutes)	**.7667 (.0056)**	**.6819 (.0029)**

Table 8.3: Summary of results in terms of the MTO and VM scores of the S-CODE algorithm when paradigmatic or syntagmatic representations are fed as input. Standard errors, when available, are given in parentheses. Results of the statistically best performing system are displayed in bold. We do not report the original results of Maron et al. (2010) since our replication achieves higher accuracies.

8.5.4 Morphological and Orthographic Features

Clark (2003) demonstrates that using morphological and orthographic features significantly improves part-of-speech induction with an HMM-based model. Section 8.1 describes a number of other approaches that show similar improvements. We integrate additional features with substitutes by using the modified S-CODE model described in Section 4.2.2.

Word-feature tuples might have null values arising from unobserved features. For example, in the case of POS induction, the word "**car**" has no morphological or orthographic features. Hence, all the elements of the tuple have null value, except the word type (w) and the context (c). We do not perform any pull or push updates on embeddings during the gradient search if the corresponding $f^{(k)}$ is null[6].

The orthographic features we used are similar to the ones in (Berg-Kirkpatrick et al.,

[6]In general, w and c represents the word type and context, so we assume that they are always observed.

2010) but with the following minor modifications:

- Initial-Capital: this feature is generated for capitalized words with the exception of sentence initial words.
- Number: this feature is generated when the token starts with a digit.
- Contains-Hyphen: this feature is generated for lowercase words with an internal hyphen.
- Initial-Apostrophe: this feature is generated for tokens that start with an apostrophe.

We generated morphological features using the unsupervised algorithm Morfessor (Creutz and Lagus, 2005). Morfessor was trained on the PTB using default settings, and a perplexity threshold of 300. We are concerned only with the splits tagged as *suffix* in the Morfessor output, and allow at most one morphological feature per word[7]. Table 8.4 presents the co-occurrence tuples of the example sentence after incorporating the orthographic and morphological features.

[7]Splits are concatenated if there are consecutive suffix parts at the end of a word.

Word	Context	Morphology	Initial Capital	Number	Contains Hypen	Initial Apostrophe
W:Pierre	*C:Mr.*		*F:IC*			
W:Vinken	*C:<unk>*		*F:IC*			
W:,	*C:,*					
W:61	*C:48*			*F:N*		
W:years	*C:years*	*F:s*				
W:old	*C:old*					
W:join	*C:head*					
W:the	*C:its*					
W:board	*C:company*					
W:as	*C:as*					
W:a	*C:a*					
W:nonexecutive	*C:non-executive*					
W:director	*C:chairman*	*F:or*				
W:Nov.	*C:May*		*F:IC*			
W:29	*C:9*			*F:N*		
W.	*C:.*					

Table 8.4: The words of input sentence *"Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29 ."* is represented with their substitutes and features. Words in the left column represent the target word, words in the second column represent the context, and tokens in the remaining columns are the features of the correponding target word. Features without values are unobserved, and are therefore set to null.

Using the training settings of the previous section, the addition of morphological and orthographic features increased the many-to-one score of the random-substitute model to .8045(.0051) and the V-measure to .7228 (.0038). Both these results improve the state-of-

the-art in part-of-speech induction significantly as seen in Table 8.2.

8.5.5 Instance-based System

In the previous section, we group words rather than word instances by clustering the word embeddings and ignoring their substitute embeddings. In the present section, we remove the *one-tag-per-word* assumption, and group the word instances by incorporating their substitute embeddings. We construct two different instance-based systems according to the representations defined in Section 4.2.3. The first system represents each instance with the average of its substitute embeddings $\overline{\psi_c}$, and the second one represents each instance with the concatenation of the target word embedding (ϕ_w) and the average of its substitute embeddings ($\overline{\psi_c}$). The first (respectively, second) system clusters the resulting $\overline{\psi_c}$ ($\phi_w \oplus \overline{\psi_c}$) vectors using the k-means algorithm, and the cluster-id for each $\overline{\psi_c}$ ($\phi_w \oplus \overline{\psi_c}$) becomes the predicted category of the corresponding word instance. Using the default settings with orthographic and morphological features, the first system achieves .7346 (.0102) MTO and .6468 (.0081) VM scores while the second one achieves .7952 (.0030) MTO and .6908 (.0027) VM scores. In the rest of the thesis, we refer to the model that represents each instance with $\phi_w \oplus \overline{\psi_c}$ as the *instance-based* model, and perform analysis and comparisons using only this model.

To analyze the sensitivity of this result to our specific parameter settings, we ran a number of experiments in which each parameter was varied over a range of values.

The first plot in Figure 8.7 illustrates that the result is fairly robust with respect to the number of random substitutes sampled for each target word instance, as long as the training algorithm can observe more than a few random substitutes per word. The second plot shows that at least 10 embedding dimensions are necessary to get within 1% of the best result, but there is no significant gain from using more than 25 dimensions. The third plot shows that the constant $\tilde{Z}$ approximation can be varied within two orders of magnitude without

any significant performance drop in the scores. For uniformly distributed points on a 25-dimensional sphere, the expected Z is 0.146. In the experiments in which we evaluated Z, we found the real Z to always be in the 0.140-0.170 range. When the constant $\tilde{Z}$ estimate is too small, all points tend to converge to the same location. When $\tilde{Z}$ is too high, it prevents meaningful clusters from coalescing.

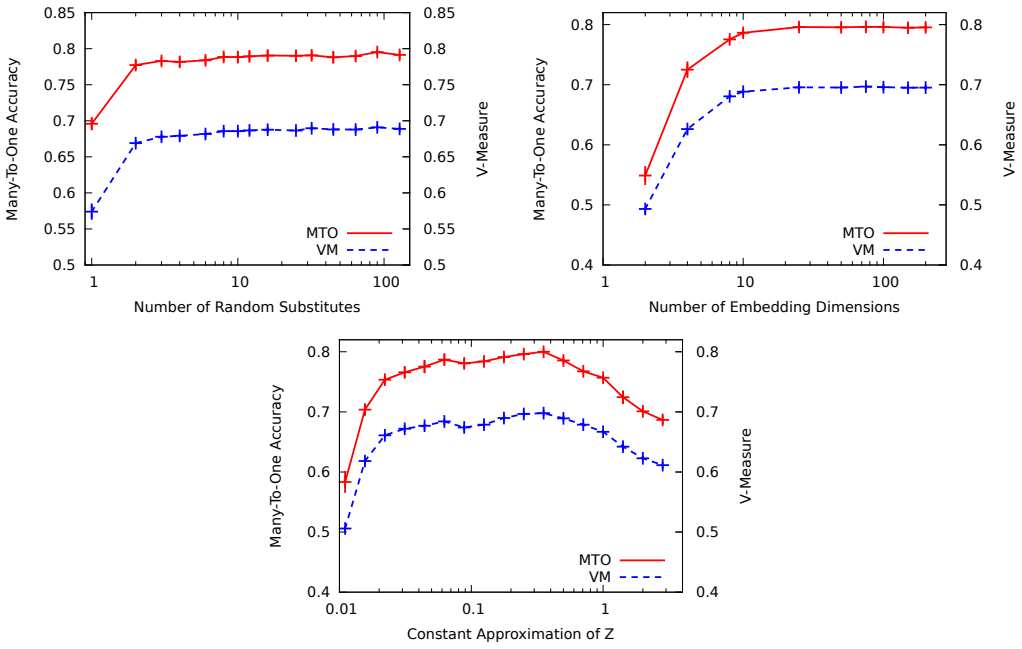


Figure 8.7: Sensitivity of instance-based POS induction performance on the PTB to (a) the number of sampled substitutes, (b) the number of embedding dimensions, (c) the constant approximation to the normalization constant $\tilde{Z}$.

8.5.6 Word vs. Instance-Based Induction

We compare the output of a word-based model in Section 8.5.2, and two instance-based POS induction systems in Section 8.5.5. All of the models that are analyzed in this section incorporate both orthographic and morphological features.

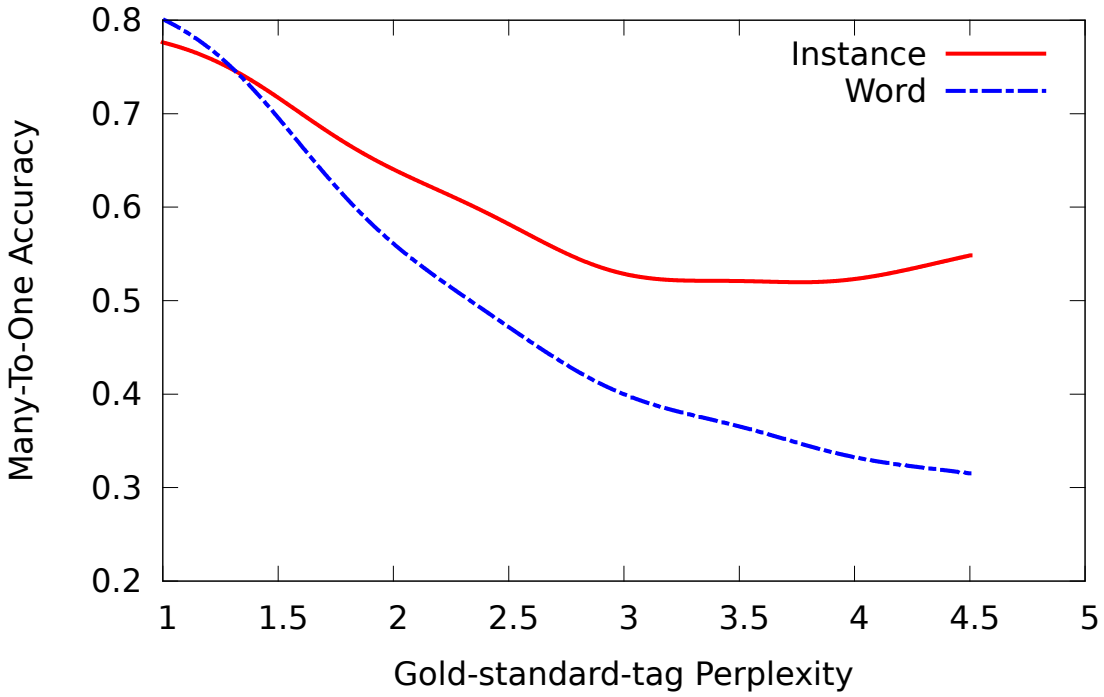


Figure 8.8: Regression lines for the word- and instance-based models on the MTO accuracy vs *GP* plot for the PTB.

Figure 8.8 plots the gold-tag perplexity versus the smoothed MTO accuracy for the word-based and the instance-based POS induction system. To compose the plot, we find the best mapping from the induced clusters to the gold-standard tags, then we compute the MTO accuracy for each word using this mapping, and plot the MTO as a function of the word's GP. We use the Nadaraya-Watson kernel regression estimate (Nadaraya, 1964; Watson, 1964) with normal kernel of bandwidth 1.0 to obtain smooth regression lines.

Figure 8.8 shows that the performance of the instance-based induction model does not degrade as much as that of the word-based model when the ambiguity of the words increase. However, only 14.94% of the instances in the PTB consists of words with GP greater than 1.5, and 45.71% consists of words with GP exactly 1. Thus, the overall accuracy numbers

do not adequately reflect the improvement on highly ambiguous words.

8.5.7 Multilingual Experiments

We performed experiments with a range of languages and three different feature configurations to establish both the robustness of our model across languages and to observe the effects of different features. Following Christodoulopoulos et al. (2011), in addition to the PTB we extend our experiments to 8 languages from MULTEXT-East (Bulgarian, Czech, English, Estonian, Hungarian, Romanian, Slovene and Serbian) (Erjavec, 2004) and 10 languages from the CoNLL-X shared task (Bulgarian, Czech, Danish, Dutch, German, Portuguese, Slovene, Spanish, Swedish and Turkish) (Buchholz and Marsi, 2006).

To sample substitutes, we trained language models of Bulgarian, Czech, Estonian, Romanian, Danish, German, Portuguese, Spanish, Swedish and Turkish with their corresponding TenTen corpora (Jakubíček et al., 2013), and Hungarian, Slovene and Serbian with their corresponding Wikipedia dump files[8]. Serbian shares a common basis with Croatian and Bosnian therefore we trained 3 different language models using Wikipedia dump files of Serbian together with these two languages and measured the perplexities on the MULTEXT-East Serbian corpus. We chose the Croatian language model since it achieved the lowest perplexity score and unknown word ratio on the MULTEXT-East Serbian corpus. We use the ukWaC corpus (Ferraresi et al., 2008) to train English language models and WaC corpus (Kilgarriff et al., 2010) to train Dutch language models. We trained the language model of all languages with the tool and default parameters detailed in Section 8.5.1. Altering these parameters did not significantly improve the perplexity. Table 8.5 summarizes the language model related statistics and perplexity scores that vary across the languages.

We used the default settings in Section 8.5.1 and incorporated only the orthographic

[8]The latest Wikipedia dump files are freely available at `http://dumps.wikimedia.org/` and the text in the dump files can be extracted using WP2TXT (`http://wp2txt.rubyforge.org/`)

features[9]. Extracting unsupervised morphological features for languages with different characteristics would be of great value, but it is beyond the scope of this thesis. For each language, the number of induced clusters is set to the number of tags in the gold-set. To perform meaningful comparisons with the previous work, we train and evaluate our models on the training section of MULTEXT-East[10] and CONLL-X languages (Lee et al., 2010).

[9] As all corpora (except German, Spanish and Swedish) label the punctuation marks with the same gold-tag, we add an extra *punctuation* feature for those languages.

[10] As the languages of MULTEXT-East corpora do not tag the punctuations, we add an extra tag for punctuation to the tag-set of these languages.

	Language	Tags	Best Published	Instance Based	Word Based
WSJ	English	45	.775 / .697‡	**.7952** / .6908	**.8045 / .7224**
MULTEXT-East	Bulgarian	12+1	.665 / .556⋆	.6644 / .5126	**.7008 / .5802**
	Czech	12+1	.642 / .539⋆	.7051 / .5109	**.7263 / .5487**
	English	12+1	.733 / .633⋆	**.8352** / .6612	**.8396 / .6914**
	Estonian	11+1	.644 / **.533**⋆	.6425 / .4569	**.6764** / .5092
	Hungarian	12+1	.682 / **.548**⋆	.6474 / .4588	**.6988** / .5202
	Romanian	14+1	.611 / .523⋆	.6600 / .5287	**.6808 / .5567**
	Slovene	12+1	.679 / **.567**⋆	.6666/ .4513	**.6996** / .5037
	Serbian	12+1	.641 / **.510**†	.5936 / .4021	.6372/ .4611
CoNLL-X Shared Task	Bulgarian	54	.704 / .596†	.7509 / .5827	**.7679 / .6179**
	Czech	12	.701‡ / .484⋆	.7006 / .4864	**.7354 / .5629**
	Danish	25	.761‡ / .591⋆	.7606 / .5840	**.7766 / .6208**
	Dutch	13	.711‡ / .547⋆	.7120 / .5368	**.7418 / .6030**
	German	54	.744⋆ / .630†	.7495 / .6185	**.7850 / .6634**
	Portuguese	22	.785‡ / .639⋆	.7820 / .6078	**.8072 / .6567**
	Slovene	29	.642⋆ / **.539**†	.6380 / .4698	**. 6639** / .5192
	Spanish	47	.788‡ / .632⋆	.7527 / .6017	.7925 / **.6505**
	Swedish	41	.682 / .589†	.6814 / .5463	**.7052** / .5812
	Turkish	30	.628 / .408⋆	.6373 / .4012	**.6602 / .4319**

Table 8.6: The MTO and VM scores on 19 corpora in 15 languages together with number of induced clusters. Statistically significant results shown in bold ($p < 0.05$). MULTEXT-East corpora do not tag the punctuation marks, thus we add an extra tag for punctuation and represent it with "+1".

For each language we report the results of instance and word-based models. Table 8.6 presents the performance of our instance- and word-based models on 19 corpora in 15 languages together with the corresponding syntagmatic baseline and the best published results from $^{\diamond}$(Yatbaz et al., 2012), ‡(Blunsom and Cohn, 2011), *(Christodoulopoulos et al., 2011) and †(Clark, 2003). All of the state-of-the-art systems in Table 8.6 are word-based and incorporate morphological features.

Our word-based model achieves the state-of-the-art MTO accuracies on all languages except Serbian and Spanish. The MTO accuracies of our model on Serbian and Spanish are comparable with the best published results. Our word-based model scores lower VM accuracies then the best published results on Estonian, Hungarian, Slovene and Serbian.

Our instance-based results are lower than the word-based results which is mainly due to the absence of one-tag-per-word assumption. However instance-based model still performs comparable to or significantly better than the best published system on most languages. We show significant improvements on MULTEXT-East Czech, Romanian, CoNLL-X Bulgarian and Turkish, and comparable results on MULTEXT-East Bulgarian, Estonian, CoNLL-X Czech, Danish, Dutch, German, Slovene, Portuguese and Swedish in terms of the MTO score. Our model achieves the comparable MTO with the word-based model on MULTEXT-East English and the PTB. Our instance-based model has statistically lower VM scores in spite of good MTO scores on 11 corpora which is discussed in Section 8.5.8.

Serbian and Slovene language models are trained on the Wikipedia corpora. TenTen, ukWaC and WaC corpora are cleaner and tokenized better compared to the Wikipedia corpora. These corpora also have larger vocabulary sizes and lower out-of-vocabulary rates (see Table 8.5). Thus language models trained on these corpora have much lower perplexities and generate better substitutes than the Wikipedia-based models.

8.5.8 *Discussion*

In this section we perform further analysis on the clustering output of our best performing models, and indicate the possible reasons for the low VM scores compared to the MTO ones. To illustrate how words are distributed in the induced clusters, we compare the output of our model with gold-tags of the PTB by using a Hinton diagram.

Analysis of Word-based Model

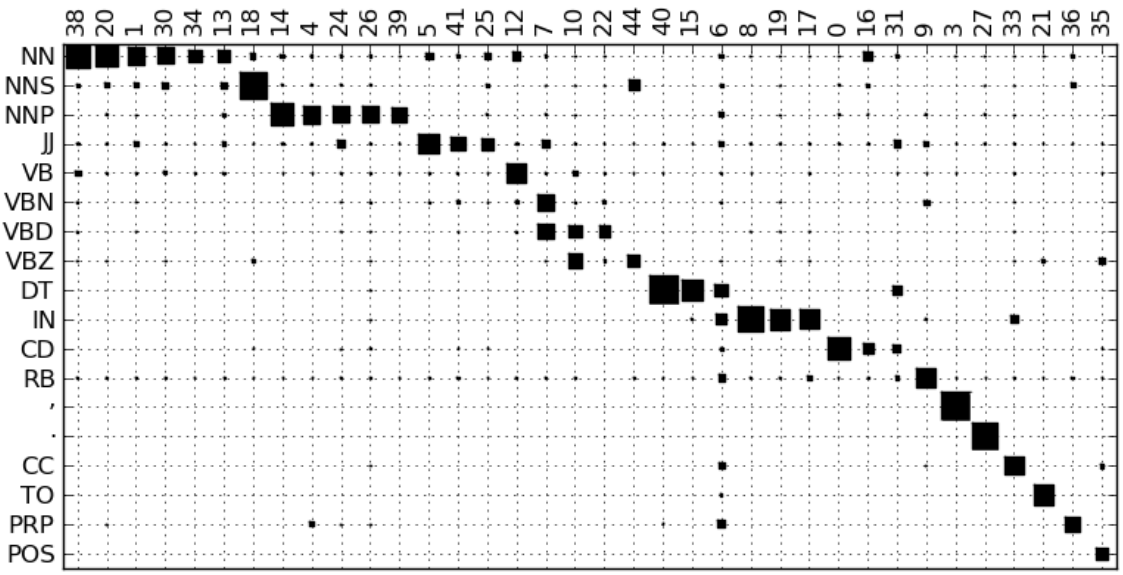


Figure 8.9: Each row corresponds to a gold tag, and each column is an induced tag in the Hinton diagram above. The area of each square is proportional to the joint probability of the given tag and cluster.

Figure 8.9 is the Hinton diagram for the PTB, showing the relationship between the most frequent tags and clusters from the experiment in Section 8.5.4. In general, the errors seem to be the lack of completeness (multiple large entries in a row), rather than the lack of

homogeneity (multiple large entries in a column). The algorithm tends to split large word classes into several clusters. Some examples are:

- Titles like Mr., Mrs., and Dr. are split from the rest of the proper nouns in cluster (39).
- Auxiliary verbs (10) and the verb "say" (22) have been split from the general verb clusters (12) and (7).
- Determiners "the" (40), "a" (15), and capitalized "The"/ "A" (6) have been split into their own clusters.
- Prepositions "of" (19), and "by", "at" (17) have been split from the general preposition cluster (8).

Nevertheless, there are also some homogeneity errors:

- The adjective cluster (5) also contains some noun members, probably caused by the difficulty of separating noun-noun compounds from adjective modification.
- Cluster (6) contains capitalized words that span a number of categories.

Most closed-class items are cleanly separated into their own clusters, as seen in the lower right hand corner of the diagram.

The completeness errors are not surprising, given that the words that have been split are not generally substitutable with the other members of their gold-tag set category. Thus it can be argued that metrics such as MTO that emphasize homogeneity are more appropriate in this context than metrics such as VM that average homogeneity and completeness as long as the number of clusters is controlled.

The gold-tags of PTB, on the other hand, do not always respect whether words with the same tag are substitutable for one another. Freudenthal et al. (2005) argues, from the

perspective of language acquisition by children, that the standard linguistic definition of syntactic groups requires the substitutability of words in a syntactic category. Word pairs that are placed in the same category in the PTB, such as "Mr." and "Friday", "be" and "run", "not" and "gladly", "of" and "into" are clearly not generally substitutable.

The completeness errors become more noticeable on languages with coarse tag-sets. Thus our models perform worse than the best published models on 4 of MULTEXT-East languages in terms of VM scores while achieving the state-of-the-art or comparable MTO scores on the same languages, as shown on Table 8.6. On CONLL-X languages the effect of completeness errors is less noticeable since all languages except Czech and Dutch have fine grained tag-sets.

Analysis of Instance-based Model

In this section, we perform further analysis on the clustering output of our instance-based model. The example below illustrates the advantage of the instance-based approach:

(1) . . . it will also **offer** buyers the option . . .

Substitutes: give, help, attract

(2) The **offer** is being launched . . .

Substitutes: campaign, project, scheme

The word **offer** is a *verb* in the first sentence and a *noun* in the second one. The word-based model cannot distinguish the different occurrences of such words, and assigns all instances of **offer** to the same cluster. On the other hand, the substitutes of *offer* in both sentences can disambiguate the correct category of the corresponding occurrences. In our experiments, our instance-based representation distinguishes the instances of **offer** as *noun* (cluster 26 and 12) and *verb* (cluster 35), as shown in Figure 8.10.

To illustrate how words are distributed in the induced clusters, we compare the most frequent clusters of our model in Section 8.5.5 with the most frequent gold-tags of PTB in Figure 8.10.

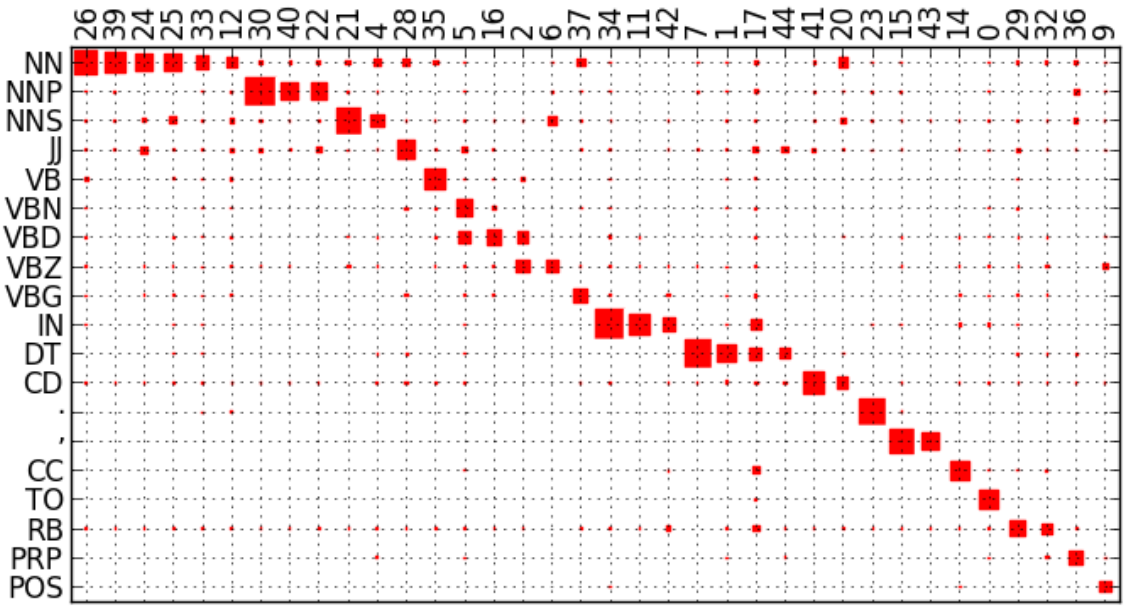


Figure 8.10: Each row corresponds to a gold tag, and each column is an induced tag in the Hinton diagram above. The area of each square is proportional to the joint probability of the given tag and cluster.

The low VM performance of our instance-based model when compared to the state-of-the-art word-based systems on some languages is attributable to the completeness part of the VM score. The Hinton diagram in Figure 8.10 shows that large gold-tag groups are split into several clusters because of the substitutability of words in that particular cluster (rows of the Hinton diagram). A noteworthy example is that instance-based model splits the punctuation mark (,) class of PTB into the clusters 15 and 43 because of the different usage patterns. The majority of the (,) instances in cluster 15 are used in relative clauses, reported speech clauses or conjunctions, while cluster 43 generally consists of (,) instances that are used in non-essential clauses (ex: Time, the largest newsweekly, . . .). On the other hand, the word-based system assigns all the instances of the (,) to a single cluster.

8.6 Conclusion

Our main contributions can be summarized as follows:

- We introduced substitute vectors as paradigmatic representations of word context and demonstrated their use in unsupervised part of speech induction on 19 corpora in 15 languages.

- We demonstrated that using paradigmatic representations of word context and modeling co-occurrences of word and context types with the S-CODE learning framework give superior results when compared to a syntagmatic bi-gram model.

- We extended the S-CODE framework to incorporate morphological and orthographic features and improved the state-of-the-art many-to-one accuracy in unsupervised part of speech induction on 17 out of 19 corpora.

- We introduced an instance based POS induction system that can handle ambiguous words and is competitive with the word-based systems in overall accuracy.

- All our code and data, including the substitute vectors for the PTB, MULTEXT-East and CoNLL-X shared task corpora are available at the authors' website at `xxx.xxx.xxx`.

		Language Model			Test Corpus			
	Language	Source	Instance Count	Word Count	Instance Count	Word Count	Unknown Word	Perplexity (ppl)
WSJ	English	ukWaC	2,303,225,131	4,254,946	1,173,766	49,206	0.0081	303.477
MULTEXT-East	Bulgarian	TenTen	849,023,297	1,965,178	101,173	16,353	.0151	295.704
	Czech	TenTen	1,791,613,805	4,758,807	100,368	19,121	.0038	294.022
	English	ukWaC	2,303,225,131	4,254,946	118,424	9,774	.0046	143.451
	Estonian	TenTen	330,671,558	2,526,585	94,898	17,847	.0166	477.805
	Hungarian	Wikipedia	66,069,788	1,065,897	98,426	20,323	.0449	654.086
	Romanian	TenTen	53,456,650	310,366	118,328	15,192	.0070	126.596
	Slovene	Wikipedia	18,969,864	363,251	112,278	17,873	.0389	648.347
	Serbian	Wikipedia	17,129,679	368,778	108,809	18,113	.0580	804.962
CoNLL-X Shared Task	Bulgarian	TenTen	849,023,297	1,965,178	190,217	32,439	.0196	168.592
	Czech	TenTen	1,791,613,805	4,758,807	1,249,408	130,208	.0050	476.434
	Danish	TenTen	1,857,746,600	5,304,957	94,386	18356	.0218	185.325
	Dutch	WaC	127,580,512	774,965	195,069	28,493	.0465	261.709
	German	TenTen	1,810,802,875	6,513,804	699,610	72,326	.0227	417.676
	Portuguese	TenTen	3,267,166,367	3,434,834	206,678	28,932	.0493	364.92
	Slovene	Wikipedia	18,969,864	363,251	28,750	7,128	.0414	596.678
	Spanish	TenTen	2,445,878,830	3,067,682	89,334	16,458	.0343	193.94
	Swedish	TenTen	113,975,094	926,875	191,467	20,057	.0179	288.16
	Turkish	TenTen	1,804,606,896	5,308,241	47,605	17,563	.0550	600.632

Table 8.5: Language model corpos and test corpus statistics are presented.

Chapter 9

CONCLUSION

This thesis represents the context by the distribution of substitute words, and proposes models with different characteristics to incorporate this new representation. The main contributions of the thesis are the following:

- It formulates a new paradigmatic representation of context in Chapter 3.
- It presents a method to calculate the substitute distributions of a given context (see Section 2.3). It also presents a discretized context representation by sampling substitutes (with replacement) from the substitute distributions (see Section 2.4).
- To incorporate substitute distributions, 4 different tagging models with different characteristics are introduced in Chapter 4.
- Section 8.4 analyzes the performance of substitute distributions according to different distance metrics, dimensionality reduction methods, and and clustering algorithms, on the unsupervised part-of-speech induction task of English.
- The word-based co-occurrence model achieves the state-of-the-art many-to-one accuracies in the unsupervised part-of-speech induction task of 19 corpora in 15 languages.
- The instance-based co-occurrence model achieves results that are competitive in

overall accuracy with the best-published systems on 15 out of 19 corpora in 15 languages.

- The probabilistic voting algorithm is applied to the morphological disambiguation task of Turkish, and achieves 64.5% accuracy on ambiguous words.

- The word-tag dictionary reduction method (see Section 4.4.1) improves the accuracy of the HMM-EM model (with a word-tag dictionary), and achieves 92.25% and 92.47% word tagging accuracies on the 24K- and 48K-word corpora, respectively. Both results are the highest reported results on the corresponding datasets (see Section 5.4).

- The data-enhanced Viterbi algorithm (see Section 4.4.2) significantly improves the prediction accuracies of the HMM-EM model (with a word-tag dictionary) in all of the corpora, and achieves %2.6 error accuracy gain in the worst case and %5.6 in the best case (see Section 5.5).

- The noisy channel model (see Section 4.5) is applied to the word-sense disambiguation task of English when a word-tag distribution is available. This model consistently outperforms any previously reported unsupervised model results, and achieves performance comparable to the best supervised model results.

- From the exploration of semantic classes at various levels of granularity, it is shown that the relationship between granularity and fine-grained accuracy is complex.

Appendix A: Solutions for $\Pr(T|C)$

To solve for $\Pr(T|C)$ using $\Pr(W|C)$ and $\Pr(W|T)$, we represent the first two as vectors: $t_j = \Pr(T = j|C = k)$ and $w_i = \Pr(W = i|C = k)$, and the last one as a matrix: $\mathbf{WT}_{ij} = \Pr(W = i|T = j)$. Our problem becomes finding a solution to the linear equation $w = \mathbf{WT} \times t$. Using the Moore–Penrose pseudoinverse, $\mathbf{WT}^+$, we find a solution $t = \mathbf{WT}^+ \times w$. This solution minimizes the distance $|\mathbf{WT} \times t - w|$. There are two potential problems with this pseudoinverse solution. First, it may violate the non-negativity and normalization constraints of a probability distribution. Second, a maximum likelihood estimate should minimize the cross entropy between $\mathbf{WT} \times t$ and w, not the Euclidean distance. We addressed the normalization problem using a constrained linear solver and the cross entropy problem using numerical optimization. However, our experiments showed the difference in WSD performance to be less than 1% in each case. The pseudoinverse solution, $t = \mathbf{WT}^+ \times w$, can be computed fast and works well in practice so this is the solution that is used in all our experiments.

BIBLIOGRAPHY

Abend, O., Reichart, R., and Rappoport, A. (2010). Improved unsupervised pos induction through prototype discovery. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1298–1307, Stroudsburg, PA, USA. Association for Computational Linguistics. 102

Adler, M. and Elhadad, M. (2006). An unsupervised morpheme-based hmm for hebrew morphological disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 665–672. 73

Agirre, E., Màrquez, L., and Wicentowski, R., editors (2007). *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic. Association for Computational Linguistics. 81

Agirre, E. and Martinez, D. (2004). Unsupervised WSD based on automatically retrieved examples: The importance of bias. In Lin, D. and Wu, D., editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 25–32, Barcelona, Spain. Association for Computational Linguistics. 82

Ambridge, B. and Lieven, E. (2011). *Child Language Acquisition: Contrasting Theoretical Approaches*, chapter 6.1. Cambridge University Press. 97, 99

Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics. 116

Bahl, L. R., Jelinek, F., and Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):179–190. 48, 81

Banko, M. and Brill, E. (2001). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, Toulouse, France. Association for Computational Linguistics. 12

Banko, M. and Moore, R. C. (2004). Part of speech tagging in context. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 556, Morristown, NJ, USA. Association for Computational Linguistics. 52

Baum, L. (1972). An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. *Inequalities*, 3(1):1–8. 43

Belkin, M. and Niyogi, P. (2003). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396. 108

Berg-Kirkpatrick, T., Bouchard-Côté, A., DeNero, J., and Klein, D. (2010). Painless unsupervised learning with features. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 582–590, Los Angeles, California. Association for Computational Linguistics. 120

Berg-Kirkpatrick, T. and Klein, D. (2010). Phylogenetic grammar induction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1288–1297, Uppsala, Sweden. Association for Computational Linguistics. 27, 98, 102, 117

Biemann, C. (2006). Unsupervised part-of-speech tagging employing efficient graph clustering. In *Proceedings of the 21st International Conference on computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 7–12. Association for Computational Linguistics. 101

Bilmes, J. and Zweig, G. (2002). The Graphical Models Toolkit: An open source software system for speech and time-series processing. In *IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS SPEECH AND SIGNAL PROCESSING*, volume 4, pages 3916–3919. Citeseer. 55

Blunsom, P. and Cohn, T. (2011). A hierarchical pitman-yor process hmm for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 865–874, Portland, Oregon, USA. Association for Computational Linguistics. 27, 97, 102, 117, 129

Brants, T. and Franz, A. (2006a). Web 1T 5-gram version 1. Linguistic Data Consortium, Philadelphia. LDC2006T13. 22, 80

Brants, T. and Franz, A. (2006b). Web 1T 5-gram Version 1. *Linguistic Data Consortium, Philadelphia*. 55

Brill, E. and Moore, R. C. (2000). An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong. Association for Computational Linguistics. 81

Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85. 48, 81

Brown, P. F., deSouza, P. V., Mercer, R. L., Pietra, V. J. D., and Lai, J. C. (1992). Class-

based n-gram models of natural language. *Comput. Linguist.*, 18:467–479. 27, 97, 98, 100, 101, 117

Buchholz, S. and Marsi, E. (2006). Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 149–164, Stroudsburg, PA, USA. Association for Computational Linguistics. 126

Chandler, D. (2007). *Semiotics: the basics*. The Basics Series. Routledge. 5, 13

Chen, S. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. *Computer Speech and Language*, 13(4):359–394. 20, 81

Chklovski, T. and Mihalcea, R. (2003). Exploiting agreement and disagreement of human annotators for word sense disambiguation. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, Borovetz, Bulgaria. 81

Christodoulopoulos, C., Goldwater, S., and Steedman, M. (2010). Two decades of unsupervised pos induction: how far have we come? In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 575–584, Stroudsburg, PA, USA. Association for Computational Linguistics. 10, 98, 99, 101, 102, 103, 117

Christodoulopoulos, C., Goldwater, S., and Steedman, M. (2011). A bayesian mixture model for pos induction using multiple features. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 638–647, Edinburgh, Scotland, UK. Association for Computational Linguistics. 97, 102, 117, 126, 129

Ciaramita, M. and Altun, Y. (2006). Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on*

Empirical Methods in Natural Language Processing, pages 594–602, Sydney, Australia. Association for Computational Linguistics. 81

Clark, A. (2003). Combining distributional and morphological information for part of speech induction. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1*, EACL '03, pages 59–66, Stroudsburg, PA, USA. Association for Computational Linguistics. 97, 101, 102, 117, 120, 129

Cotton, S., Edmonds, P., Kilgarriff, A., and Palmer, M., editors (2001). *SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*, Toulouse, France. 81

Crestan, E., El-Bèze, M., and De Loupy, C. (2001). Improving WSD with Multi-Level View of Context Monitored by Similarity Measure. In *Proceedings of SENSEVAL-2: Second International Workshop on Evaluating Word Sense Disambiguation Systems*. 81, 89

Creutz, M. and Lagus, K. (2005). Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of AKRR'05, International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*, pages 106–113, Espoo, Finland. 121

Daume III, H. and Marcu, D. (2002). A noisy-channel model for document compression. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 449–456, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics. 81

Dolan, W. (1994). Word sense ambiguation: clustering related senses. *Proceedings of the 15th conference on Computational linguistics*, pages 05–09. 81

Echihabi, A. and Marcu, D. (2003). A noisy-channel approach to question answering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 16–23, Sapporo, Japan. Association for Computational Linguistics. 81

Erjavec, T. (2004). MULTEXT-east version 3: Multilingual morphosyntactic specifications, lexicons and corpora. In *Fourth International Conference on Language Resources and Evaluation, LREC'04*, pages 1535–1538. ELRA. 126

Ferraresi, A., Zanchetta, E., Baroni, M., and Bernardini, S. (2008). Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54. 115, 126

Freudenthal, D., Pine, J., and Gobet, F. (2005). On the resolution of ambiguities in the extraction of syntactic categories through chunking. *Cognitive Systems Research*, 6(1):17–25. 98, 131

Ganchev, K., Graça, J. a., Gillenwater, J., and Taskar, B. (2010). Posterior regularization for structured latent variable models. *J. Mach. Learn. Res.*, 99:2001–2049. 27, 98, 101, 117

Gao, J. and Johnson, M. (2008). A comparison of bayesian estimators for unsupervised hidden markov model pos taggers. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 344–352, Stroudsburg, PA, USA. Association for Computational Linguistics. 98, 99, 101

Globerson, A., Chechik, G., Pereira, F., and Tishby, N. (2007). Euclidean embedding of co-occurrence data. *J. Mach. Learn. Res.*, 8:2265–2295. 33, 37, 38, 100

Goldberg, Y., Adler, M., and Elhadad, M. (2008). Em can find pretty good hmm pos-taggers (when given a good start). *Proceedings of ACL-08. Columbus, OH*, pages 746–754. 53, 54

Goldwater, S. and Griffiths, T. (2007a). A fully Bayesian approach to unsupervised part-of-speech tagging. In *Annual Meeting-Assosiation for Computational Linguistics*, volume 45, page 744. 7, 42, 53, 54, 64

Goldwater, S. and Griffiths, T. (2007b). A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 744–751, Prague, Czech Republic. Association for Computational Linguistics. 27, 98, 101, 117

Goodman, J. (2001). A bit of progress in language modeling. *Computer Speech and Language*, pages 403–434. 20, 21, 81

Graff, D., Rosenfeld, R., and Paul, D. (1995). Csr-iii text. Linguistic Data Consortium, Philadelphia. 106

Haghighi, A. and Klein, D. (2006). Prototype-driven learning for sequence models. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 320–327, Stroudsburg, PA, USA. Association for Computational Linguistics. 98, 99, 102

Hakkani-Tür, D. Z., Oflazer, K., and Tür, G. (2002). Statistical morphological disambiguation for agglutinative languages. *Computers and the Humanities*, 36(4):381–410. 12

Hawker, T. (2007). Usyd: Wsd and lexical substitution using the web1t corpus. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 446–453, Prague, Czech Republic. Association for Computational Linguistics. 81

Jakubíček, M., Kilgarriff, A., Kovář, V., Rychlỳ, P., and Suchomel, V. (2013). The tenten corpus family. In *International Conference on Corpus Linguistics, Lancaster*. 126

Jansen, B., Spink, A., and Pfaff, A. (2000). Linguistic aspects of web queries. In *Proceedings of the ASIS Annual Meeting*, volume 37, pages 169–76. 88

Johnson, M. (2007a). Why doesn't EM find good HMM POS-taggers? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305, Prague, Czech Republic. Association for Computational Linguistics. 27, 98, 101

Johnson, M. (2007b). Why doesn't EM find good HMM POS-taggers. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 296–305. 43, 52, 53

Kaufman, L. and Rousseeuw, P. (2005). *Finding groups in data: an introduction to cluster analysis*. Wiley series in probability and mathematical statistics. Applied probability and statistics. Wiley. 110

Kilgarriff, A., Reddy, S., Pomikálek, J., and Avinesh, P. (2010). A corpus factory for many languages. In *LREC*. 126

Kohomban, U. S. and Lee, W. S. (2005). Learning semantic classes for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 34–41, Ann Arbor, Michigan. Association for Computational Linguistics. 81, 89

Kohomban, U. S. and Lee, W. S. (2007). Optimizing Classifier Performance in Word Sense Disambiguation by Redefining Word Sense Classes. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1635–1640. 82

Kucera, H. and Francis, W. N. (1967). *Computational Analysis of Present-Day American English*. Brown University Press. 88

Lamar, M., Maron, Y., and Bienenstock, E. (2010a). Latent-descriptor clustering for unsupervised pos induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 799–809, Stroudsburg, PA, USA. Association for Computational Linguistics. 97, 101, 117

Lamar, M., Maron, Y., Johnson, M., and Bienenstock, E. (2010b). Svd and clustering for unsupervised pos tagging. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 215–219, Uppsala, Sweden. Association for Computational Linguistics. 24, 97, 101

Lee, Y. K., Haghighi, A., and Barzilay, R. (2010). Simple type-level unsupervised pos tagging. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 853–861, Stroudsburg, PA, USA. Association for Computational Linguistics. 27, 127

Levinger, M., Itai, A., and Ornan, U. (1995). Learning morpho-lexical probabilities from an untagged corpus with an application to Hebrew. *Computational Linguistics*, 21(3):404. 73

MacWhinney, B. (2000). *The CHILDES Project: Tools for Analyzing Talk, Volume II: The Database*, volume 2. Lawrence Erlbaum. 25

Magnini, B., Strapparava, C., Pezzulo, G., and Gliozzo, A. (2003). The role of domain information in word sense disambiguation. *Natural Language Engineering*, 8(04):359–373. 81

Manning, C., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*, chapter 17. Cambridge University Press. 110

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts. 26

Marcus, M. P., Santorini, B., Marcinkiewicz, M. A., and Taylor, A. (1999). Treebank-3. Linguistic Data Consortium, Philadelphia. 98, 106, 115

Maron, Y., Lamar, M., and Bienenstock, E. (2010). Sphere embedding: An application to part-of-speech induction. In Lafferty, J., Williams, C. K. I., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 1567–1575. 10, 24, 33, 34, 37, 38, 117, 119, 120

Martinez, D., de Lacalle, O. L., and Agirre, E. (2008). On the use of automatically acquired examples for all-nouns word sense disambiguation. *Journal of Artificial Intelligence Research*, 33:79–107. 82

Merialdo, B. (1994). Tagging english text with a probabilistic model. *Computational linguistics*, 20(2):155–171. 42, 43, 53

Mihalcea, R. and Edmonds, P., editors (2004). *SENSEVAL-3: The Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain. 81

Mintz, T. (2003). Frequent frames as a cue for grammatical categories in child directed speech. *Cognition*, 90(1):91–117. 24, 25, 26, 100, 119

Mitzenmacher, M. (2004). A brief history of generative models for power law and lognormal distributions. *Internet mathematics*, 1(2):226–251. 43, 53

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142. 125

Navigli, R. (2006). Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 105–112, Sydney, Australia. Association for Computational Linguistics. 81

Navigli, R. (2009). Word sense disambiguation: a survey. *ACM Computing Surveys*, 41(2):1–69. 81

Ng, A., Jordan, M., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856. 110

OflazerH, K., Hakkani-Tür, D., and Tür, G. (2002). Design for a Turkish treebank. 72, 75

Peters, W., Peters, I., and Vossen, P. (1998). Automatic Sense Clustering in EuroWordNet. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 409–416. 81

Pustejovsky, J., Hanks, P., and Rumshisky, A. (2004). Automated induction of sense in context. In *Proceedings of the 20th international conference on Computational Linguistics*, page 924. Association for Computational Linguistics. 71

Ravi, S. and Knight, K. (2009a). Minimized models for unsupervised part-of-speech tagging. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 504–512, Suntec, Singapore. Association for Computational Linguistics. 43

Ravi, S. and Knight, K. (2009b). Minimized models for unsupervised part-of-speech tagging. In *ACL-IJCNLP '09: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1*, pages 504–512, Morristown, NJ, USA. Association for Computational Linguistics. 54, 58, 63

Redington, M., Crater, N., and Finch, S. (1998). Distributional information: A powerful cue for acquiring syntactic categories. *Cognitive Science*, 22(4):425–469. 24, 100

Rosenberg, A. and Hirschberg, J. (2007). V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 Joint Conference on Empirical*

Methods in Natural Language Processing and Computational Natural Language Learning, pages 410–420. 103

Rosenfeld, R. (2000). Two decades of statistical language modeling: Where do we go from here. In *Proceedings of the IEEE*, volume 88, pages 1270–1278. 20, 81

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323. 108

Sahlgren, M. (2006). *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. PhD thesis, Stockholm University. 27

Sak, H., Güngör, T., and Saraçlar, M. (2008). Turkish language resources: Morphological parser, morphological disambiguator and web corpus. *Advances in natural language processing*, pages 417–427. 12, 77

Schütze, H. (1995). Distributional part-of-speech tagging. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics*, EACL '95, pages 141–148, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 27, 28, 97, 100, 101

Schütze, H. and Pedersen, J. (1993). A Vector Model for syntagmatic and paradigmatic relatedness. In *Proceedings of the 9th Annual Conference of the University of Waterloo Centre for the New OED and Text Research*, Oxford, England. 24, 25, 27, 98

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656. 80

Smith, N. A. and Eisner, J. (2005). Contrastive estimation: training log-linear models on unlabeled data. In *ACL '05: Proceedings of the 43rd Annual Meeting on Association*

for Computational Linguistics, pages 354–362, Morristown, NJ, USA. Association for Computational Linguistics. 53, 54, 63

St Clair, M. C., Monaghan, P., and Christiansen, M. H. (2010). Learning grammatical categories from distributional cues: flexible frames for language acquisition. *Cognition*, 116(3):341–60. 24, 26, 100, 101, 119

Stevenson, M. (2003). *Word Sense Disambiguation: The Case for Combinations of Knowledge Sources*. CSLI. 81

Stolcke, A. (2002a). SRILM-an extensible language modeling toolkit. In *Seventh International Conference on Spoken Language Processing*, volume 3. Citeseer. 55

Stolcke, A. (2002b). Srilm-an extensible language modeling toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286. 106, 115

Tenenbaum, J., Silva, V., and Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319. 108

Tomasello, M. (2009). *Constructing a language: A usage-based theory of language acquisition*. Harvard University Press. 30

Toutanova, K. and Johnson, M. (2007). A Bayesian LDA-based model for semi-supervised part-of-speech tagging. In *Proceedings of NIPS*, volume 20. 63

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics. 98, 99, 101

Turney, P. D. and Pantel, P. (2010). From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res. (JAIR)*, 37:141–188. 24

Véronis, J. (2004). Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language*, 18(3):223–252. 25

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269. 43

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372. 125

Yarowsky, D. (1992). Word sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 454–460, Nantes. 81

Yarowsky, D. and Florian, R. (2002). Evaluating sense disambiguation across diverse parameter spaces. *Natural Language Engineering*, 8(4):293–310. 11

Yatbaz, M. A., Sert, E., and Yuret, D. (2012). Learning syntactic categories using paradigmatic representations of word context. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 940–951, Jeju Island, Korea. Association for Computational Linguistics. 97, 103, 129

Yatbaz, M. A. and Yuret, D. (2010). Unsupervised part of speech tagging using unambiguous substitutes from a statistical language model. In *Coling 2010: Posters*, pages 1391–1398, Beijing, China. Coling 2010 Organizing Committee. 99

Yuret, D. (2004). Some experiments with a Naive Bayes WSD system. In Mihalcea, R. and Edmonds, P., editors, *Senseval-3: Third International Workshop on the Evaluation of*

Systems for the Semantic Analysis of Text, pages 265–268, Barcelona, Spain. Association for Computational Linguistics. 12

Yuret, D. (2007). KU: Word sense disambiguation by substitution. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 207–214, Prague, Czech Republic. Association for Computational Linguistics. 81

Yuret, D. (2008). Smoothing a tera-word language model. In *Proceedings of ACL-08: HLT, Short Papers*, pages 141–144, Columbus, Ohio. Association for Computational Linguistics. 81, 85

Yuret, D. (2012). Fastsubs: An efficient and exact procedure for finding the most likely lexical substitutes based on an n-gram language model. *Signal Processing Letters, IEEE*, 19(11):725–728. 22, 115

Yuret, D. and Biçici, E. (2009). Modeling morphologically rich languages using split words and unstructured dependencies. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 345–348. Association for Computational Linguistics. 72

Yuret, D. and Ture, F. (2006). Learning morphological disambiguation rules for turkish. In Moore, R. C., Bilmes, J. A., Chu-Carroll, J., and Sanderson, M., editors, *HLT-NAACL*. The Association for Computational Linguistics. 15