On the importance of decoding in semi-supervised learning

by

Mohammad Mahdi Soleimani Nasab

B.Sc., University of Tehran, 2013

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science

in the School of Computing Science Faculty of Applied Sciences

© Mohammad Mahdi Soleimani Nasab 2016 SIMON FRASER UNIVERSITY Spring 2016

All rights reserved. However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for “Fair Dealing.” Therefore, limited reproduction of this work for the purposes of private study, research, education, satire, parody, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.
Approval

Name: Mohammad Mahdi Soleimani Nasab
Degree: Master of Science (Computing Science)
Title: On the importance of decoding in semi-supervised learning
Examinining Committee: Chair: Dr. Nick Sumner
Assistant Professor
Dr. Anoop Sarkar
Senior Supervisor
Professor
Dr. Fred Popowich
Supervisor
Professor
Dr. Martin Ester
Internal Examiner
Professor

Date Defended: April 1, 2016
Abstract

In many natural language processing (NLP) tasks a large amount of unlabelled data is available while labelled data is hard to attain. Bootstrapping techniques have been shown to be very successful on a variety of NLP tasks using only a small amount of supervision. In this research we have studied different bootstrapping techniques that separate the training step of the algorithm from the decoding step which produces the argmax label on test data. We then explore generative models trained in the conventional way using the EM algorithm but we use an initialization step and a decoding techniques similar to the Yarowsky bootstrapping algorithm. The new approach is tested on the named entity classification and word sense disambiguation tasks and has shown significant improvement over previous generative models.

Keywords: EM; Co-train; Self-train; semi-supervised learning
Dedication

To my family, whom their unconditional love has made me more fulfilled than I could ever ask for.
# Table of Contents

Approval ii  
Abstract iii  
Dedication iv  
Table of Contents v  
List of Tables vii  
List of Figures ix  

1 Introduction 1  
1.1 Supervised, semi-supervised, and unsupervised learning 2  
1.2 Generative and discriminative algorithms 3  
1.3 Inductive and transductive learning 3  
1.4 General approaches to semi-supervised learning 4  
1.5 Contributions 5  
1.6 Notation for semi-supervised learning 5  
1.7 Summary 5  

2 Task and Data 7  
2.1 Named entity classification 7  
2.2 Task of Collins and Singer (1999) 7  
2.3 CoNLL 2003 shared task 8  
2.4 Word sense disambiguation 12  
2.5 Methodology and reporting results 13  
2.6 Summary 15  

3 Algorithms 16  
3.1 Training Algorithms 16  
3.1.1 The Yarowsky algorithm 16  
3.1.2 Collins and Singer (1999)’s Yarowsky-cautious algorithm 17
3.1.3 EM algorithm .................................................. 17
3.1.4 Co-perceptron algorithm .................................. 20
3.1.5 DL-CoTrain algorithm ....................................... 22
3.1.6 EM algorithm with posterior regularization ............ 23
3.2 Decoding Algorithms ......................................... 24
  3.2.1 Using full joint distribution ............................... 25
  3.2.2 Using most “informative” feature(s) ................. 25
3.3 Summary ......................................................... 26

4 Experiments ...................................................... 27
  4.1 Experimental setup ........................................... 27
  4.2 NYT Results .................................................. 27
  4.3 CoNLL 2003 Results ......................................... 28
  4.4 WSD Results .................................................. 30
  4.5 Analysis ....................................................... 30

5 Conclusion and Future Work .................................. 36

Bibliography ....................................................... 37
Table 1.1 Notation of the symbols. .......................................................... 6

Table 2.1 An example of the extracted named entities and the corresponding features. .......................................................... 12

Table 2.2 Distribution of labels in training and test data sets extracted from CoNLL-2003 shared task. .......................................................... 12

Table 4.1 Accuracy for different learning methods on the named entity classification task of Collins and Singer (1999). The Decoding columns shows the function $f(x, y)$ that we have computed the $\arg\max_y f(x, y)$ on. For the joint distribution it will be $f(x, y) = P(x, y)$ and the most confident feature refers to $f(x, y) = P(\hat{f}|y)$ where $\hat{f}$ is computed by equation 3.15. The result of using only the whole phrase of proper nouns to classify each is shown under “Whole phrase feature” column. 28

Table 4.2 Accuracy of different learning methods (previous works) on the named entity classification task of Collins and Singer (1999). .......................................................... 28

Table 4.3 Accuracy for different learning methods on the named entity classification task for CoNLL 2003 data set. The Decoding columns shows the function $f(x, y)$ that we have computed the $\arg\max_y f(x, y)$ on. For the joint distribution it will be $f(x, y) = P(x, y)$ and the most confident feature refers to $f(x, y) = P(\hat{f}|y)$ where $\hat{f}$ is computed by equation 3.15. The result of using only the whole phrase of proper nouns to classify each is shown under “Whole phrase feature” column. 29

Table 4.4 Accuracy of different learning methods (previous works) on the named entity classification task for CoNLL 2003 data set. .......................................................... 29

Table 4.5 Accuracy of EM algorithm with different decoding methods for the word sense disambiguation tasks. The Decoding column shows the function $f(x, y)$ that we have computed the $\arg\max_y f(x, y)$ on. For the joint distribution it will be $f(x, y) = P(x, y)$ and the most confident feature refers to $f(x, y) = P(\hat{f}|y)$ where $\hat{f}$ is computed by equation 3.15. 31
Table 4.6  Accuracy of different learning methods (previous works) on the word sense disambiguation tasks of Eisner and Karakos (2005) (*Drug*, *Land*, and *Sentence*).
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Three sample named entity training instances and the corresponding true labels which are not provided to the algorithm.</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Number of instances for each class in the test set of Collins and Singer (1999) named entity classification task.</td>
<td>8</td>
</tr>
<tr>
<td>2.3</td>
<td>Named entity classification data point example.</td>
<td>8</td>
</tr>
<tr>
<td>2.4</td>
<td>Set of seed rules for the named entity classification task.</td>
<td>9</td>
</tr>
<tr>
<td>2.5</td>
<td>An example of the extracted data set from CoNLL 2003 shared task.</td>
<td>10</td>
</tr>
<tr>
<td>2.6</td>
<td>A sample sentence from the CoNLL 2003 data set. The named entity and the corresponding features extracted from this sentence are shown in Figure 2.5.</td>
<td>11</td>
</tr>
<tr>
<td>2.7</td>
<td>Number of data points in the test sets and their label distributions for the WSD tasks. The names for the senses are obtained from Whitney (2012).</td>
<td>13</td>
</tr>
<tr>
<td>2.8</td>
<td>Size of training and test sets for different WSD tasks.</td>
<td>13</td>
</tr>
<tr>
<td>2.9</td>
<td>Examples from the “drug” WSD data set and the set of extracted features. The NEXT[OL]_ and PREV[OL]_ prefixes define the words following and preceding the word “drug” in the sentence. O features are the original words and L features are the lemmatized words.</td>
<td>14</td>
</tr>
<tr>
<td>2.10</td>
<td>The set of seed-rules for WSD tasks.</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>A sample decision list taken from the Yarowsky algorithm on the named entity task of Collins and Singer (1999).</td>
<td>18</td>
</tr>
<tr>
<td>3.2</td>
<td>Few instances from the named entity data and the corresponding label from the decision list of Figure 3.1. NA indicates that the decision list could not generate a label for the instance hence the label is not available.</td>
<td>18</td>
</tr>
<tr>
<td>4.1</td>
<td>Accuracy (clean) of co-perceptron algorithm on the named entity classification task of Collins and Singer (1999) at different iterations. Three lines represent different decoding methods: using spelling features only (view 1), using contextual features only (view 2), or using all features.</td>
<td>29</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Number of unlabelled examples in the training set of named entity task (Collins and Singer (1999)) which the two views (co-perceptron algorithm) did not agree upon.</td>
<td>30</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Number of seed-labelled examples in the training set of named entity task (Collins and Singer (1999)) which the label generated by the view is not the correct label. Note that in order to keep the function always decreasing one can try to decrease the $\epsilon$ amount in later iterations.</td>
<td>31</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Log likelihood of parameters learnt by EM algorithm over named entity (Collins and Singer (1999)) training data.</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Accuracy of EM algorithm with different decoding methods for named entity classification task of Collins and Singer (1999).</td>
<td>32</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Log likelihood of parameters learnt by EM algorithm over CoNLL 2003 named entity training data.</td>
<td>33</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Accuracy of EM algorithm with different decoding methods for named entity classification task for CoNLL 2003 data set.</td>
<td>33</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In this thesis we explore semi-supervised learning using a generative model. We argue that by using a different decoding algorithm that uses a sorted order over features in the model, a simple generative model trained by the expectation-maximization (EM) algorithm can obtain results that are competitive with less well understood discriminative learning methods.

Our formulation of semi-supervised learning is a natural extension to generative models used for unsupervised learning. We assume the labelled and unlabelled data points are from the same domain and distribution. The supervision is provided in the form of a small set of labelled instances or a small set of seed rules which is then used to derive the labelled data points. Abney (2002) refers to the setting where supervision is provided in the form of a small set of seed rules as Bootstrapping. Self-training, introduced by Scudder (2006) and Co-training by Blum and Mitchell (1998) are two dominant discriminative methods for bootstrapping in semi-supervised learning. The Yarowsky algorithm was one of the first bootstrapping algorithms to become widely known in NLP. It iteratively learns a decision list that consists of weighted rules of the form “If input \( x \) has feature \( f \), then the output is \( y \)”. Abney (2004) analyzed the Yarowsky algorithm and showed that variants of the algorithm optimize an upper bound on the negative log data likelihood. Haffari and Sarkar (2012) further proposed and analyzed new variations on the Yarowsky algorithm and the variations proposed by Abney (2004). They showed there is a strong connection between graph-based semi-supervised learning algorithms and the Yarowsky algorithm. In Whitney and Sarkar (2012), the authors proposed a graph propagation algorithm which uses a graph structure and objective function similar to Haffari and Sarkar (2012) (based on cross entropy) but does the propagation with the Yarowsky algorithm.

We start by explaining different approaches to machine learning problems, the difference between generative and discriminative models, and how the labelled test data can be used in the training step of a semi-supervised learning task. We then explain the training and decoding algorithms used in this thesis in greater detail. The results for different
training and decoding approaches are shown and we are especially interested in compar-
ing results with other discriminative algorithms. Our main contribution is that using the
well-understood EM algorithm for training our model we can achieve the same state-of-the-
art results as hard-to-analyze discriminative algorithms simply by changing the decoding
algorithm. Finally we analyze our findings and discuss the future work.

1.1 Supervised, semi-supervised, and unsupervised learning

There are three general approaches to machine learning: Unsupervised Learning, Supervised
Learning, and Semi-supervised Learning. In unsupervised learning a large amount of un-
labelled data is available; \( X = (x_1, \ldots, x_n) \) where data points are drawn from a common
distribution \( X \). The goal of unsupervised learning is to find interesting structure in the
unlabelled data set by estimating the parameters of a model (by maximizing the data like-
lihood) that is assumed to have generated the data points. There are also weaker forms of
unsupervised learning where the main goal is not to find the common distribution but to
facilitate preparing the data for other tasks. For example Chapelle et al. (2010) mention
clustering, dimensionality reduction, and outlier detection as some of these forms.

Supervised learning is a common approach to machine learning tasks where a large
amount of labelled data is available. The goal of supervised learning is to find a mapping
from input \( x \) to a label \( y \). Under this definition the training set is assumed to be a set of
\((x_i, y_i)\) pairs where each input \( x_i \) is drawn i.i.d. (independently and identically distributed)
from a common distribution. The labels \( y_i \in Y \) can accept discrete or continuous values.
The task will then be called a classification or regression task respectively. Eventually the
model, learnt using the training set, is evaluated on a previously unseen data set called the
test set.

Semi-supervised learning is something in between the two approaches mentioned above.
Here the algorithm is provided with a data set \( X \) where the labels are given for a proportion
of data points, \((x_1, y_1), \ldots, (x_l, y_l)\), and the label for the rest of data points is unknown,
\(x_{l+1}, \ldots, x_n\). The goal is to infer a function that will generate the label for the test data.
There is also another variation of semi-supervised learning where the algorithm is provided
with a set of constraints on the data points and structure of the problem, i.e. data points
which have a certain feature must have the same label. Chang et al. (2007) have proposed a
framework for guiding semi-supervised learning with the use of constraint-driven learning.

Many tasks in natural language processing (NLP) involve textual data. For example
machine translation, named entity recognition/classification, part of speech tagging and
parsing are among such tasks where the input is usually a set of sentences or some sort of
structured data extracted from a text document. In these tasks a large amount of unlabelled
data is available for example on the world wide web or can be constructed automatically but
these data sources rarely contain the labels. Generating the labels for these data sources is a
time consuming and expensive task. Moreover, not only unsupervised and semi-supervised learning are practical approaches to many NLP tasks but they also can help in learning a model for the data that is not concentrated on just one particular domain.

In this work we will be using a semi-supervised approach in order to solve a named entity classification task. Our research is closely related to those of Collins and Singer (1999) and Whitney and Sarkar (2012) and as a result we have executed the experiments on the same data sets and in a similar setting to be able to do the comparison.

1.2 Generative and discriminative algorithms

Learning algorithms can also be classified into another two groups. Generative algorithms try to model the conditional probability distribution of an input $x$ given the label $y$. After estimating $P(x|y)$, by applying Bayes theorem one can compute the joint probability distribution or the predictive conditional distribution, Eq. 1.1. By having the joint distribution new pairs of $(x_i, y_i)$ can be generated, thus it is called a generative approach.

$$P(y|x) = \frac{P(x, y)}{P(x)} = \frac{P(y) \cdot P(x|y)}{\sum_{y'} P(y', x)} = \frac{P(y) \cdot P(x|y)}{\sum_{y'} P(y') \cdot P(x|y')}$$

In contrast with generative algorithms, discriminative algorithms do not try to estimate the underlying joint distribution but to model $P(y|x)$ directly. For example logistic regression is a discriminative algorithm where the output of the algorithm is a function that given an input $x$ will generate the most likely output label. In such a discriminative approach one does not explicitly model the joint distribution and this makes using $P(x)$ (e.g. when using unlabelled data) more challenging in discriminative learning. However, the aim of supervised learning is to infer the labels of the test set and discriminative methods are more aligned with this goal.

1.3 Inductive and transductive learning

Recall the input to a semi-supervised algorithm, a set of labelled data points $(x_1, y_1), ..., (x_l, y_l)$ and the unlabelled instances $x_{l+1}, ..., x_n$. The idea of transductive learning is to infer the labels for the unlabelled data points only where with inductive learning the goal is to find a prediction function that can generate the correct label for any input $x$ from the input space.

In the experiments done in this research we have mainly used and reported the results for inductive learning, meaning the training data is used by learning algorithms to train
the models and then it is tested on a separate test set. The reason behind it is that we had aimed to compare our findings with previous approaches and both data sets for the named entity classification task and word sense disambiguation tasks have separate training and test sets.

1.4 General approaches to semi-supervised learning

One generative approach to semi-supervised learning is maximum likelihood (ML) or maximum a posteriori (MAP) estimation with the EM algorithm. EM is a powerful general approach especially when there are latent variables in the data. Given a generative model and an initial set values for the parameters, EM finds the parameters estimation by iteratively maximizing the likelihood. As a result it is important to provide it with a good model and initial values.

Another generative approach to unsupervised and semi-supervised learning is clustering. The idea here is to organize the data points into a set of clusters, such that “similar” points, i.e. sharing the same hidden label, get assigned to the same cluster. As you may notice with unsupervised learning it can be hard to assign labels to clusters. Even with semi-supervised learning and the provided supervision the assignment might still be unclear specially when the number of clusters found by the algorithm is different from the expected number of classes. The supervision in semi-supervised learning can be provided in the form of a set of labelled instances or general information about the data in the form of constraints (Basu et al. (2004)).

Self-training and co-training mentioned earlier are among other discriminative approaches to semi-supervised learning. In self-training, by starting with a simple model, the algorithm repeatedly generates the labels for the data points and gets retrained on the output. The initial simple model can be built from some seed information, like a small set of labelled data. A well-known self-training algorithm is the Yarowsky algorithm (Yarowsky (1995)). It starts with an initial set of seed rules. Using these seed rules the algorithm generates the labels for a part of the input data which matches one of the seed rules. These new labelled instances are then used to extract more rules. The rules with a probability over a threshold are added to the seed rules. This process is then repeated for a fixed number of iterations or until no more new label is generated.

Co-training, introduced by Blum and Mitchell (1998), or multi-view learning refers to the semi-supervised tasks where redundant views of the same input data is used to train latent variable models. These redundant views for example can be context and spelling features in the case of named entity classification, images of an object taken from different angles in the case of object recognition, or multiple microphones in the case of voice analysis. These multiple views can are then used in different ways. Ganchev et al. (2008) uses constraints to regularize the posterior distribution of each view. To make the latent variable distributions
of two views close to each other, they minimize the Kullback-Leibler (KL) distance by finding its projections to a space where all the constraints are met. This posterior regularization based on constraints makes it possible for the multiple views to be generated by different processes and have different prior distributions.

1.5 Contributions

We make three particular contributions in this research:

1. We provide evidence that using different initialization and decoding methods, a simple generative model can perform as well as the hard-to-analyze discriminative algorithms for the task of named entity classification and word sense disambiguation on multiple data sets.

2. We provide a new data set for semi-supervised learning extracted from the 2003 CoNLL shared task of Tjong Kim Sang and De Meulder (2003) which will be freely available to researchers.

3. We provide results from our implementation of several algorithms (EM algorithm, the EM algorithm with posterior regularization, Yarowsky algorithm, and co-perceptron), allowing consistent comparison. The source code will be available online (under SFU natural language laboratory GitHub repository) for others to replicate our experiments.

1.6 Notation for semi-supervised learning

We use the notation by Abney (2004) with minor changes. The notation is shown in table 1.1.

1.7 Summary

In this chapter we explained the different approaches that have been used in semi-supervised learning. Bootstrapping methods use a small set of labelled data or a small set of seed rules to train an initial model. This model is then used to make inference on the unlabelled set which later will change the model itself. Generative approaches try to estimate the parameters of the underlying distribution of the input data hence enabling us to make inference on unseen data points. We also explained inductive and transductive learning and the differences between the two.
\( x \)  
—an input example

\( f \)  
denotes a feature

\( y \)  
denotes a label

\( Y \)  
denotes the set of labels

\( y^v \)  
denotes the label associated with view \( v \)

\( L \)  
number of labels

\( X \)  
set of all training examples

\( \zeta \)  
denotes the threshold

\( m_i \)  
denotes the length of instance \( i \)

\( x_j \)  
denotes the \( j^{th} \) feature for input \( x \)

\( \theta \)  
denotes the set of parameters

\( \theta^t \)  
denotes the parameters at iteration \( t \)

\( v \)  
denotes the view number

\( t \)  
denotes iteration number

\( \delta(y|i) \)  
denotes the probability of label \( y \) for input \( X_i \)

\( \Lambda_f \)  
denotes the set of input examples which possess feature \( f \)

\( \Lambda_{fy} \)  
denotes the set of input instances which possess \( f \) and are labelled as \( y \)

\( F \)  
set of all features

Table 1.1: Notation of the symbols.
Chapter 2

Task and Data

2.1 Named entity classification

Named entity (NE) classification is the task of learning a classifier function, \( f \), that given an input string (proper noun) outputs the type of the name. Here we assume the type will be one of the categories person, location, or organization. For example a good classifier would classify Mr. Gorbachev as a person, Palestine National Council as an organization, and San Antonio as a location. In named entity classification tasks an instance contains both spelling and contextual features. A spelling feature represents the actual proper noun phrase. For example in Mr. Gorbachev, the feature \( X01_{\text{Mr.}} \) shows a token from the proper noun. Contextual features represent the context where the proper noun has appeared in. For example in “...says Mr. Cooper, a vice president of...”, the word president is a modifier for the proper noun Mr. Cooper and can be considered as a contextual feature.

For the named entity classification task we use two data sets. One is the task of Collins and Singer (1999) (NYT) and the other one is a data set we extracted from the CoNLL 2003 shared task, Tjong Kim Sang and De Meulder (2003).

2.2 Task of Collins and Singer (1999)

The first NE data set is the same data set as Collins and Singer (1999). Here each instance is a proper noun and the goal is to classify each phrase as either person, organization, or location. Figure 2.1 shows an example from the training set for each class. The number of examples in training and test sets are shown in Figure 2.2. The test set also contains 85 examples that are not labelled as any of the three classes and are considered as noise. In our experiments we report separate results for the whole test set (Noise) and the test set without noise instances (clean).

The data set has first been extracted in the form of sentences from New York Times. These sentences were then parsed by Collins (1996) statistical parser. Using the information
Figure 2.1: Three sample named entity training instances and the corresponding true labels which are not provided to the algorithm.

<table>
<thead>
<tr>
<th>Training example</th>
<th>Label (not provided)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X0_Washington X1_headquarters-in X3_RIGHT</td>
<td>Location</td>
</tr>
<tr>
<td>X0_Bruce-Jenkins X01_analyst X2_Bruce X2_Jenkins X3_LEFT</td>
<td>Person</td>
</tr>
<tr>
<td>X0_OPEC X11_state-of X5_ALLCAP X3_RIGHT</td>
<td>Organization</td>
</tr>
</tbody>
</table>

Figure 2.2: Number of instances for each class in the test set of Collins and Singer (1999) named entity classification task.

<table>
<thead>
<tr>
<th>Label</th>
<th>Number of test examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>186</td>
</tr>
<tr>
<td>Person</td>
<td>289</td>
</tr>
<tr>
<td>Organization</td>
<td>402</td>
</tr>
<tr>
<td>Noise</td>
<td>123</td>
</tr>
<tr>
<td>Total</td>
<td>1000</td>
</tr>
</tbody>
</table>

provided in the parse trees, word sequences which met certain criteria were selected. In particular these word sequences were either a sequence of consecutive proper nouns (words tagged as NNP or NNPS) with the last word being the head or were contained in an NP node which had appeared in one of the two contexts: The NP had an appositive modifier with a singular noun (tagged NN) as its head or there was a preposition (tagged PP) with a singular noun as its head and the NP was a complement to that PP. An example for such contexts is shown in Figure 2.3.

We have used the same set of seed rules used by Collins and Singer (1999) with learning algorithms. These seven rules are shown in Figure 2.4. “Xx_” prefixes determine the type of each feature, i.e. X0_ and X2_ are both spelling features where X0_ features represent the whole phrase while X2_ features represent a single word inside the phrase.

### 2.3 CoNLL 2003 shared task

The next NE data set is extracted from the CoNLL-2003 shared task, Tjong Kim Sang and De Meulder (2003). The data set contains named entities in two languages, English and German. We use the English data in our experiments. The English data was initially

Figure 2.3: Named entity classification data point example.
extracted from the Reuters Corpus. It contains separated test, training (labelled), and raw
(unlabelled) preprocessed data. The script to extract the data set from the CoNLL-2003
shared task data will be available online (under SFU natural language laboratory GitHub
repository).

The preprocessing done on the extracted sentences includes applying a tokenizer, part-
of-speech (POS) tagger, and the chunker by Daelemans et al. (2002). The labels for the
training and test data were generated by human readers. The data set contains four different
labels: persons (PER), locations (LOC), organizations (ORG), and miscellaneous (MISC).
The last category denotes all the names which are not in other categories. This diverse
category includes adjectives, like French, or name of an event, like 1000 Lakes Rally.

In the data files, each sentence is written on multiple lines with a blank line separating
sentences. Each line represents one word from a sentence and contains the following fields:
the word, its POS tag, its phrase tag, and the named entity label (only for training and
test data). Words tagged with O are outside of any named entities and I-XXX denotes
that the word is inside a named entity labelled as XXX. In case two named entities appear
next to each other the first word of the second named entity is labelled as B-XXX. Here
is an example sentence from the training set:

```
U.N NNP I-NP I-ORG
official NN I-NP O
Ekeus NNP I-NP I-PER
heads VBZ I-VP O
for IN I-PP O
Baghdad NNP I-NP I-LOC
.
```

Each example (named entity) consists of few consecutive words tagged as NNP or NNPS.
For the examples extracted from the data set we also make sure that if two proper nouns
are next to each other, they will be considered as separated data instances. For the training
and test sets, it is done using the provided label (second one starts with B-XXX) where
for the unlabelled data since we do not have access to the labels we use the corresponding
phrase tag.
Example
I-PER S0_Nikolaus-van-der-Pas S1_Nikolaus S1_van S1_der S1_Pas C_-2_chief
C_-2_JJ C_-1_spokesman C_-1_NN C_1_told C_1_VBD C_2_a C_2_DT

Figure 2.5: An example of the extracted data set from CoNLL 2003 shared task.

Since we intended to compare our results with the results reported by Ganchev et al. (2008) we have tried to extract a similar data set as them. In order to do so after finding the proper noun, it alongside all its words are added as spelling features. The contextual features are up to two words before and after the proper noun phrase and their corresponding POS tags. A sample data example is shown in Figure 2.5. The sentence from which this example is extracted is shown in Figure 2.6. For the labelled data the label, I-XXX, is added as the first element. For the unlabelled data −1 is added as the label. For the contextual features the relative position of the word is added as part of the feature. One should note that Ganchev et al. (2008) do not provide details on which part of the Reuters corpus they use or how they extract named entities and as a result it is not unusual to get slightly different results.

Here is another example from the data set and the named entities that get extracted:

| The | DT | I-NP | I-ORG |
| New | NNP | I-NP | I-ORG |
| York | NNP | I-NP | I-ORG |
| Times | NNP | I-NP | I-ORG |
| said | VBD | I-VP | O |
| on | IN | I-PP | O |
| Tuesday | NNP | I-NP | O |
| some | DT | B-NP | O |
| of | IN | I-PP | O |
| the | DT | I-NP | O |
| options | NNS | I-NP | O |
| trading | NN | I-NP | O |
| in | IN | I-PP | O |
| MFS | NNP | I-NP | I-ORG |
| last | JJ | B-NP | O |
| Friday | NNP | I-NP | O |
| may | MD | I-VP | O |
| suggest | VB | I-VP | O |
| insider | NN | I-NP | O |
| trading | NN | I-NP | O |
| . | . | O | O |

We extract New York Times, Tuesday, Friday, and MFS. The corresponding spelling and contextual features for each one is shown in table 2.1. For the labelled (training and test) data we ignore the phrases that are labelled as O. Similar to Collins and Singer (1999)
<table>
<thead>
<tr>
<th>“&quot;</th>
<th>“&quot;</th>
<th>O</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>We</td>
<td>PRP</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>do</td>
<td>VBP</td>
<td>I-VP</td>
<td>O</td>
</tr>
<tr>
<td>n’t</td>
<td>RB</td>
<td>I-VP</td>
<td>O</td>
</tr>
<tr>
<td>support</td>
<td>VB</td>
<td>I-VP</td>
<td>O</td>
</tr>
<tr>
<td>any</td>
<td>DT</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>such</td>
<td>JJ</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>recommendation</td>
<td>NN</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>because</td>
<td>IN</td>
<td>I-SBAR</td>
<td>O</td>
</tr>
<tr>
<td>we</td>
<td>PRP</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>do</td>
<td>VBP</td>
<td>I-VP</td>
<td>O</td>
</tr>
<tr>
<td>n’t</td>
<td>RB</td>
<td>I-VP</td>
<td>O</td>
</tr>
<tr>
<td>see</td>
<td>VB</td>
<td>I-VP</td>
<td>O</td>
</tr>
<tr>
<td>any</td>
<td>DT</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>grounds</td>
<td>NNS</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>for</td>
<td>IN</td>
<td>I-PP</td>
<td>O</td>
</tr>
<tr>
<td>it</td>
<td>PRP</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>,</td>
<td>,</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>”</td>
<td>”</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>the</td>
<td>DT</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>Commission</td>
<td>NNP</td>
<td>I-NP</td>
<td>I-ORG</td>
</tr>
<tr>
<td>’s</td>
<td>POS</td>
<td>B-NP</td>
<td>O</td>
</tr>
<tr>
<td>chief</td>
<td>JJ</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>spokesman</td>
<td>NN</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td><strong>Nikolaus</strong></td>
<td>NNP</td>
<td>I-NP</td>
<td>I-PER</td>
</tr>
<tr>
<td><strong>van</strong></td>
<td>NNP</td>
<td>I-NP</td>
<td>I-PER</td>
</tr>
<tr>
<td><strong>der</strong></td>
<td>FW</td>
<td>I-NP</td>
<td>I-PER</td>
</tr>
<tr>
<td><strong>Pas</strong></td>
<td>NNP</td>
<td>I-NP</td>
<td>I-PER</td>
</tr>
<tr>
<td>told</td>
<td>VBD</td>
<td>I-VP</td>
<td>O</td>
</tr>
<tr>
<td>a</td>
<td>DT</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>news</td>
<td>NN</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>briefing</td>
<td>NN</td>
<td>I-NP</td>
<td>O</td>
</tr>
<tr>
<td>.</td>
<td>.</td>
<td>O</td>
<td>O</td>
</tr>
</tbody>
</table>

Figure 2.6: A sample sentence from the CoNLL 2003 data set. The named entity and the corresponding features extracted from this sentence are shown in Figure 2.5.
we use a dictionary to remove names of months and days of the week from the data set. As it is shown below, the features starting with S and C are the spelling and context features respectively. The next number is the relative position of the word to the named entity.

<table>
<thead>
<tr>
<th>Named entity</th>
<th>Extracted features</th>
</tr>
</thead>
<tbody>
<tr>
<td>New York Times</td>
<td>S0_The-New-York-Times S1_The S1_New S1_York S1_Times C_1_said C_1_VBD C_2_on C_2_IN</td>
</tr>
<tr>
<td>MFS</td>
<td>S0_MFS C_-2_trading C_-2_NN C_-1_in C_-1_IN C_1_last C_1_JJ C_2_Friday C_2_NNP</td>
</tr>
</tbody>
</table>

Table 2.1: An example of the extracted named entities and the corresponding features.

For our experiments we extracted 30,000 unlabelled named entities, 2000 labelled instances as training data, and 2000 examples as test data. The distribution of labels in the training and test data is shown in Table 2.2. Similar to Collins and Singer (1999) we consider the examples labelled MISC as noise and do not add them to training or test sets.

<table>
<thead>
<tr>
<th></th>
<th>PER</th>
<th>LOC</th>
<th>ORG</th>
<th>MISC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training (labelled)</td>
<td>742</td>
<td>745</td>
<td>513</td>
<td>0</td>
</tr>
<tr>
<td>Test</td>
<td>646</td>
<td>814</td>
<td>540</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.2: Distribution of labels in training and test data sets extracted from CoNLL-2003 shared task.

2.4 Word sense disambiguation

Word sense disambiguation (WSD) is the process of identifying the sense of a word (in a given context) which has multiple senses in the dictionary. For example consider the following two sentences, each with a different sense of the word sentence.

1. He typed a short sentence and then stopped.

2. The maximum sentence is two years imprisonment, or a fine, or both.

The reader can easily recognize that in the first sentence the mentioned word refers to the grammatical unit while in second sentence the sense of the word is declaration of a punishment for a crime.

For the word sense disambiguation tasks we use the same data sets as Eisner and Karakos (2005). We have done our experiments on ‘drug’, ‘land’, and ‘sentence’ tasks. Each of these data sets focus on one English word that may represent two senses(i.e. the word drug may refer to medical or illicit substance) and the goal is to classify each word as the correct sense. Figure 2.7 shows the number of data points in the test set and the distribution of labels for each task.
The data set had first been extracted from the Canadian Hansards, a parallel sentence-aligned corpus of parliamentary debates in English and French. They then selected the sentences for the same six words as Gale et al. (1992). The sentences in English were used to produce the training set while the gold labels were obtained from the corresponding translations in French. For each sentence and the intended word, the features were then selected. These features include the original and the lemmatized form of the immediate adjoining words (left and right), and the original and the lemmatized form of the words appearing in the same sentence. The size of training and test sets for each case is shown in fig. 2.8. Figure 2.9 shows sample sentences from the Hansard parallel corpus and the corresponding sets of extracted features. Whitney (2012) argues that the small training set alongside the relatively large test set sizes introduce sensitivities when learning on this data, meaning that the choice of seed rules may have a huge impact on the accuracy. But since our aim is to compare the result with previous approaches we have used the same set of seed rules. Figure 2.10 shows the set of seed rules used for each WSD task.

### 2.5 Methodology and reporting results

Our goal is to show that the initialization and decoding steps of a semi-supervised learning algorithm are important and the simple EM algorithm can achieve equivalent performance as existing algorithms. To do so, we have replicated the experiments done by Collins and Singer (1999) for the named entity classification tasks and Whitney and Sarkar (2012) for word sense disambiguation task.

In Collins and Singer (1999) the authors have defined clean accuracy as the accuracy obtained from classifying instances which are not noise. Number of instances for each class in the corresponding test set is shown in Figure 2.2. In Collins and Singer (1999), the
This drug has long been regarded as a cause of death if not used properly.

Such people should certainly qualify for free drugs.

Figure 2.9: Examples from the “drug” WSD data set and the set of extracted features. The NEXT[OL]_ and PREV[OL]_ prefixes define the words following and preceding the word “drug” in the sentence. O features are the original words and L features are the lemmatized words.

Figure 2.10: The set of seed-rules for WSD tasks.
authors exclude the 38 temporal expressions from the test set that can be easily identified by a list of days/months. Clean accuracy is then computed as $N_c/(962 - 85)$ where $N_c$ is the total number of instances classified correctly by the algorithm. Noise accuracy is the accuracy over all instances, $N_c/962$. We have reported the clean classification accuracy. For the word sense disambiguation tasks there are no noise examples and we only report one number, the result of the experiment on the test set.

For the CoNLL-2003 named entity data set we use the 2000 labelled and 30,000 unlabelled examples as training data. After the learning algorithms finish parameter estimation we compute and report the accuracy. The accuracy is computed as total number of examples classified as the correct label divide by number of examples in the test data, 2000.

### 2.6 Summary

In this chapter the data sets and tasks on which we have conducted our experiments were explained. These tasks include the named entity classification task of Collins and Singer (1999) and CoNLL 2003 shared task, Tjong Kim Sang and De Meulder (2003). The latter data set was extracted by us for the purpose of this research. The other one is the word sense disambiguation where given a word with two senses in a sentence, the task is to identify the correct sense of the word.
Chapter 3

Algorithms

In this chapter the semi-supervised learning models and algorithms used in this research are discussed in detail.

3.1 Training Algorithms

3.1.1 The Yarowsky algorithm

The Yarowsky algorithm by Yarowsky (1995) is a self-training bootstrapping method based on decision lists. A decision list (DL) (Rivest (1987)) is an ordered list of rules, feature and the corresponding label. There is a score associated with each rule and the list is ordered based on these scores. Different algorithms associate different scores with DL rules. More details on this is provided when the algorithm is discussed. One can think of each rule as a function $r : f \rightarrow y$ where $f$ is a feature in input $x$ and $y$ is the predicted output label predicted by this rule. To label an example $x$, a DL will use the rule with the highest score which matches one of the features in $x$ and return the label from that rule. The DL leaves input $x$ unlabelled if no rule matches any feature in $x$.

The original Yarowsky algorithm is shown in table 1. The algorithm starts with a set of “seed rules”. These initial seed rules are added to the DL. At each iteration the algorithm generates the labels for the input data based on the decision list. After finding the labels a new decision list is extracted from this set of labelled instances. The algorithm halts when the training parameters are held constant.

To learn scores for the rules in decision list, Yarowsky uses smoothed precision. For a rule $f \rightarrow y$, the (un-smoothed) precision is the probability of an input $x$ which has feature $f$ to be labelled as $y$:

$$q_f(y) = \begin{cases} 
\frac{|\Lambda_{fy}|}{|\Lambda_f|} & \text{if } |\Lambda_f| > 0 \\
1/L & \text{otherwise}
\end{cases}$$

(3.1)
where $\Lambda_f$ is the set of input instances $x$ which possess feature $f$, and $\Lambda_{fy}$ is the set of instances with feature $f$ and label $y$.

The smoothed precision for rule $f \mapsto y$ is defined as follows. The $\epsilon$ is a small positive number used for smoothing the distribution.

$$q(y|f) = \frac{|\Lambda_{fy}| + \epsilon}{|\Lambda_f| + L\epsilon}$$  \hspace{1cm} (3.2)

As a result the scores for DL can be computed from $q(y|f)$ either by using this value or after normalization. Figure 3.1 shows an example decision list from training the algorithm on the named entity classification task of Collins and Singer (1999). Figure 3.2 shows the result of labelling sample input data using this decision list.

Algorithm 1: The generic Yarowsky algorithm.

Input: Unlabelled examples $X$ and an initial seed DL $\theta^0$

1: for $t = 1, 2, ...$ do
2: Generate labels $Y^{(t)}$ using DL $\theta^{(t-1)}$
3: Extract a new DL $\theta^t$ by using labels $Y^{(t)}$ with a score over a threshold $\zeta$
4: if $Y^{(t)} = Y^{(t-1)}$ then stop
5: end if
6: end for

Output: The final decision list

3.1.2 Collins and Singer (1999)’s Yarowsky-cautious algorithm

Collins and Singer (1999) introduce a variant to Yarowsky algorithm, called Yarowsky-cautious. With cautiousness, for each label $y$, only the the top $n$ rules $f \mapsto y$ over the threshold get added to the DL. Additionally the threshold $\zeta$ is compared with the un-smoothed score $|\Lambda_{fy}|/|\Lambda_f|$ instead of the smoothed scores but in the decision list the entries are sorted based on smoothed score. The limit on the number of added rules, $n$, starts at $n_0$ and is incremented by $\Delta n$ at each iteration. The seed rules are also added to the new decision list. In order to achieve a better recall at the final iteration the Yarowsky-cautious does another iteration without the threshold or rule trimming, (Whitney (2012)).

3.1.3 EM algorithm

EM algorithm, Dempster et al. (1977), is a common approach to semi-supervised and unsupervised learning. Here we have used the EM algorithm to train a Naive Bayes model similar to Collins and Singer (1999). In this model the joint probability of an instance $X_i$ and its corresponding label $y_i$ is written as

$$P(y_i, X_i) = P(y_i) \cdot P(m_i) \prod_{j=1}^{m_i} P(x_{ij}|y_i)$$  \hspace{1cm} (3.3)
Table 1 shows the output of the Yarowsky algorithm on the named entity task of Collins and Singer (1999).

Figure 3.1: A sample decision list taken from the Yarowsky algorithm on the named entity task of Collins and Singer (1999).

Figure 3.2: Few instances from the named entity data and the corresponding label from the decision list of Figure 3.1. NA indicates that the decision list could not generate a label for the instance hence the label is not available.
where $m_i$ is the number of features for instance, $X_i$. An advantage of this model is that although each feature $x_{ij}$ is a binary random variable, the data is sparse (few features in each instance) while there are a large number of features. This will decrease the computational cost since the time complexity of EM algorithm is dependent on the dimension size of the data.

We now explain our approach to find the estimates. Since the experiments are done on the same data set as Collins and Singer (1999) we use the same set of initial seed rules. The initial parameters $\theta^0$ are learnt from the set of seed-labelled examples. These parameters include the prior on class labels $P_0(y)$, probability of observing a feature given the class label $P(x|y)$, and $P(m)$. Let $Count(y)$ be the number of examples labelled as class $y$ and $Count(y,x)$ the number of examples that are labelled as class $y$ and have feature $x$. We then have

$$P_0(y) = \frac{\text{Count}(y)}{n}$$

$$P_0(x|y) = \frac{1 + \text{Count}(y,x)}{|F| + \text{Count}(y)}$$  \hspace{1cm} (3.4)$$

where $F$ is the set of all features. Add-one smoothing is used to avoid having zero probability for features that are not seen in the seed-labelled set. The initial seed rules for the named entity classification task of Collins and Singer (1999) is shown in Figure 2.4. Note that since the number of features in an example, $m_i$, is fixed, its distribution is computed and kept fixed throughout the algorithm.

The next step is to find the ML parameter estimates. Based on the naive Bayes model the log likelihood of the observed data $X$ is

$$L(\theta) = \sum_{X_i \text{seed-labelled}} \log(P(y_i, X_i|\theta)) + \sum_{X_i \text{unlabelled}} \log(\sum_{y=1}^{k} P(y, X_i|\theta))$$  \hspace{1cm} (3.5)$$

Since the ML estimates can not be found analytically, the EM algorithm is used to hill climb to a local maximum of the log-likelihood function $L(\theta)$. At each iteration $t$ the new parameter values $\theta^t$ are computed as

$$P(y|\theta^t) = \frac{\sum_{i=1}^{n} \delta(y|i)}{n}$$

$$P(x|y; \theta^t) = \frac{\sum_{i:x \in X_i} \delta(y|i)}{\sum_{i} \delta(y|i)}$$  \hspace{1cm} (3.6)$$
where $\delta(y|i)$ is defined as

$$
\delta(y|i) = \begin{cases} 
0, & \text{if } X_i \in \text{seed-labelled set and } y_i \neq y. \\
1, & \text{if } X_i \in \text{seed-labelled set and } y_i = y. \\
\frac{p(y,X_i;\theta^{t-1})}{\sum_{y'=1}^{k} P(y',X_i)}, & \text{O.W.}
\end{cases}
$$

(3.7)

The difference in the learning part of our approach with the baseline system in Collins and Singer (1999) is that Collins and Singer (1999) use random initial values for parameter $\theta$ but we use values which are computed on the seed-labelled examples. The algorithm is shown in detail as algorithm 2. In the initialization step, the initial set of parameters are computed using the equations 3.4. Steps 2 and 3 show the E step of the EM algorithm where the conditional probabilities are computed using the equation 3.7. The M step of the algorithm is computing the new set of parameters using equations 3.6.

**Algorithm 2** The EM training algorithm. $Count(y)$ is the number of examples seed-labelled as class $y$ and $Count(y, x)$ is the number of examples that are seed-labelled as class $y$ which have feature $x$. $F$ is the set of all features.

**Initialize:**

$q^0(y) \leftarrow \frac{Count(y)}{n}$  
$q^0(x|y) \leftarrow \frac{1+Count(y,x)}{|F|+Count(y)}$, $\forall x \in F$

1: for $t = 1, \ldots, t_{max}$ do  
2: for $i = 1, \ldots, n$ for $y = 1, \ldots, k$ do  
3: $\delta(y|i) = p(y|X_i;\theta^{t-1}) = \frac{q^{t-1}(y) \prod_{y' \neq y} q^{t-1}(X_{ij}|y)}{\sum_{y'=1}^{k} q^{t-1}(y) \prod_{y' \neq y} q^{t-1}(X_{ij}|y)}$  
4: end for  
5: Update: $q_t(y) = \frac{1}{n} \sum_{i=1}^{n} \delta(y|i)$, $q_t(x|y) = \frac{\sum_{i:X_{ij}=x} \delta(y|i)}{\sum_i \delta(y|i)}$  
6: end for

**Output:** Parameter values $q^{t_{max}}(y) = p(y)$ and $q^{t_{max}}(x|y) = p(x|y)$.

### 3.1.4 Co-perceptron algorithm

The perceptron algorithm, introduced by Rosenblatt (1988), is a learning algorithm for supervised learning of binary classifiers. The perceptron learns a linear decision boundary by updating a set of weights:

$$
f(x) = \text{step}(W^T x)
$$

(3.8)

where
\[
\text{step}(y) = \begin{cases} 
1 & \text{if } y \geq 0 \\
0 & \text{O.W.}
\end{cases}
\] (3.9)

The Perceptron algorithm can be used in two different settings: batch learning and online learning. In online learning the input data is a stream of examples which upon arrival, needs to be classified. In the traditional setup for online learning there is no distinction between the training and test set. On the other hand batch learning is usually an easier task, where the algorithm can try to optimize an objective function over all the training data by using different methods like gradient decent (if the objective function is differentiable) or an iterative approach.

Given the training data \(X = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\), perceptron considers the number of misclassified examples as an objective function. Algorithm 3 illustrates the steps needed to find the \(W\) that best separates the data. At each iteration the algorithm computes the label for an instance (step 3) and if the generated label is not correct it will change the weight vector accordingly (steps 5 and 7).

Algorithm 3 The perceptron algorithm. The input \(X\) is a set of vectors in \(\{0, 1\}^d\)

1: Initialize: Set \(W_0^j = 0\) or some random value
2: for \(t = 1, \ldots, t_{\text{max}}\) do
3:   for \(i = 1, \ldots, n\) do
4:     \(y = \text{step}(W^T X_i)\)
5:     if \(y = -1\) and \(y_i = 1\) then
6:       \(W_j^t \leftarrow W_j^{t-1} + X_i\)
7:     end if
8:     if \(y = 1\) and \(y_i = -1\) then
9:       \(W_j^t \leftarrow W_j^{t-1} - X_i\)
10:   end for
11: end for

Since in unsupervised and semi-supervised learning we do not have access to the labels, the perceptron algorithm shown in Algorithm 3 cannot be used. In order to apply the perceptron to semi-supervised learning we need to make some changes to the objective function and the algorithm itself. In this section we talk about Co-perceptron algorithm as a discriminative approach to semi-supervised learning. Algorithm 4 shows the co-perceptron algorithm used for the named entity task of Collins and Singer (1999).

Co-perceptron is a co-training algorithm that can be used for semi-supervised learning tasks. Since some of the instances in the training set may be unlabelled, co-training techniques require two views of the data, for example spelling and context views for the named entity classification task. By training a separate classifier on each of two views, co-training algorithms try to find a set of parameters that will make two views agree on a label for the
input. The idea is that both views can represent the data and can be used separately to find a classifier. With co-perceptron two linear classifiers are trained on two views of the data. The weights are initialized to zero or a random value. The algorithm then tries to minimize the number of data points that cause disagreement between the two views. As stated before, since this function is not convex, the algorithm will minimize it in a number of iterations. At each iteration both views generate a label for an input and based on the input being labelled or unlabelled different steps will be taken. These steps are shown in Algorithm 4.

**Algorithm 4 Co-perceptron algorithm for the named entity task**

Initialize: $w_{j,y}^v = 0 \text{ or } 1$, [selected at random], $v = 1, 2$.

1: for $t = 1, \ldots, t_{\text{max}}$ do
2: for $i = 1, \ldots, n$ do
3: $y_1 = \arg\max_y (w_{j,y}^1 \cdot X_{1,i,j})$
4: $y_2 = \arg\max_y (w_{j,y}^2 \cdot X_{2,i,j})$
5: $y = \arg\max_y (w_{j,y}^1 \cdot X_{1,i,j} + w_{j,y}^2 \cdot X_{2,i,j})$
6: if $X_i \in L$ then
7: if $y^v \neq Y_i$, $v = 1, 2$ then
8: Update: $w_{j,y}^v \leftarrow w_{j,y}^v - 1$
9: end if
10: else
11: if $y_1 \neq y_2$ then
12: Update: $w_{j,y}^v \leftarrow w_{j,y}^v - \epsilon, v = 1, 2$
13: Update: $w_{j,y}^v \leftarrow w_{j,y}^v + \epsilon, v = 1, 2$
14: end if
15: end if
16: end if
17: end for
18: end for

**3.1.5 DL-CoTrain algorithm**

DL-CoTrain (Collins and Singer (1999)) is another bootstrapping method, which is shown as Algorithm 5. The algorithm uses two disjoint views of the features in the data and alternatively trains a decision list for each view. At each step the algorithm extracts a decision list for each view and then using a decision list from a view, it generates the labels for the other view. At the end the two decision lists are combined and ordered together using the scores for their rules. Similar to Yarowsky-cautious a final retraining step is done where a single labelling is generated and a decision list is extracted. This list gets returned as the output of the algorithm and can be used to classify new examples.
Algorithm 5 Collins and Singer (1999)’s DL-CoTrain algorithm.

**Inputs:**
training data $X$, two sets of features (views) $V_0$ and $V_1$ and a seed DL $\theta^{(0)}$

1. let $\theta^{(0,0)} = \theta^0 : V_0$
2. **for** $t = 1, \ldots, t_{max}$ or till convergence **do**
3. **for** view $(v,v') \in \{(0,1), (1,0)\}$ **do**
4. Use $\theta^{(t-1,v)}$ to $X$ to generate labels $Y^{(t,v')}$
5. Extract a new DL $\theta^{(t,v')}$ from $Y^{(t,v')}$ by using only feature from $V_{v'}$ with a score above the threshold $\zeta$
6. Add the initial seed rules $\theta^{(0,v')}$ to $\theta^{(t,v')}$
7. **end for**
8. **end for**
9. Combine two DLs $\theta^{(t,0)}$ and $\theta^{(t,1)}$ to generate the labels $Y^{(t)}$ for input examples $X$
10. Extract a final DL $\theta$ from $Y^{(t)}$ and add all seed rules $\theta^{(0)}$

3.1.6 EM algorithm with posterior regularization

The generative models that are used for semi-supervised learning tasks are simplified models of the underlying phenomena; for example the Naive Bayes model discussed in section 3.1.3. These models are usually estimated by maximizing the likelihood of the observations through a few iterations of the EM algorithm. One of the problems that arises with semi-supervised and unsupervised tasks is that the model may end up explaining an irrelevant correlation in input data. In order to capture the intended distribution, controlling the latent distribution through EM iterations is necessary. Ganchev et al. (2010) have proposed a framework which allows expressing constraints on latent variables’ distributions. These constraints usually come from the knowledge about the structure of the data. For example for the task of POS tagging it is known that most of the sentences have at least one verb. Using this information would help in finding a better distribution of POS tags.

Here again we use two views of the input data, same as the views in section 3.1.4. By assuming that each view can separately be a good classifier for the task at hand, the framework proposed by Ganchev et al. (2010) can be used to regularize the distributions learnt from each view by penalizing disagreement on unlabelled instances.

The input to the algorithm is the training instances in the form of $(X_1, X_2)$, representing two views of the data. The same Naive Bayes model described in Equation 3.3 is used for both views. The likelihood function is defined as follows:

$$\min_{\theta} L_1(\theta_1) + L_2(\theta_2) + c\mathbb{E}_U[B(p_1(\theta_1), p_2(\theta_2))]$$  \hspace{1cm} (3.10)

where $L_i$ is the negative likelihood of each view, $\mathbb{E}_U[B(p_1(\theta_1), p_2(\theta_2))]$ is the expected value of Bhattacharyya distance (Kailath (1967)) between the two distributions on the unlabelled instances and $c$ is a constant defining the relative weight of the unlabelled data. Proposition
3.10 can be written as follows:

$$\min_{\theta} L_1(\theta_1) + L_2(\theta_2) + cE_{\mathcal{U}}[\min_{q \in Q(x)} KL(q(y_1,y_2) || p(y_1)p(y_2))]$$  

(3.11)

where $Q = q : E_q[\delta(y_1 = y) - \delta(y_2 = y) = 0 \forall y]$. The proof can be found in Ganchev et al. (2008).

The general EM with posterior regularization is shown in algorithm 6. When the only constraint in the model is to minimize the KL distance between distributions from two views, the step 4 in algorithm 6 can be computed in closed form. It is shown in algorithm 7. In order to train the models using EM algorithm we compute the initial parameters by counting the number of seed-labelled examples, similar to training with EM in section 3.1.3, and use add-one smoothing for each view. Then we use the iterative algorithm in Algorithm 7 to minimize the likelihood function 3.11.

Algorithm 6 General definition of EM algorithm with Posterior Regularization.
1: $\theta_v \leftarrow \min_{\theta} L_v(\theta_v), v = 1, 2$
2: for $i = 1, \ldots, n$ do
3: \quad $q(y_1, y_2|x) \leftarrow \text{agree}(p_1(y_1|x), p_2(y_2|x)) \forall x \in U$
4: \quad $\theta_v \leftarrow \min_{\theta} L_v(\theta) - cE_{x \sim U, y_v \sim q}[\log p_v(y_v|x; \theta)], v = 1, 2$
5: end for

Algorithm 7 Our Implementation of EM algorithm with Posterior Regularization for Multi-View Learning.

Initialize:
\begin{align*}
q^0_v(y) &\leftarrow \frac{\text{Count}(y)}{n}, v \in \{0, 1\} \\
q^0_v(x|y) &\leftarrow \frac{1}{|F^{\neg y}| + \text{Count}(y)}, \forall x \in F, v \in \{0, 1\}
\end{align*}

1: for $t = 1, \ldots, t_{\text{max}}$ do
2: \quad for $i = 1, \ldots, n$ for $y = 1, \ldots, k$ do
3: \quad \quad $\delta_v(y|i) = p(y|X^i_t; \theta_v^{t-1}) = \frac{q_v^{t-1}(y) \prod_j q_{v,j}^{t-1}(x_{ij}|y)}{\sum_{y=1}^k q_v^{t-1}(y) \prod_j q_{v,j}^{t-1}(x_{ij}|y)}, v \in \{0, 1\}$
4: \quad end for
5: Update: $q^t_v(y) = \frac{1}{n} \sum_{i=1}^n \delta_v(y|i) \delta_{\gamma_1}(y|i), q^t_{v,j}(x|y) = \sum_{i}^{\gamma_q \delta_{\gamma_1}(y|i)} \sum_{y}^{\delta_k(y|i)} \sum_{j}^{\gamma_{\delta_{\gamma_1}(y|i)}}$
6: end for

The term $\text{agree}(p_1, p_2)$ is defined to be the minimizer of proposition 3.11.

### 3.2 Decoding Algorithms

After finding the best parameters for the distribution, we are going to use the distribution to classify a new instance $x$ as one of the labels. In the case of named entity problem there are three classes, Organization, Person, and Location. The main focus of this study has
been on different approaches for the decoding stage of a generative approach which is learnt in a semi-supervised fashion.

### 3.2.1 Using full joint distribution

The straightforward approach that is usually being used to decode after a generative model is learnt is to select the label which maximizes the joint probability. In the case of named entity task of Collins and Singer (1999), they use \( \text{argmax}_y P(x, y) \) where \( P(x, y) \) is defined in equation 3.3. The results of this approach can be found in Table 4.1.

### 3.2.2 Using most “informative” feature(s)

Before explaining the next classifier function, it is useful to recall the set of features in the named entity classification task. Collins and Singer (1999) extracted a set of proper nouns that met a certain criteria from a set of parsed sentences. For each proper noun, a set of spelling features (related to the actual phrase) and a set of contextual features (related to the context the noun had appeared in) are extracted.

To find the label for a test example we then use only the feature that best represents the corresponding proper noun, the \( \text{full-string}(x) \) feature that contains the whole phrase. Thus, the label for a test example \( x \) is defined as

\[
y = \text{argmax}_y P(f'_x | y)
\]

where \( f'_x \) is the feature that best represents the test example \( x \). We call this feature the ‘most informative feature’. In the experiments we show that using such feature in decoding will lead to higher classification accuracy.

Since the term “a single feature that best represents an instance” may not be well defined, we may select the most confident feature. Let us define a function \( h(x, y) \) which is an estimate of the conditional probability \( P(y|x) \). Using Bayes rule we get a definition of \( h(x, y) \) in terms of a parameter that is computed by EM: \( P(x|y) \).

\[
h(x, y) = \frac{P(y|x)}{\sum_{y=1}^{k} P(x_j|y)} \cdot P(y)
\]

For a single feature \( x_j \) and a specific label \( y_i \) we can define \( h(x_j, y_i) \):

\[
h(x_j, y_i) = P(y_i) \cdot \frac{P(x_j|y_i)}{\sum_{y=1}^{k} P(x_j|y)}
\]
We assume the prior distribution is uniform (since the likelihood is being maximized over the training data set and not the test set). As in the Yarowsky algorithm, $h(x_j, y_i)$ can be viewed as a rule $x_j \rightarrow y_i$ ranked by their strength: $h(x_j, y_i)$. The most informative feature for input $x$ is then defined as:

$$f'_x = \arg\max_{x_j, y_i} h(x_j, y_i) \quad (3.15)$$

The output of this argmax is interpreted as a decision list rule $x \rightarrow y$ with weight $h(x, y)$ just as in the Yarowsky algorithm.

### 3.3 Summary

In this chapter we explained the algorithms used in semi-supervised learning tasks which are related to our work. The Yarowsky algorithm is a self-training bootstrapping method based on decision lists. In Yarowsky cautious, by limiting the number of rules that get added to the DL at each iteration, the Yarowsky method will not suffer from possible classification errors in early iterations which then might propagate to later iterations. The DL-CoTrain is another bootstrapping method which uses two views of the data, spelling and contextual. Each view has a separate DL associated with it. At each iteration the decision list for one view is used to generate labels for the other view. The assumption is that the two views represent the same underlying distribution and can train each other. Co-perceptron is another multi-view semi-supervised learning algorithm. In the co-perceptron algorithm, at each iteration the discriminative classifiers of both views generate labels for input data and the algorithm tries to make two views to agree on the output label. The EM algorithm and EM algorithm with posterior regularization find the maximum likelihood parameter estimates for the input data. Later in this chapter we introduced the term “most informative feature” and defined it. We introduced multiple possible options for choosing such feature(s), either without supervision or one chosen by a human expert.
Chapter 4

Experiments

In this chapter we describe and analyze the results of different learning and decoding techniques in more detail.

4.1 Experimental setup

Where applicable we have used add-one smoothing. Unlike Whitney and Sarkar (2012) we do not use different weights for labelled and unlabelled instances while training for the named entity classification task of Collins and Singer (1999). For the CoNLL 2003 data set, since we did not have access to the data set used by Ganchev et al. (2008) we used weight of 0.01 for unlabelled instances and 1 for the labelled instances to achieve the accuracy reported by Ganchev et al. (2008). For the parameter $\epsilon$ in the co-perceptron algorithm we set it to 0.01 and kept it constant through all iterations. In our experiments we refer to one of the decoding techniques as "whole phrase feature" decoding. This whole phrase feature for the named entity classification tasks is the feature representing the actual noun phrase that we intend to classify. For example in Figure 2.3, the whole phrase features are "X0_Barack-Obama" and "X0_USA" respectively. To report the statistical significance we use a approximate randomization significance test (using the software provided by Padó (2006)) and check that $p < 0.05$.

4.2 NYT Results

The clean accuracy of different learning and decoding techniques for the named entity classification task of Collins and Singer (1999) are shown in tables 4.1 and 4.2. Table 4.1 contains the results of different decoding techniques introduced earlier while table 4.2 contains the results of previous works on the same data set. EM algorithm guarantees that the likelihood of the data will increase at each iteration. The log likelihood of the training
<table>
<thead>
<tr>
<th>Training algorithm</th>
<th>Joint distribution</th>
<th>Most confident feature</th>
<th>Whole phrase feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM</td>
<td>88.6</td>
<td>89.5</td>
<td>91.11</td>
</tr>
<tr>
<td>Hard EM</td>
<td>79.25</td>
<td>81.07</td>
<td>85.86</td>
</tr>
<tr>
<td>EM with PR</td>
<td>83.24</td>
<td>83.58</td>
<td>88.14</td>
</tr>
</tbody>
</table>

Table 4.1: Accuracy for different learning methods on the named entity classification task of Collins and Singer (1999). The Decoding columns shows the function $f(x, y)$ that we have computed the $\text{argmax}_y f(x, y)$ on. For the joint distribution it will be $f(x, y) = P(x, y)$ and the most confident feature refers to $f(x, y) = P(\hat{f}|y)$ where $\hat{f}$ is computed by equation 3.15. The result of using only the whole phrase of proper nouns to classify each is shown under “Whole phrase feature” column.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy (clean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM (Collins and Singer (1999))</td>
<td>83.1</td>
</tr>
<tr>
<td>Yarowsky</td>
<td>81.19</td>
</tr>
<tr>
<td>Yarowsky cautious</td>
<td>91.11</td>
</tr>
<tr>
<td>DL-CoTrain</td>
<td>91.3</td>
</tr>
<tr>
<td>Yarowsky prop (Whitney and Sarkar (2012))</td>
<td><strong>92.47</strong></td>
</tr>
</tbody>
</table>

Table 4.2: Accuracy of different learning methods (previous works) on the named entity classification task of Collins and Singer (1999).

data for the first 10 iterations is shown in Figure 4.4. Figure 4.5 shows the (clean) accuracy for different decoding algorithms on the named entity data set.

The classification accuracy for the co-perceptron algorithm at first 10 iterations is shown in Figure 4.1. Figure 4.2 shows the number of instances where in the co-perceptron algorithm the two views did not agree upon. The condition for terminating the co-perceptron algorithm is for the two views to reach a state where the number of disagreements does not change during an iteration. In Figure 4.3, the total number of seed-labelled examples in which the views do not generate the right label is shown.

### 4.3 CoNLL 2003 Results

We conducted similar experiments to ones done on the task of Collins and Singer (1999) on the data set extracted from CoNLL 2003 shared task. Table 4.3 contains the results of different decoding techniques where we add the test data (without the labels) to the training data (transductive learning) and without using the test set for training (inductive learning). Table 4.4 shows the results of previous works on the same data set. The log likelihood of the training data for the first 10 iterations is shown in Figure 4.6. Figure 4.7 shows the (clean) accuracy for different decoding techniques at different iterations.
Figure 4.1: Accuracy (clean) of co-perceptron algorithm on the named entity classification task of Collins and Singer (1999) at different iterations. Three lines represent different decoding methods: using spelling features only (view 1), using contextual features only (view 2), or using all features.

<table>
<thead>
<tr>
<th>Training algorithm</th>
<th>Joint distribution</th>
<th>Most confident feature</th>
<th>Whole phrase feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM</td>
<td>81.05</td>
<td>81.80</td>
<td>84.50</td>
</tr>
<tr>
<td>EM (inductive)</td>
<td>71.55</td>
<td>74.00</td>
<td>78.50</td>
</tr>
<tr>
<td>EM with PR</td>
<td>79.50</td>
<td>80.80</td>
<td>84.60</td>
</tr>
</tbody>
</table>

Table 4.3: Accuracy for different learning methods on the named entity classification task for CoNLL 2003 data set. The Decoding columns shows the function $f(x, y)$ that we have computed the argmax$_y f(x, y)$ on. For the joint distribution it will be $f(x, y) = P(x, y)$ and the most confident feature refers to $f(x, y) = P(\hat{f}|y)$ where $\hat{f}$ is computed by equation 3.15. The result of using only the whole phrase of proper nouns to classify each is shown under “Whole phrase feature” column.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Accuracy (clean)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yarowsky</td>
<td>72.90</td>
</tr>
<tr>
<td>Yarowsky (inductive)</td>
<td>70.70</td>
</tr>
<tr>
<td>Yarowsky cautious</td>
<td>62.75</td>
</tr>
<tr>
<td>Yarowsky cautious (inductive)</td>
<td>62.55</td>
</tr>
<tr>
<td>DL-CoTrain</td>
<td>70.35</td>
</tr>
</tbody>
</table>

Table 4.4: Accuracy of different learning methods (previous works) on the named entity classification task for CoNLL 2003 data set.
4.4 WSD Results

Our final set of experiments are done on word sense disambiguation task of Eisner and Karakos (2005). Table 4.5 contains the results of different decoding methods while table 4.6 shows the results of previous works on the same data set.

4.5 Analysis

Abney (2004) assumes that there is no threshold and once an input \( x \) gets labelled it would not become unlabelled again. He shows that under these assumptions the Yarowsky algorithm minimizes an upper bound for the negative log likelihood of the data. Figure 4.1 defines the log likelihood of the data.

\[
l(\theta) = \sum_x \sum_j \phi_x(j) \log \pi(j|x; \theta) \tag{4.1}
\]

where
Figure 4.3: Number of seed-labelled examples in the training set of named entity task (Collins and Singer (1999)) which the label generated by the view is not the correct label. Note that in order to keep the function always decreasing one can try to decrease the $\epsilon$ amount in later iterations.

<table>
<thead>
<tr>
<th>Drug</th>
<th>Train algorithm</th>
<th>joint</th>
<th>confident</th>
<th>whole</th>
</tr>
</thead>
<tbody>
<tr>
<td>EM</td>
<td>53.62</td>
<td>57.51</td>
<td>58.81</td>
<td></td>
</tr>
<tr>
<td>Hard EM</td>
<td>51.14</td>
<td>51.30</td>
<td>53.63</td>
<td></td>
</tr>
<tr>
<td>Sentence</td>
<td>EM</td>
<td>76.41</td>
<td>75.00</td>
<td>64.65</td>
</tr>
<tr>
<td>Hard EM</td>
<td>78.83</td>
<td>78.16</td>
<td>70.50</td>
<td></td>
</tr>
<tr>
<td>Hard EM</td>
<td>53.39</td>
<td>65.24</td>
<td>61.75</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5: Accuracy of EM algorithm with different decoding methods for the word sense disambiguation tasks. The Decoding column shows the function $f(x, y)$ that we have computed the $\text{argmax}_y f(x, y)$ on. For the joint distribution it will be $f(x, y) = P(x, y)$ and the most confident feature refers to $f(x, y) = P(\hat{f} | y)$ where $\hat{f}$ is computed by equation 3.15.
Figure 4.4: Log likelihood of parameters learnt by EM algorithm over named entity (Collins and Singer (1999)) training data.

Figure 4.5: Accuracy of EM algorithm with different decoding methods for named entity classification task of Collins and Singer (1999).
Figure 4.6: Log likelihood of parameters learnt by EM algorithm over CoNLL 2003 named entity training data.

Figure 4.7: Accuracy of EM algorithm with different decoding methods for named entity classification task for CoNLL 2003 data set.
Table 4.6: Accuracy of different learning methods (previous works) on the word sense disambiguation tasks of Eisner and Karakos (2005) (Drug, Land, and Sentence).

<table>
<thead>
<tr>
<th></th>
<th>EM (Whitney and Sarkar (2012))</th>
<th>Yarowsky</th>
<th>Yarowsky cautious</th>
<th>DL-CoTrain</th>
</tr>
</thead>
<tbody>
<tr>
<td>drug</td>
<td>55.96</td>
<td>55.70</td>
<td>54.40</td>
<td>59.59</td>
</tr>
<tr>
<td>land</td>
<td>32.86</td>
<td>79.03</td>
<td>79.10</td>
<td>78.36</td>
</tr>
<tr>
<td>sentence</td>
<td>67.88</td>
<td>62.91</td>
<td>78.64</td>
<td>68.16</td>
</tr>
</tbody>
</table>

\[
\phi_x(j; \theta^t) = \begin{cases} 
[j = \arg\max_y \pi(y|x; \theta^t)] & x \in \Lambda(\text{labelled}) \\
\frac{1}{2} & \text{O.W.}
\end{cases}
\]

\[
\pi(j|x; \theta^t) = \arg\max_{f \in x} \{ DL(f, j; \theta^t) \}
\]

\[
DL(f, j; \theta^t) = \frac{\sum_{x_1(f \in x)} \phi_j(x; \theta^{t-1})}{\sum_{x_1} \sum_{f \in x} \sum_{y} \phi_y(x; \theta^{t-1})}
\]

The log likelihood function of 4.1 and the optimization done at each iteration is similar to the EM algorithm with two differences. First is that the Yarowsky algorithm computes and assigns a label to each instance at each iteration (similar to hard EM). The second difference is that unlike (hard) EM it does not use the join distribution \((p(y) \cdot p(m) \cdot \prod p(x_j|y))\) but uses the most confident feature.

In the previous results sections (sections 4.2, 4.3, and 4.4) we provide the clean test set accuracy for bootstrapping and generative algorithms. The bootstrapping methods which are used are the Yarowsky algorithm (a self training method) and DL-CoTrain (a multi view approach). Variations of EM algorithm are used to find the parameter estimates for a naive Bayes model.

On the NYT data set, table 4.1 and 4.2 show that our approach can achieve the same clean test accuracy as Yarowsky cautious algorithm (91%). Whitney and Sarkar (2012) reported the highest accuracy on the NYT data set by using their graph propagation algorithm (92.47%). Their result is not significantly better than the result of our generative approach. It is worth noting that Collins and Singer (1999) and Whitney and Sarkar (2012) both reported \(\approx 83\%\) for the accuracy of EM algorithm. The effect of similar initialization and decoding steps are shown for hard EM and EM with PR, showing the same increase in accuracy by using different decoding functions.
On the CoNLL data set, table 4.3 and 4.4 again show that our approach achieves higher accuracy with different decoding methods. As shown in table 4.4, the Yarowsky and DL-CoTrain algorithm does not achieve a comparable clean accuracy on the test set. It may be because of the fact that the data set is sparse, with possibly many features that are not relevant. To make it more clear, in the NYT named entity data set, the contextual features for a named entity were modifiers, selected after parsing the sentence. For the CoNLL data set the features are the adjacent words. Also in order to get an accuracy close to the one reported by Ganchev et al. (2008) we assigned a small weight (0.01) to unlabelled instances but Yarowsky algorithm does not use such different weights. Another thing to notice in table 4.4 is that the Yarowsky cautious algorithm achieves a lower accuracy than the generic Yarowsky algorithm. The reason may be that because of the higher sparsity, most rules in the DL get the same score of 1, especially during initial iterations. The Yarowsky cautious loses some of these rules when the DL gets trimmed.

The results of our approach on the WSD data sets are shown in table 4.5. Here we observe that for all but the sentence data set, using a decoding method different from the joint probability achieves higher accuracy. Our approach achieves a better accuracy on the drug data set and a significantly better accuracy on the land data set than the ones reported by Whitney and Sarkar (2012) for the EM algorithm. For the sentence data set, there is not a statistically significant difference between the result of our approach and the ones reported by Whitney and Sarkar (2012) for the EM algorithm. Since the size of the WSD data sets are small (as stated by Whitney (2012)) we put our focus on the named entity data sets which are relatively larger and did not investigate the few abnormalities.
Chapter 5

Conclusion and Future Work

In this research we studied a number of bootstrapping approaches, including the self-training Yarowsky algorithm, and instead of analyzing a variation of it, we use the well-known EM algorithm. We show that by using a similar initialization and decoding step as Yarowsky, the easy-to-analyze EM algorithm can achieve similar or better results than Yarowsky algorithm. In particular, we make three contributions in this research:

1. We provide evidence that using different initialization and decoding methods, a simple generative model can perform as well as the hard-to-analyze discriminative algorithms for the task of named entity classification and word sense disambiguation on multiple data sets.

2. We provide a new data set for semi-supervised learning extracted from the 2003 CoNLL shared task of Tjong Kim Sang and De Meulder (2003) which will be freely available to researchers.

3. We provide results from our implementation of several algorithms (EM algorithm, the EM algorithm with posterior regularization, Yarowsky algorithm, and co-perceptron), allowing consistent comparison.

For future work we intend to study the decoding step, specifically choosing the most informative feature, in more detail and its effect on the final accuracy. One may also study the application of this bootstrapping method on other tasks in NLP or other fields.
References


