Investigations into the Value of Labeled and Unlabeled Data in Biomedical Entity Recognition and Word Sense Disambiguation

by

Golnar Sheikhshabbafghi

M.Sc., Oregon Health & Science University, 2014
B.Sc., Sharif University of Technology, 2006

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Sciences

Declaration of Committee

Name:	**Golnar Sheikhshabbafghi**
Degree:	**Doctor of Philosophy**
Thesis title:	**Investigations into the Value of Labeled and Unlabeled Data in Biomedical Entity Recognition and Word Sense Disambiguation**
Committee:	**Chair:** Arrvindh Shriraman Associate Professor, Computing Science
	Anoop Sarkar Supervisor Professor, Computing Science
	Inanc Birol Committee Member Adjunct Professor, Computing Science
	Fred Popowich Committee Member Professor, Computing Science
	Angel Chang Examiner Assistant Professor, Computing Science
	Fei Xia External Examiner Professor Department of Linguistics University of Washington

Abstract

Human annotations, especially in highly technical domains, are expensive and time consuming to gather, and can also be erroneous. As a result, we never have sufficiently accurate data to train and evaluate supervised methods.

In this thesis, we address this problem by taking a semi-supervised approach to biomedical named entity recognition (NER), and by proposing an inventory-independent evaluation framework for supervised and unsupervised word sense disambiguation.

Our contributions are as follows:

- We introduce a novel graph-based semi-supervised approach to named entity recognition (NER) and exploit pre-trained contextualized word embeddings in several biomedical NER tasks.
- We propose a new evaluation framework for word sense disambiguation that permits a fair comparison between supervised methods trained on different sense inventories as well as unsupervised methods without a fixed sense inventory.

Keywords: Biomedical Named Entity Recognition, Word Sense Disambiguation, Word Sense Induction

To my family

Acknowledgements

I would like to thank the examining committee for taking the time to read my thesis and for providing insightful feedback; my supervisors Dr. Fred Popowich and Dr. Inanc Birol for guiding me through this at-times-arduous journey; and my family and friends for their continuous encouragement and for putting up with me when I was struggling.

I would also like to express my deepest gratitude towards my senior supervisor, Dr. Anoop Sarkar. I was incredibly lucky to be his student, not only because of his praiseworthy knowledge and unparalleled patience, but also because of his utmost investment in the success of his students. If I ever end up being in a position to mentor someone, I would aspire to be like him.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

We live in a time when unlabeled data is available like never before as more people make more data publicly available. Examples of such data include weblogs, vlogs, peer-reviewed or non-peer-reviewed academic papers, as well as photos and short texts publicly available on social media, among other things. High-quality annotated data, needed to decide how well a method performs on a given task, or to train supervised machine learning techniques is scarce, however.

Human annotated data is expensive and time-consuming to collect. This is even more intensified for domains where the annotators need to be highly skilled. In such cases, it is helpful to take approaches that need no annotated data, can stretch the little available annotated data, or can lower the cost by transforming tasks to ones that need different lower-cost annotated data.

In this thesis, we focus on two tasks where labeled data by highly skilled human annotators are needed for training and evaluation purposes. We take semi-supervised approaches to biomedical named entity recognition (NER) problem, thus stretching the use of existing annotated data by taking advantage of abundant unlabeled data. We also propose a sense-inventory independent evaluation framework for word sense disambiguation (WSD) and word sense induction (WSI), both making use of existing annotated data, and making it possible to expand the evaluation datasets using easier to obtain annotations.

This thesis is organized into two parts, one part focuses on semi-supervised approaches to biomedical NER, and the other, on our proposed approach to WSDI/WSI evaluation. In the rest of this chapter, we introduce these tasks and outline our contributions.

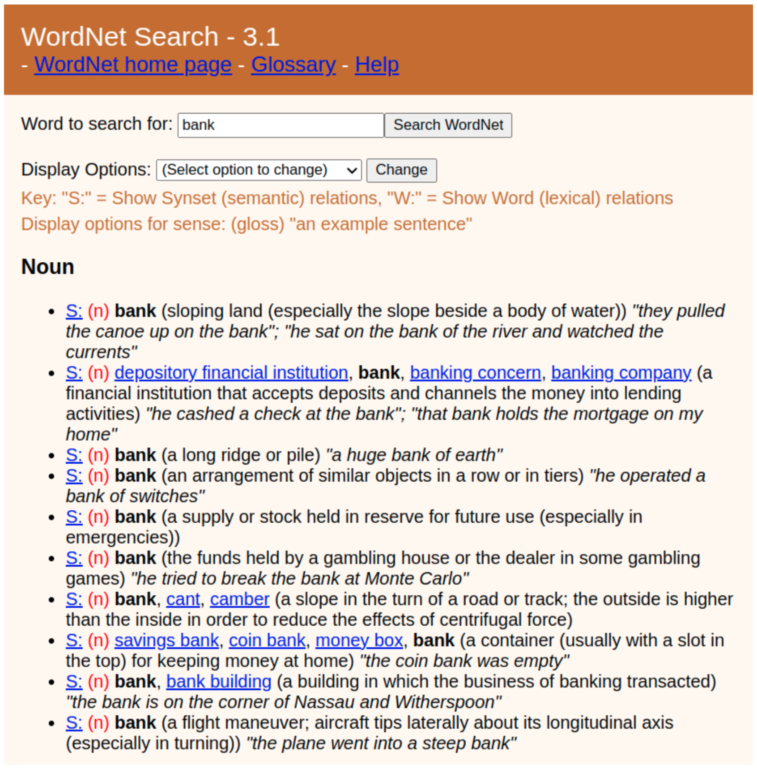
1.1 Tasks

The first task we are focusing on, is **biomedical NER**. Named entity recognition itself, refers to the task of detecting named entities such as people or organization names in a context. Biomedical NER refers to the task of detecting biomedical named entities such as names of genes, proteins, drugs, or diseases. For example, in the sentence "Golnar Sheikhshab is a student in Simon Fraser University, working with Dr. Anoop Sarkar. ", an NER method is expected to detect "Golnar Sheikhshab" and "Dr. Anoop Sarkar" as named entities of type *PERSON* and "Simon Fraser University" as a named entity of type *organization*. Going back to the biomedical domain, in the sentence "Recently, the mutation of lymphocyte adaptor protein (LNK or SH2B3) was detected in MPN.", there are three named entities, "lymphocyte adaptor protein", "LNK", and "SH2B3", all of the type *gene name*, and a biomedical NER should be able to detect them all.

Part II of this thesis focuses on evaluating **WSD** and **WSI** methods. Both of these tasks try to make sense of a particular word/phrase in context. For example, both WSD and WSI techniques are expected to detect that the word "bank" is used to convey different meanings(senses) in the following sentences:

- He sat on the *bank* of the river and watched the currents.
- He cashed a check at the *bank*.

The difference between WSD and WSI is that WSD assumes a given sense-inventory and is defined as a classification task, whereas WSI is sense-inventory independent and is defined as a clustering task. A WSD method takes a word-usage in a given context as input and attempts to pick the appropriate sense from a given sense-inventory like the one in Figure 1.1. WSI, on the other hand, takes several usages of a word in context as input and attempts to cluster them so that usages conveying the same meaning fall into the same cluster while usages conveying different meanings fall into different clusters.


WordNet Search - 3.1
- WordNet home page - Glossary - Help
Word to search for: bank
Search WordNet
Display Options: (Select option to change)
Change

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations
Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **bank** (sloping land (especially the slope beside a body of water)) *"they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents"*
- S: (n) depository financial institution, **bank**, banking concern, banking company (a financial institution that accepts deposits and channels the money into lending activities) *"he cashed a check at the bank"; "that bank holds the mortgage on my home"*
- S: (n) **bank** (a long ridge or pile) *"a huge bank of earth"*
- S: (n) **bank** (an arrangement of similar objects in a row or in tiers) *"he operated a bank of switches"*
- S: (n) **bank** (a supply or stock held in reserve for future use (especially in emergencies))
- S: (n) **bank** (the funds held by a gambling house or the dealer in some gambling games) *"he tried to break the bank at Monte Carlo"*
- S: (n) **bank**, cant, camber (a slope in the turn of a road or track; the outside is higher than the inside in order to reduce the effects of centrifugal force)
- S: (n) savings bank, coin bank, money box, **bank** (a container (usually with a slot in the top) for keeping money at home) *"the coin bank was empty"*
- S: (n) **bank**, bank building (a building in which the business of banking transacted) *"the bank is on the corner of Nassau and Witherspoon"*
- S: (n) **bank** (a flight maneuver; aircraft tips laterally about its longitudinal axis (especially in turning)) *"the plane went into a steep bank"*

Figure 1.1: WordNet senses for *bank* as a noun.

The go-to sense-inventory in English WSD is WordNET [Miller, 1995], a lexical resource developed at Princeton University since 1985, listing arguably all the senses of many English and non-English words, and providing definitions(glosses), examples, and relationships between them. For *bank as a noun*, for example, WordNet lists 10 senses shown in Figure 1.1.

WordNet, an example of an expensive and time-consuming human expert annotation, has an interesting drawback. Its senses are so fine-grained that humans struggle to distinguish between some of the listed senses for a given lemma. Figure 1.1 can give us an idea of how fine-grained the senses are. Regardless of this problem, almost all evaluation sets for WSD and WSI have been annotated with WordNet fine-grained senses. Part II of this thesis focuses on this problem and proposes an evaluation framework to rectify that.

1.2 Contributions

Our contributions in each of the two Parts are as follows.

1. Part I
   (a) We introduce GraphNER, a graph-based semi-supervised approach to Named Entity Recognition, and show its efficacy on two biomedical NER tasks.
   (b) We show the efficacy of contextualized word embeddings in several biomedical NER tasks.
2. Part II
   (a) We propose a new evaluation framework for WSD and its unsupervised counterpart, WSI, that allows comparing methods in both fields.
   (b) We show that our evaluation framework makes it possible to use alternative sense inventories such as OntoNotes instead of the current go-to WordNet sense inventory.
   (c) We show that our evaluation datasets highlight the strength of some WSD methods that consistently do well across datasets.
   (d) As part of our evaluation framework, we transform several existing annotated datasets for other tasks to a format that can be used for WSD/WSI evaluation, effectively introducing new evaluation datasets.

1.3 Thesis Outline

Chapter 2 provides the background on graph-based semi-supervised learning, laying the ground for Chapter 3 that introduces GraphNER, our graph-based semi-supervised approach to biomedical NER. Chapter 4 describes the other semi-supervised approach we took to biomedical NER taking advantage of contextual embeddings pre-trained on large unlabeled data.

In Chapter 5, we describe the current evaluation framework used for WSD methods as well as recent methods achieving state-of-the-art performance and comment on where the evaluation framework falls short. In Chapter 6, we do the same for WSI methods. Chapter 7 describes our proposed evaluation framework for both tasks and shows how it can be used to address some of the issues of the current evaluation frameworks. Finally, Chapter 8.1 outlines a few future directions.

Part I

Semi-Supervised Named Entity Recognition

Chapter 2

Graph Based Semi-supervised Learning for NLP

In many NLP applications, labeled data is scarce, un-available in the domain of interest, and too expensive to obtain. Abundant unlabeled data, on the other hand, is readily available. Semi-supervised learning (SSL) takes advantage of big unlabeled data sets as well as relatively small labeled datasets. Graph-based SSL is a special kind of SSL, based on the assumption that similar data points should have similar labels.

A graph is constructed whose vertices represent data points and whose edge-weights represent how strongly we believe the adjacent vertices (data points) should get the same label. The graph will connect labeled and unlabeled data points and each vertex is associated with a distribution over output labels (label distribution for short) that represents the current belief about its label. In NLP applications, often the data points are represented as n-grams, n consecutive words in a sentence, also known as shingles.

Having this graph that encodes the similarities between data points, the goal is to find label distributions for all vertices so that

1. for any labeled vertex v, the associated label distribution is as close as possible to its reference distribution obtained from the labeled data based on the number of times each data (point, label) pair appeared together;
2. adjacent vertices in the graph have similar label distributions;
3. the label distributions of all vertices comply with the prior knowledge if such knowledge exists.

There are two different settings in Graph-based SSL: transductive and inductive. In transductive settings, the graph is constructed over train and test sets, and the most probable label for any test data-point is chosen after propagation. For new test data, the graph should be re-constructed and labels should be propagated again. In inductive settings, on the other hand, a model such as a conditional random field is trained and can assign labels to new data-points, eliminating the need for constructing a graph or propagating labels for new unseen test data.

In the following sections, we elaborate on how to construct the graph and how to propagate the label distributions, describe two inductive methods that are especially interesting in NLP applications, and provide examples of NLP applications of graph-based SSL.

Since many of the papers we mention in this chapter have shown experiments on Part of speech (POS) tagging, we use this application as a running example whenever illustration is necessary. Given a sentence, the task of a POS tagger is to predict the part of speech of each word. An example of a POS annotated sentence is "The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.".

2.1 Graph Construction

A very important part of any application of graph-based SSL is to construct a graph that represents the data points as vertices and their similarities as edge weights in a way that one would expect similar vertices to get similar labels.

Unfortunately, there is no universal way of modeling the data as a graph. What vertices represent and how their similarities should be computed, are task-specific problems.

2.1.1 Graph construction for NLP

Many NLP applications adapt the graph construction method of Subramanya et al. [2010]. For the POS tagging task, Subramanya et al. suggested that vertices in the graph represent n-gram types (3-grams in their experiments). This choice of vertex type makes sense for their task because the POS tag of a word is believed to depend the most on the word itself and few words in its proximity.

They use a set of contextual feature types (shown in Table 2.1) and for any given vertex u compute the point-wise mutual information (PMI) between u and any feature instance present in the corpus. $PMI(u, f)$ is a measure of association between a vertex u and a feature instance f and is computed as follows. $PMI(u, f) = \log \frac{p(u,f)}{p(u)p(f)} = \log \frac{p(u|f)}{p(u)} = \log \frac{p(f|u)}{p(f)}$. As a result, each vertex u is represented by a PMI vector whose length is the number of all different feature instances in the corpus. The weight of an edge (u, v) is computed as the cosine similarity between the PMI vectors of u and v.

Description	Feature
Trigram + Context	$x_{-2}\ x_{-1}\ x_0\ x_1\ x_2$
Trigram	$x_{-1}\ x_0\ x_1$
Left Context	$x_{-1}\ x_{-2}$
Right Context	$x_1\ x_2$
Center Word	x_0
Trigram $-$ Center Word	$x_{-1}\ x_1$
Left Word + Right Context	$x_{-1}\ x_1\ x_2$
Left Context + Right Word	$x_{-2}\ x_{-1}\ x_1$
Suffix	$x_0[-3:]$

Table 2.1: Set of feature types for POS tagging

For example, if there are 1000 tokens of type “to book a" in the training data, and 100 of them are in the context of “how to book a flight", and the training data consists of 100,000 words, the feature vector of the vertex corresponding to “to book a" will have a value of $log(0.1/0.001)$ for feature “context= “how to book a flight"". Again, the choice of features is task-specific. Here, context features that go as far as 2 words in each direction, plus the suffix of the word itself, are relevant features for determining a word’s POS.

Figures 2.1 - 2.4 show some of the properties of such a graph. In figure 2.1 we can see labeled and unlabeled vertices represented as 3-grams. Solid lines are edges and dotted bi-directional arrows show the distance between un-connected vertices. Figure 2.2 shows how each of these vertices has a PMI feature vector associated with them. [to start a] and [to book a] end up being neighbors because they share 2 context features “R: a hotel" and “R: a program" (feature vector values are for illustration purposes only). Figure 2.3 shows how two vertices can be in the same connected component but not too close to each other. In this case, [you book a] and [you booked a] are distant, probably because of a large weight for the suffix feature type, and their POS tags should be different (VB vs. VBD). This illustrates the importance of using relevant feature types for a given task. Finally, figure 2.4 shows the extreme form of distance where verbs and nouns fall into different connected components.

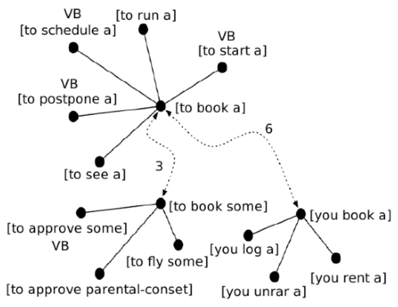
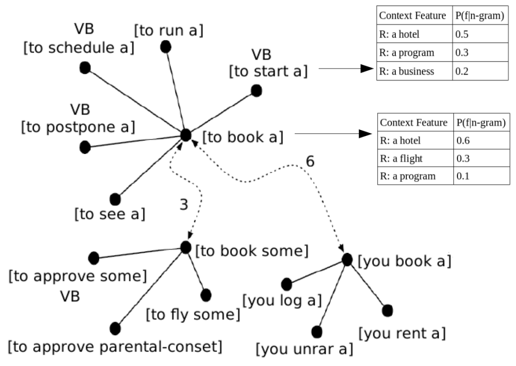


Figure 2.1: An excerpt of the graph Subramanya et al. [2010] constructed for the task of POS tagging.

An important issue in graph-based SSL is to avoid dense graphs. This plays an important role in keeping the graph-propagation tractable because the complexity of graph propagation depends on the number of edges. Also, it is intuitive to keep the graph sparse to prevent unrelated vertices from getting connected. Subramanya et al. suggest keeping up to k nearest neighbors for each vertex. They choose k to be 10.

2.1.2 Other graph construction methods

Although in most NLP applications of graph-based SSL the graph has been constructed similarly to Subramanya et al, there have also been other ways and some of them have interesting properties. Here, I describe three properties of two other graph construction methods.



Context Feature	P(f|n-gram)
R: a hotel	0.5
R: a program	0.3
R: a business	0.2

Context Feature	P(f|n-gram)
R: a hotel	0.6
R: a flight	0.3
R: a program	0.1

Figure 2.2: PMI feature vectors associated to vertices in the graph are the base for similarity calculation.

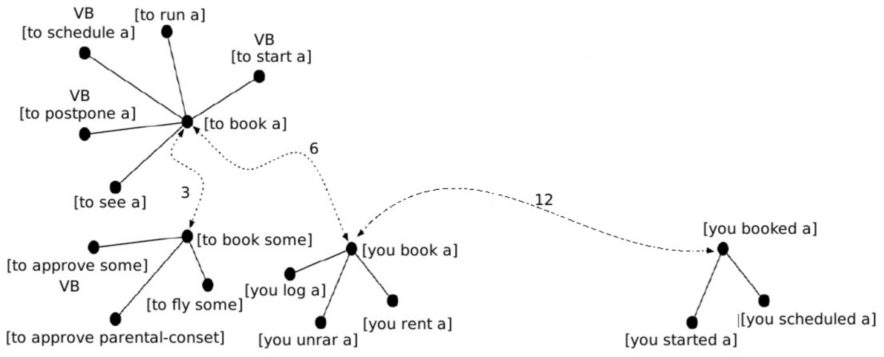


Figure 2.3: Distant vertices are less likely to get the same labels.

2.1.3 Learning the graph structure

Zhu et al. [2003] framed their graph propagation problem based on a given edge-weight matrix that represents the graph and later suggested the edge-weight matrix can be learned by minimizing the average label entropy. They assume the weights to be of the form

$$w_{ij} = exp(-\sum_{d=1}^{m} \frac{(x_{id} - x_{jd})^2}{\sigma_d^2}) \tag{2.1}$$

where x_i and x_j are two data points represented as a vector of length d. Note that σ_d parameters determine the edge weights, thus determining the graph structure.

To classify the vertices, they introduce a function $f : V \rightarrow \mathbb{R}$. If labels are assumed to be 0 and 1, function f gets the value of the reference label for the labeled vertices and then the propagation step is done by matrix operations so that the value of f at each unlabeled data point is the weighted average of the value of f at its neighbors:

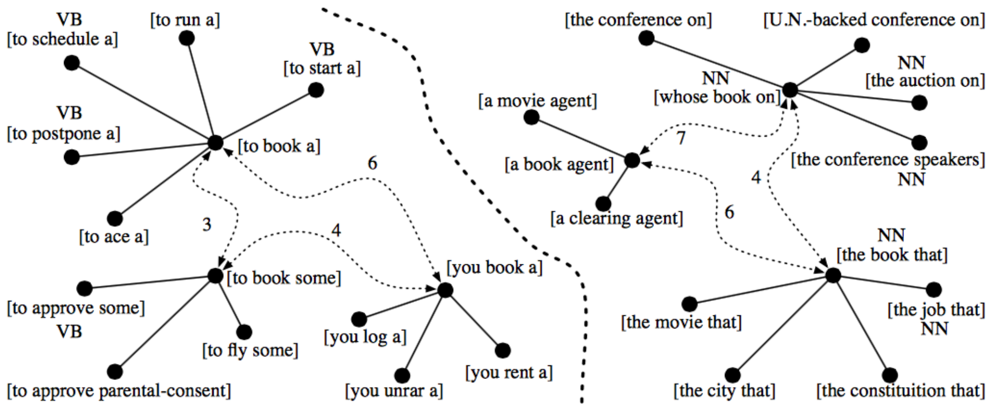


Figure 2.4: Verbs and Nouns constitute two different components in the graph Subramanya et al. [2010] constructed for the task of POS tagging.

$$\forall j \in N \; f(j) = \frac{1}{d_j} \sum_{i \in N(j)} w_{ij} f(i). \tag{2.2}$$

where N is the set of unlabeled data points (not to be confused with $N(j)$ that is the set of neighbors of j).

After the propagation is done, a vertex u gets classified as 1 if $f(u) > \frac{1}{2}$. To deal with differences in label distributions, they suggest class mass normalization: if q is the prior for the proportion of data points with label 1 to all data points, instead of testing if $f(u) > \frac{1}{2}$ to classify u as 1, we test

$$q \frac{f(u)}{\sum_i f(i)} > (1-q) \frac{1 - f(u)}{\sum_i (1 - f(i))}. \tag{2.3}$$

Notice that if $\frac{\sum_i f(i)}{\sum_i (1-f(i))} = \frac{q}{1-q}$ this test will be equivalent to $f(u) > \frac{1}{2}$.

To construct the graph they introduce the average label entropy as a function of f:

$$H(f) = \frac{1}{|N|} \sum_{i \in N} H_i(f(i)) \tag{2.4}$$

where $H_i(f(i)) = -f(i)log(f(i)) - (1 - f(i))log(1 - f(i))$, and find the σ_d parameters that minimizes it using gradient descent on $\frac{\partial H}{\partial \sigma_d}$. Note that minimizing $H(f)$ means finding a model that is as certain as possible about the labels it assigns to data points. We will re-visit this idea in Section 2.2.

2.1.4 Fixed vs. changing graph construction

Almost always in the literature, the graph structure and edge-weights are computed once and are fixed afterward. One exception to this rule is the work of [Dhillon et al., 2012] who take advantage of the labels in the training set to learn a metric over data points and construct the graph based on it. Algorithms 2.1 and 2.2 detail their procedure. Propagating labels through the current graph, they label the unlabeled data-points and expand their labeled data with confidently-labeled ones. They repeat this metric-learning, graph-construction, and labeling process until no new data point is confidently labeled.

In Algorithm 2.1, they use Information Theoretic Metric Learning (ITML) [Davis et al., 2007] as their metric learning method. To find a good choice of matrix A in the Mahalanobis distance, where $d_A(x_i, x_j) = (x_i - x_j)^\top A(x_i - x_j)$, ITML chooses a matrix A that

1. encourages the distance between the data points with the same label to be smaller than an upper bound u, and the distance between data points with different labels to be more than a lower bound l where $u < l$; and
2. is close to a matrix A_0 that comes from prior knowledge about the data points.

Examples of A_0 are the identity matrix when squared Euclidean distance between data points is appropriate or the inverse of the sample covariance where the data is believed to have been generated by Gaussian distribution.

In Algorithm 2.2, X is the set of data points, S is the set of labeled ones, Y is the corresponding set of labels, while $\hat{S}$ and $\hat{Y}$ are the set of expanding labeled data points and their labels which

Algorithm 2.1 Supervised Graph Construction (SGC) **Input:** instances X, training labels Y, training instance indicator S, neighborhood size k **Output:** Graph edge weight matrix, W

```
1: $A \leftarrow$ MetricLearner$(X, S, Y)$
2: $W \leftarrow$ ConstructKnnGraph$(X, A, k)$
3: return $W$
```

Algorithm 2.2 Iterative Graph Construction (IGC) **Input:** Instances X, training labels Y, training instance indicator S, label entropy threshold β, neighborhood size k **Output:** Graph edge weight matrix W

```
1: $\hat{Y} \leftarrow Y, \hat{S} \leftarrow S$
2: repeat
3:     $W \leftarrow$ SGC($X, \hat{Y}, k$)
4:     $\hat{Y}' \leftarrow$ GraphLabelInference($W, \hat{S}, \hat{Y}$)
5:     $U \leftarrow$ SelectLowEntInstances($\hat{Y}', \hat{S}, \beta$)
6:     $\hat{Y} \leftarrow \hat{Y} + U\hat{Y}'$
7:     $\hat{S} \leftarrow \hat{S} + U$
8: until convergence (i.e., $U_{ii} = 0, \forall i$)
9: return $W$
```

get initialized to S and Y respectively before the algorithm repeatedly constructs the graph (via algorithm 2.1), infers the labels for unlabeled data (via graph propagation where the cost function is the quadratic energy function formulated by Zhu et al. [2003] and discussed in Section 2.2), selects the previously unlabeled data points that are confidently labeled now, and adds them to $\hat{S}$ and their labels to $\hat{Y}$.

It is important to note that in algorithm 2.1, not only the edge-weights but also the structure of the graph changes. The change in metric results in a change in the edge-weights and since they keep only k nearest neighbors of a vertex, the structure of the graph can potentially change as well. As a result, this algorithm will be more time-consuming than others for big graphs.

2.1.5 Embedding another classifier

If the training data is big enough, we can train supervised classifiers and we might want to incorporate their output for unlabeled data points in label inference. Zhu et al. [2003] suggest using information from another (usually supervised) classifier in constructing the graph. Assuming that the other classifier assigns label $l_{u'}$ to the unlabeled vertex u, they propose adding a labeled vertex u' with label $l_{u'}$, connecting it to u with an edge of weight η, and discounting the weights of edges connecting other vertices to u by $1 - \eta$. The propagation is done on this augmented graph. This will ensure that unlabeled data leans toward the label the given (supervised) classifier has assigned to it unless there is reason to do otherwise.

In Section 2.3 we will see other ways of incorporating a classifier's output into label inference. However, they will not affect the structure or parameters (edge weights) of the graph.

2.2 Graph Propagation

The notion of graph propagation has been explored in various fields: in machine learning, of course, but also in data mining [Sun and Han, 2012]. Graph propagation in these related fields often focuses on recommendation networks or ranking tasks. Here, we will focus on machine learning methods using graph propagation as applied to NLP tasks focusing on cases where graph propagation is used to build models for structured prediction tasks.

In graph propagation, we associate any given vertex u with a label distribution X_u that represents how likely we think each label is for that vertex. Figure 2.5 shows how these label distributions might look like in an excerpt of the graph constructed by Subramanya et al. [2010].

The goal of graph-based SSL is to propagate existing knowledge about the labels, provided by the labeled data and potentially some prior knowledge, through the graph. This is usually done by optimizing an objective function over the above-mentioned label distributions. The objective function usually consists of 3 different types of constraints:

1. for any labeled vertex u, the associated label distribution X_u should be close to the reference distribution $\hat{X}_u$ (obtained from labeled data);

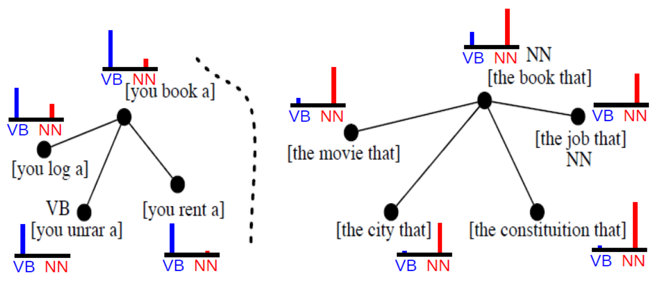


Figure 2.5: Every vertex of the graph has a label distribution.

2. adjacent vertices u and k should have similar label distributions X_u and X_k;
3. the label distributions of all vertices should comply with the prior knowledge if such knowledge exists. The prior knowledge is usually incorporated by a regularizer term in the objective function.

2.2.1 General Formulation

Figure 2.6 shows these constraints in a schematic way. X_i's are label distributions, $\hat{X}_i$'s are reference label distributions, and the shaded square between X_i's and $\hat{X}_i$'s shows their dependency; the white solid-edge squares between X_i's and X_j's show the dependency of label distributions for adjacent vertices; and finally, the dashed-edge white squares adjacent to X_i's show the dependency of label distributions to prior knowledge.

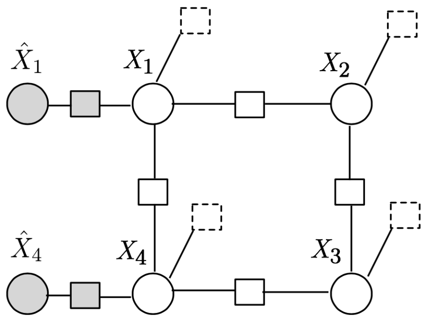


Figure 2.6: A schematic of constraints over label distributions [Das and Smith, 2012]

A general form objective function corresponding to these constraints looks like

$$\sum_{u \in L} f^{(1)}_{dist}(X_u, \hat{X}_u) + \sum_{u,k \in E} f^{(2)}_{dist}(X_u, X_k) + \sum_{u \in N} f_{prior}(X_u) \tag{2.5}$$

where L is the set of labeled vertices, N is the set of unlabeled vertices, E is the set of edges with non-zero weight, $f^{(1)}_{dist}$ and $f^{(2)}_{dist}$ are some distance functions, and finally, f_{prior} is the function representing our prior knowledge.

$f^{(1)}_{dist}$ and $f^{(2)}_{dist}$ are often chosen from usual distance functions such as Euclidean distance, K-L divergence, or the symmetrized generalization of K-L divergence– J-S divergence [Lin, 1991]; whereas f_{prior} can reflect any intuition about the data although it often has to do with entropy or distance from the uniform distribution. When an entropy-related term appears in the cost function, for example in the work of Das and Smith [2012], the idea is to prefer more certain models to uncertain ones; similar to the intuition of Zhu et al. [2003] for graph construction discussed in Section 2.1. The intuition of putting distance from the uniform distribution in the cost function, on the other hand, is to encourage uncertainty so that the model does not pick a label over the others unless there is strong evidence in data to encourage that decision.

Subramanya et al. [2010] use Euclidean distance as distance functions and Uniform distribution as the prior knowledge:

$$C(X) = \sum_{u \in L} ||X_u - \hat{X}_u||_2^2 + \sum_{u \in V} \Big(\sum_{k \in N(u)} \mu w_{u,k} ||X_u - X_k||_2^2 + \nu ||X_u - U||_2^2 \Big). \tag{2.6}$$

while Subramanya and Bilmes [2009] use K-L divergence distance functions along with entropy as the prior knowledge:

$$C(X) = \sum_{u \in L} D_{KL}(\hat{X}_u||X_u) + \mu \sum_{u \in V} \Big(\sum_{k \in N(u)} w_{u,k} D_{KL}(X_u||X_k) + \nu \sum_{u \in N}^{m} H(X_u) \Big). \tag{2.7}$$

The cost function Zhu et al. [2003] chose is also interesting in that it includes neither $f^{(1)}_{dist}$ nor f_{prior}. $f^{(1)}_{dist}$ is unnecessary in their case since at all times $\forall u \in L : \; X_u = \hat{X}_u$; and f_{prior} is redundant since they construct their graph based on their prior knowledge (average label entropy minimization). Thus, they only use the energy function

$$E(f) = \frac{1}{2} \sum_{(u,k) \in E} w_{u,k} (f(u) - f(k))^2. \tag{2.8}$$

which is equivalent to choosing Euclidean distance as $f^{(2)}_{dist}$.

2.2.2 Efficient optimization

The size of the graph makes the optimization of the objective function time-consuming. That is why there are usually iterative updates (most of the time based on some kind of gradient descent) that re-estimate distributions in each iteration based on the previous distributions. These updates sometimes come with guarantees and proofs of convergence to local optimas [He et al., 2013] and sometimes are only proven useful empirically [Subramanya et al., 2010].

For example Subramanya et al. [2010] obtained the derivative of their cost function $C(X)$ (in 2.6) with respect to each $X_i(y)$, and set this derivative to zero ($\frac{\partial C}{\partial X_i(y)} = 0$) to derive their iterative update rule. Details of this process are as following. Note that δ is a function that converts true/false to 1/0 values.

$$
\begin{aligned}
C(X) &= \sum_{j \in L} ||X_j - \hat{X}_j||_2^2 + \sum_{i \in V} \Big(\sum_{k \in N(i)} \mu w_{i,k} ||X_i - X_k||_2^2 + \nu ||X_i - \frac{1}{Y}||_2^2 \Big) \\
&= \sum_{i \in V} \Big((\delta(i \in L)) ||X_i - \hat{X}_i||_2^2 + \sum_{k \in N(i)} \mu w_{i,k} ||X_i - X_k||_2^2 + \nu ||X_i - \frac{1}{Y}||_2^2 \Big) \\
&= \sum_{i \in V} \sum_{y=1}^{Y} \Big((\delta(i \in L))(X_i(y) - \hat{X}_i(y))^2 + \sum_{k \in N(i)} \mu w_{i,k} (X_i(y) - X_k(y))^2 + \nu (X_i(y) - \frac{1}{Y})^2 \Big).
\end{aligned}
$$

$$
\frac{\partial C}{\partial X_i(y)} = 2(\delta(i \in L))(X_i(y) - \hat{X}_i(y)) + 2 \sum_{k \in N(i)} \mu w_{i,k} (X_i(y) - X_k(y)) + 2\nu (X_i(y) - \frac{1}{Y}).
$$

$$
\begin{gathered}
\frac{\partial C}{\partial X_i(y)} = 0 \leftrightarrow \frac{1}{2} \frac{\partial C}{\partial X_i(y)} = 0 \leftrightarrow \\
\Big(\delta(i \in L) + \Big(\sum_{k \in N(i)} \mu w_{i,k} \Big) + \nu \Big) X_i(y) = \delta(i \in L) \hat{X}_i(y) + \Big(\sum_{k \in N(i)} \mu w_{i,k} X_k(y) \Big) + \nu \frac{1}{Y}.
\end{gathered}
$$

The last equality gives us the exact update rule they use in each iteration:

$$
\begin{aligned}
X_i^{(m)}(y) &= \frac{\gamma_i(y)}{k_i} \\
\gamma_i(y) &= \hat{X}_i(y) \delta(i \in L) + \Big(\mu \sum_{k \in N(i)} w_{i,k} X_k^{m-1}(y) \Big) + \nu \frac{1}{Y}, \\
k_i &= \delta(i \in L) + \nu + \mu \sum_{k \in N(i)} w_{i,k}
\end{aligned}
\tag{2.9}
$$

where $X_i^{(m)}$ and $X_i^{(m-1)}$ are the label distributions of vertex i in iterations m and $m-1$, Y is the number of labels, and δ is 1 if and only if i is a labeled vertex.

As another approach to tame the complexity of graph propagation, many authors have mentioned parallelization [Das and Smith, 2012] [Talukdar and Pereira, 2010] [Subramanya and Bilmes, 2009].

2.3 Training CRF with Graphs

Graph-based semi-supervised methods can be categorized into transductive and inductive. In transductive settings, the output is a graph with stabilized label distributions for all vertices according to the objective function. This graph can not be used for label prediction for unseen data. In case we need to predict the labels for more data-points absent from the graph, we need to construct a new graph that includes the new data points and repeat the graph-propagation.

In inductive settings, the output is a trained model that can predict labels of unseen data. In NLP applications of inductive graph-based SSL, chain conditional random fields (CRF's) [Lafferty et al., 2001b] are popular, probably due to the fact that NLP works with sequences of words (sentences) and CRF's are suitable for modeling sequence structures.

In this section, we briefly describe training a supervised chain CRF, explain two graph-based semi-supervised CRF training methods, compare them, and mention a few alternative semi-supervised CRF training methods.

2.3.1 Training a supervised chain CRF

CRF's are the discriminative counterpart of Markov Random Fields and in general, can be defined on any factor graph. Chain CRF's are defined on factor graphs where the latent variables form a chain and are especially of interest in NLP because firstly, their parameters can be estimated in linear time via the forward-backward algorithm, and secondly, they are suitable for sequence tagging tasks in NLP such as POS tagging.

Figure 2.7 shows examples of chain CRF's. As we can see, the labels are assumed to be independent of all other labels except for the previous and the next one but can depend on all the observed variables (features) in the sequence.

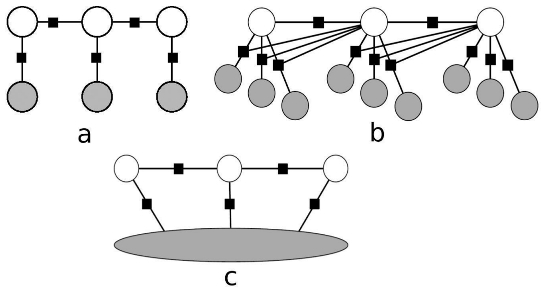


Figure 2.7: Examples of Chain CRF graphical model. [Sutton and McCallum, 2011] a. An HMM-like CRF; the current label only depends on the current word, the previous label, and the next label. b. A chain CRF where the label also depends on the features from the previous word. c. The general form of a chain CRF where the current label depends on the neighbor labels and features from the whole sequence.

The usual formulation of a chain CRF is

$$p(y_i|x_i;\Lambda) \propto \Big(\sum_{j=1}^{N_i}\sum_{k=1}^{K}\lambda_k f_k(y_i^{j-1}, y_i^{(j)}, x_i, j)\Big). \tag{2.10}$$

where x_i is the i^{th} observed sequence in the data-set (in NLP usually the i^{th} sentence in the corpus), y_i is the label sequence for x_i, Λ is the set of λ_k parameters (the weights for features f_k), K is the total number of features, and N_i is the length of x_i. It is clear from 2.10 and 2.7 that f_k's can be any feature that depends on the entire sentence, the current position, and the current and previous labels.

The model parameters, λ_k's, are obtained by optimizing the conditional log-likelihood of the labeled data normally regularized by an L2-norm term:

$$\Lambda^* = argmin_{\Lambda \in R^K} \Big[-\sum_{i=1}^{l} logp(y_i|x_i;\Lambda) + \lambda||\Lambda||^2 \Big]. \tag{2.11}$$

This cost function and its gradient can be computed efficiently using the forward-backward algorithm and due to the concavity of the function, following the gradients finds us the global optimum [He et al., 2013]. For a thorough introduction to CRF please see [Sutton and McCallum, 2011].

2.3.2 Iterative CRF training

Figure 2.8 shows the iterative approach proposed by Subramanya et al. [2010] for re-training a CRF. Having a partially-labeled corpus where the size of the labeled subset is much smaller than the size of the unlabeled subset, they train a CRF in a supervised fashion based on the labeled data (crf-train, line 1); and use this model to assign label probability distributions to each word in the entire (labeled + unlabeled) corpus (posterior decode, line 4). As a result, each n-gram token in the corpus has a label distribution (the posteriors). For each n-gram type u (a vertex in the graph), they find all instances (n-gram tokens) of u and average over the label distributions of these instances to get a label distribution for u (token to type, line 5). Next, they do graph-propagation to stabilize the label distributions for all vertices as discussed in Section 2.2 (graph propagate, line 6). Finally, they linearly interpolate the token label distributions according to the trained CRF and according to the corresponding vertex in the graph, and get the best label for all words in the entire corpus by Viterbi-decoding (viterbi-decode, line 7). This means they have a best label for every token now, which means their labeled corpus has grown to include the entire corpus. They re-train the CRF on this expanded train set (crf-train, line 8); and iterate until convergence. Note that the steps of lines 1, 4, and 8 work on the corpus whereas graph propagation in line 6 works on the graph. So, the step in line 5 takes us from the corpus to the graph, and the step in line 7, takes us back from the graph to the corpus.

2.3.3 Training with posterior regularization

Instead of an iterative alternation between supervised training of a model (such as CRF) and graph propagation as Subramanya et al. [2010] did, He et al. [2013] propose training the model by optimizing an objective function that not only includes a likelihood term over the labeled data but also includes a second term that represents prior knowledge in terms of posterior regularization. We are interested in their work because as we will see the prior knowledge can be provided by a graph.

```
1. $\Lambda^s$ =crf-train$(D_l, \Lambda^0)$
2. Set $\Lambda_0^{(t)} = \Lambda^{(s)}$
3. while not converged do
4.     $\{p\}$ =posterior_decode$(D_u, \Lambda_{old})$
5.     $\{q\}$ =token to type$(\{p\})$
6.     $\{q'\}$ =graph propagate$(\{q\})$
7.     $D_u^{(1)}$ =viterbi_decode$(\{q'\}, \Lambda_{old})$
8.     $\Delta_{n+1}^{(t)}$ = crf-train$(D_l \cup D_u^{(1)}, \Delta_n^{(t)})$
9. return $t$
```

Figure 2.8: Iterative training of CRF proposed by [Subramanya et al., 2010]

Concretely, they show adding any convex differentiable function of the model marginals (even a nonlinear one like graph Laplacian) into the objective function can be handled efficiently in the optimization step. For computational efficiency (based on previous work on posterior regularization [Ganchev et al., 2010]) they suggest using an auxiliary distribution instead of working directly with the model marginals and enforce similarity between this auxiliary distribution and the model by adding a KL-Divergence term to the objective function.

So, if p_θ represents the model, $p_\theta(u, y)$ is the probability of label y for vertex v according to p_θ. Since the KL-Divergence term is supposed to keep p_θ and X close, instead of defining the marginals of P_θ, we can define marginals of X :

$$m_{t,k,j}^i = \sum_y \delta(y_t = k, y_{t-1} = j) X(y|x^i). \tag{2.12}$$

Simply put, $m_{t,k,j}^i$ is the probability of words $t-1$ and t getting labels j and k in sentence x^i.

Although m (vector of $m_{t,k,j}^i$'s) consists of marginals of auxiliary distributions X, since X is going to be close to p_θ, m is considered to be the vector of marginals of the model. Posterior regularization is done by adding a function of these marginals into the objective function. Therefore, overall, the objective function looks like

$$J(\theta, X) = L(\theta) - R(\theta, X) \tag{2.13}$$

where

$$R(\theta, X) = D_{KL}(X||p_\theta) + \lambda h(m) \tag{2.14}$$

and $L(\theta)$ is a conditional likelihood objective such as

$$L(\theta) = \sum_{i=1}^{l} logp_\theta(y^i|x^i) - \frac{||\theta||_2^2}{2\sigma^2}. \tag{2.15}$$

In the graph constructed for POS tagging, for example, we can use the Laplacian of the graph $L = D - W$ (where D is the diagonal matrix with $d_{aa} = \sum_{j \in V} w_a j$) to define $h(m)$. L is such that $\forall v \in R^{|V|}$ $v^\top L v = \frac{1}{2} \sum_{a \in V} \sum_{b \in V} w_{a,b}(v_a - v_b)^2$. So, if for a given label l, we define vector v_l of length $|V|$ such that $v_l[u]$ estimates $X_u[l]$, we can define $h(m) = 2 \sum_{l \in Y} v_l^\top L v_l$ and $h(m)$ would estimate $\sum_{i \in V} \sum_{j \in V} w_{i,j} ||X_i - X_j||_2^2$. Here is how we can define such v_l's:

$$\forall u \in V \;\; v_l[u] = \frac{\sum\limits_{\substack{i=1 \\ B(i,t)=u}}^{n} \sum\limits_{t=1}^{T} \sum\limits_{y_{t-1}}^{Y} M^i_{t,l,y_{t-1}}}{\sum\limits_{i=1}^{n} \sum\limits_{t=1}^{T} \delta(B(i,t) = u)}. \tag{2.16}$$

where n is the number of tokens in the corpus and $B(i,t)$ maps an n-gram token to its n-gram type (a vertex in the graph).

Exponentiated Gradient Descent

He et al. [2013] call any function of the marginals a p-factor and show that the X^* minimizing $R(\theta, X)$ is decomposable to p-factors. Then, they argue that since exponentiated gradient descent preserves this property by its multiplicative update rule, and since this property leads to lower complexity for gradient descent (making it $O(F)$ where F is the number of p-factors), using exponentiated gradient descent is preferable to the conventional gradient descent.

We can show X^* decomposes to a product of p-factors by getting the derivative of R with respect to X^i_y and set X^* to the distribution that sets this derivative to zero. Since

$$R(\theta, X) = D_{KL}(X||p_\theta) + \lambda h(m) = \sum_{i,y} X^i_y log \frac{X^i_y}{p_\theta(y|x^i)} + \lambda h(m),$$

we have

$$\frac{\partial R}{\partial X^i_y} = \log X^i_y + 1 - \log p_\theta(y|x^i) + \lambda \sum_{t=1}^{T} \frac{\partial h(m)}{\partial m}^\top \frac{\partial m}{\partial X^i_y}. \tag{2.17}$$

Since $\frac{\partial m}{\partial X^i_y}$ is a 0-1 vector indicating which marginals of m apply to X^i_y, we have

$$\frac{\partial R}{\partial X^i_y} = \log X^i_y + 1 - \log p_\theta(y|x^i) + \lambda \sum_{t=1}^{T} \frac{\partial h(m)}{\partial m^i_{t,y_t,y_{t-1}}}. \tag{2.18}$$

Now, setting the derivative to zero, gives us a solution for X^* that decomposes to products of p-factors (Z is a normalization factor):

$$\frac{\partial R}{\partial X^i_y} = 0 \leftrightarrow \log X^i_y = -1 + \log p_\theta(y|x^i) - \lambda \sum_{t=1}^{T} \frac{\partial h(m)}{\partial m^i_{t,y_t,y_{t-1}}} \leftrightarrow X^i_y = \frac{p_\theta(y|x^i) exp\big[-\lambda \sum_{t=1}^{T} \frac{\partial h(m)}{\partial m^i_{t,y_t,y_{t-1}}}\big]}{Z_X(x^i)}. \tag{2.19}$$

The fact that multiplicative update rule in exponentiated gradient descent, $X^i_y \leftarrow X^i_y exp[-\eta \frac{\partial R}{\partial X^i_y}]$, preserves the decomposition property is clear when we consider the value of the new X^i_y which is

$$(X^i_y)^{1-\eta} p_\theta(y|x^i)^\eta exp[-\eta\lambda \sum_{t=1}^{T} \frac{\partial h(m)}{\partial m^i_{t,y_t,y_{t-1}}}] \tag{2.20}$$

up to a normalization constant. Since the old value of X^i_y and $p_\theta(y|x^i)$ both decompose to products of p-factors and the exponential term is a product of p-factors itself, the updated X^i_y still has the decomposition property.

Finally, since X^i_y decomposes to products of p-factors at all steps, $\frac{\partial R}{\partial X^i_y}$ can be re-formulated as sum of p-factors and He et al. [2013] use this property to do the gradient descent efficiently. The decomposition property of X^i_y means that we can re-write it as follows:

$$X^i_y = \frac{1}{Z_X(x^i)} exp[\sum_{t=1}^{T} r_{i,t}(y_t, y_{(t-1)})]. \tag{2.21}$$

Since

$$\frac{\partial R}{\partial X^i_y} = \log X^i_y + 1 - \log p_\theta(y|x^i) + \lambda \sum_{t=1}^{T} \frac{\partial h(m)}{\partial m^i_{t,y_t,y_{t-1}}} \tag{2.22}$$

and

$$p_\theta(y|x^i) = \frac{exp[\sum_{t=1}^{T} \theta^\top f(y_t, y_{t-1}, x^i)]}{Z_p(x^i)}, \tag{2.23}$$

we can write

$$\frac{\partial R}{\partial X^i_y} = C + \sum_{t=1}^{T} [r_{i,t}(y_t, y_{t-1}) - \theta^\top f(y_t, y_{t-1}, x^i) + \lambda \frac{\partial h(m)}{\partial m^i_{t,y_t,y_{t-1}}}], \tag{2.24}$$

where $C = 1 - \log Z_X(x^i) + \log Z_p(x^i)$ is a constant with respect to y. This means that r-factors can be updated as follows:

$$r_{i,t}(y_t, y_{t-1}) \leftarrow (1-\eta) r_{i,t}(y_t, y_{t-1}) + \eta \theta^\top f(y_t, y_{t-1}, x^i) - \eta\lambda \frac{\partial h(m)}{\partial m^i_{t,y_t,y_{t-1}}}. \tag{2.25}$$

Since the number of r-factors is equal to the number of marginals (the dimension of $m = F$), the complexity of overall update is $O(F)$. Note that starting with $X^i_y = p_\theta(y|x^i)$ is equivalent to starting with $r_{i,t}(y_t, y_{t-1}) = \theta^\top f(y_t, y_{t-1}, x^i)$. The only remaining issue is to ensure $\frac{\partial h(m)}{\partial m}$ is computed efficiently. He et al. [2013] argue that based on [Griewank et al., 1989], if computing h requires c operations, its gradient vector can be computed in $O(c)$; thus, as long as h itself can be computed efficiently, $\frac{\partial h(m)}{\partial m}$ is efficiently computable. It is worth mentioning that in case of graph-based posterior regularization computing h is $O(|Y||V|^2)$ which can be too time consuming for big graphs.

2.3.4 Comparing the two CRF training methods

While the iterative algorithm Subramanya et al. [2010] proposed for re-training the CRF doesn't enjoy a theoretical local optimality proof similar to what He et al. [2013] provide for their posterior regularization method, based on their experiments and results, it seems like the iterative algorithm of Subramanya et al. [2010] is more scalable. Also, looking more closely at their settings and the results they provide, the improvements He et al. [2013] report are not as impressive as the improvements Subramanya et al. [2010] report.

Subramanya et al. [2010] experiment in a domain adaptation setting. Their labeled data contains about 38,000 labeled WSJ sentences (about 900,000 tokens) from Wall Street Journal (WSJ) of the Penn Treebank [Marcus et al., 1994] while their unlabeled data consist of 10 million questions from QuestionBank [Judge et al., 2006] for the “questions" target domain and about one thousand sentences from PennBioTreeBank [Liberman and Mandel, 2008] for the “biomedical" target domain. In the QuestionBank experiment, they randomly selected 100,000 unlabeled sentences from web search logs and added the trigram types of only those sentences to the trigram types of the labeled sentences to form the vertices in the graph. However, they used the whole set to estimate the point-wise mutual information between features and vertices. Table 2.2 shows their improvements in the semi-supervised setting. They obtain an absolute 3% improvement for the “questions" target domain.

	Questions	Bio
Supervised CRF	83.8	86.2
Self-trained CRF	84.0	87.1
Semi-supervised CRF	86.8	87.6

Table 2.2: Performances reported by Subramanya et al. [2010] in the domain adaptation setting.

On the other hand, there is evidence that He et al. [2013] experimented on smaller graphs and this could be because their method is not as scalable as that of Subramanya et al. [2010]. He et al. [2013] do not mention how many unlabeled sentences they have but they do mention small train-sets, 100 labeled sentences, and graphs ranging in size from about 25,000 vertices (for Slovene) to about 612,000 (for English). Since they use trigram types as their vertices, we can say they have roughly the same number of tokens which (considering 10 to be the length of an average sentence) means they have about 62,000 sentences in comparison with about 138,000 sentences in the QuestionBank experiment of Subramanya et al. [2010] But this is not the only reason we believe He et al. [2013] used smaller graphs than the one Subramanya et al. [2010] used in the “questions" target domain. Other reasons include the fact that He et al. [2013] only used the twelve universal POS tag set of [Das and Smith, 2012]; that they compute their edge weights based on a subset of less-frequent features that Subramanya et al. [2010] used (see table 2.1) because too-frequent features match in unrelated trigrams; and finally the fact that in spite of using a relatively large k(=60) they only include edges between pair of vertices that are mutual neighbors ($(u, v) \in E \leftrightarrow u \in N_k(v) \wedge v \in N_k(u)$).

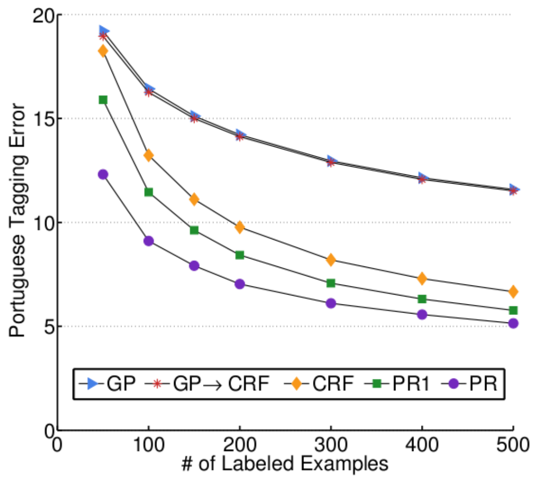


Figure 2.9: Effect of number of labeled sentences on accuracy for one language (reported by He et al. [2013]) in different methods: Graph Propagation, CRF trained based on output of Graph Propagation, CRF, First Iteration of Posterior Regularization, and Posterior Regularization .

Furthermore, He et al. [2013] train and test their model on the same domain. A supervised CRF can get more than 95% accuracy with a decent training set [Stratos and Collins, 2005]. But to train their model and the baselines, He et al. [2013] use very small training sets and report error rates worse than the state of the art. Also, it is clear from Figure 2.9 that the improvement over supervised CRF decreases when the number of labeled sentences increases. Since Subramanya et al. [2010] use plenty of labeled sentences no such concern exists there.

2.3.5 Alternative semi-supervised CRF models

Graph-based semi-supervised learning is not the only way to inject information from unlabeled data into a CRF. Here, we briefly mention a few alternatives.

Self-training [Scudder, 1965] is the simplest form of SSL applied for CRF's. The idea is to train the model on the labeled data, have the trained model assign labels to unlabeled data points, and pretend these labels (or at least the ones that the model is confident about) are true labels, expand the labeled data with these newly labeled data points, and re-train the model on the expanded labeled data. Subramanya et al. [2010] have reported better accuracies for self-trained CRF in comparison with supervised CRF; although their graph-based semi-supervised CRF obtained even better results. Jiao et al. [2006] on the other hand, reported that self-training led to no improvement over supervised CRF on the task of gene-mention detection.

Yang et al. [2012] adapted the idea of co-training [Blum and Mitchell, 1998] to train a CRF. They split the features into two subsets that are weakly dependent and trained two supervised CRF's on the same training data, each based on one of these feature subsets. Then, they had the trained models assign labels to unlabeled data, expanded the training set for each of the models with data points confidently labeled by the other model, re-trained models on these expanded train sets, and repeated the process for a few iterations.

Jiao et al. [2006] have applied entropy minimization [Grandvalet and Bengio, 2004], to a CRF model and improved upon a gene-name-tagger. To minimize the entropy of the output labels the model assigns to the unlabeled data, they added an entropy regularizer to CRF's cost function. Although the update rules they used for minimizing the cost function were not efficient, Mann and McCallum [2007] presented efficient update rules to train such a semi-supervised CRF.

Finally, Suzuki and Isozaki [2008] trained a semi-supervised CRF by combining CRF's (that could model labeled data) with generative models (that could model unlabeled data).

For further information on semi-supervised CRF's (or SSL in general) please refer to [Zhu, 2005].

2.4 NLP Applications

In addition to POS tagging that we have already mentioned, graph-based semi-supervised learning has been successfully applied to various applications including handwritten digit recognition [Zhu et al., 2003], phone classification [Subramanya and Bilmes, 2009], statistical machine translation [Alexandrescu and Kirchhoff, 2009; Liu et al., 2012; Saluja et al., 2014], bilingual lexicon extraction [Tamura et al., 2012], projecting POS tags from one language to another [Das and Petrov, 2011], class instance acquisition [Talukdar et al., 2008], and sentiment analysis [Dhillon et al., 2012]. In this section, we briefly explain two of these applications to show some of the similarities and differences of different NLP applications of graph-based SSL.

2.4.1 Bilingual POS tagging

Das and Petrov [2011] constructed a graph on a bilingual corpus and projected the twelve universal part-of-speech tags of Petrov et al. [2012] from one language to another via this graph to find a set of possible POS tags that could be used as features in an unsupervised model. They used English as the source language and experimented with eight European languages at the receiving end.

Their graph consists of two sets of vertices: one set for the source language, and one for the foreign language. The source-side vertices are unigrams as opposed to the 3-grams for the vertices of the destination language. The source-side vertices are not connected to each other, whereas the destination side vertices are connected following the graph construction procedure of Subramanya et al. [2010] explained in Section 2.1, with exactly the same set of features but the maximum number of neighbors in the graph, K is 5 instead of 10. They connect source-side vertices to destination-side vertices based on the alignment information provided by a sentence-aligned parallel corpus. They only connect source vertex u to a foreign vertex v if the unigram in u and the middle word of v appear in the high-confidence intersected alignments of the corpus. They add weights to the edges in proportion to the incoming degrees. Note that they only have edges going from the source-side vertices to the foreign vertices.

The graph propagation happens in two stages. In the first stage, they propagate the labels from source-side vertices to foreign vertices in one step that results in a tag distribution that encodes the proportion of times the middle word of a foreign vertex is confidently aligned to a source word carrying any of the twelve universal tags. Since only high-confidence alignments are used in graph construction, many of the foreign vertices do not receive any tag distribution in this stage. In the second stage, the label distributions in the destination side get propagated by iteratively minimizing the following cost function:

$$\begin{gathered}\sum_{u_i \in V_f \setminus V_f^l} \sum_{k \in N(i)} \mu w_{i,k} ||X_i - X_k||_2^2 + \lambda ||X_i - \frac{1}{Y}||_2^2 \\ s.t. \quad X_i = r_i \quad \forall u_i \in V_f^l\end{gathered} \tag{2.26}$$

where V_f is the set of foreign vertices and V_f^l is a subset of them that received label distributions, r_i's, in the first stage of propagation. Note that r_i's don't change during this stage.

Finally, for each foreign word, they compute the probability of each label by marginalizing the previous and next words out and pick the labels that are more probable than a threshold as possible labels for that word. These possible labels are fed into a feature-based unsupervised model. They achieved a statistically significant improvement over the state-of-the-art monolingual unsupervised model of Berg-Kirkpatrick et al. [2010]: the average accuracy increased from 73.0% to 83.4% where the accuracy of the fully supervised POS tagging is 96.6%.

2.4.2 Frame-semantic identification

Das and Smith applied graph-based semi-supervised learning to restrict the candidate semantic frames for given lexical predicates that led to a 15% absolute improvement in frame identification over the state-of-the-art.

They built their model on top of the-state-of-the-art frame-semantic parsers [Johansson and Nugues, 2007][Das et al., 2010] and use a supervised feature-based probabilistic model for frame identification. However, instead of allowing for all possible frames for predicates absent from the training set, they perform graph propagation to infer frame distributions for predicates in un-annotated data (with marked predicates but no annotation for frames) and allow only the M most probable frames where M is a hyper-parameter chosen via cross-validation.

The vertices in the graph are given predicates and the edge-weights are computed based on a linear combination of two similarity functions:

1. A thesaurus on the similarity of predicates (words or phrases) was constructed based on Lin's syntactic co-occurrence statistics [Lin, 1998] after parsing a corpus of 64 million words with a fast dependency parser [Lin, 1993] [Lin, 1994]. Only the top 20 most similar predicates

were connected to a given predicate (vertex) in graph construction and the edge weight was their similarity from the thesaurus. The intuition behind this similarity function is that lexical units evoking the same set of frames are expected to appear in similar syntactic contexts.

2. The other similarity function was based on the similarity of frame-distributions in FrameNet [Fillmore et al., 2003] and the training set. The Euclidean distances were computed, subtracted from the maximum distance in the data, and normalized to give a score between 0 and 1. The intuition behind linearly combining this similarity function with the previous one was that involving the annotated data can reduce the noise in the automatically constructed thesaurus.

Combining the two functions, they obtained a similarity measurement and used it to construct a K nearest neighbors (KNN) graph. They set the value of K to 10 in their experiments.

2.5 Summary

In this chapter, we presented a detailed overview of graph-based semi-supervised learning for NLP. We explained a few ways that the graph can be constructed and how the information can be propagated through the weighted edges. We also distinguished the transductive and inductive settings in this approach and focused on training CRF models as outputs of an inductive graph-based SSL approach. Finally, we outlined how the approach can be adapted to work for two NLP tasks, Bilingual POS tagging, and Frame-semantic identification, on top of the English POS tagging which we used as a running example throughout the chapter.

The next chapter introduces GraphNER, our adaptation of this idea for named entity recognition(NER). We show the efficacy of this approach on biomedical NER datasets.

Chapter 3

GraphNER

The rapidly growing amount of research papers in computational biology makes it difficult for researchers to keep up to date on new results. The motivation behind this work is to use natural language processing to automatically understand relevant concepts from a large amount of text data in published papers in computational biology. As a proof-of-concept, we focus on the gene mention detection task, which allows us to identify genes that are being discussed in papers, making it possible to search for concepts like genes rather than searching on words.

In this chapter, we introduce GraphNER, a semi-supervised machine learning model for named entity recognition (NER). In particular, we use GraphNER to identify gene mentions in natural language data such as biomedical papers. It combines training data where the gene mentions are identified by human experts and unlabeled data that contains many other relevant gene mentions. The labeled and unlabeled data are linked together using similarities between n-grams that occur in the two data sources (an n-gram is a contiguous sequence of n words in the text). GraphNER uses the information gleaned from this graph and combines it with a conditional random field (CRF) model for NER. We consider two different CRF-based NER systems on two different datasets combined with our graph model for semi-supervised learning for the task of gene mention detection. We show that GraphNER consistently improves over previous state-of-the-art thanks to its higher precision. This work was published in two parts [Sheikhshab et al., 2016, 2018b]. Even though GraphNER was surpassed by later methods [Jin et al., 2019; Lee et al., 2020; Lin et al., 2021], we do not describe them in this chapter. The next chapter will introduce our next effort to improve gene mention detection and mentions later advances as well. GraphNER is freely available at `http://www.bcgsc.ca/platform/bioinfo/software/graphner`.

3.1 Introduction

Detecting the named entities in a document is often the first step in many natural language processing (NLP) tasks, such as relation extraction and knowledge discovery. Named entities of interest in the biomedical domain include mentions of genes, mutations, proteins, and diseases in text.

Current approaches formulate named entity recognition (NER) as a sequence tagging problem, where the input is a sentence represented as a sequence of words $x_1, x_2, ..., x_l$, and the output is a sequence of corresponding tags $t_1, t_2, ..., t_l$. The tag t_i also marks if the term x_i represents the *beginning*, *inside*, or *outside* of a named entity type, the first two corresponding to the first, and internal terms of a mention, respectively, and the last one indicating that it is not part of a named entity mention. Hence, the tag-set consists of two tag types for any of the desired entity types (B and I), and a tag type for all other terms (O). For example, if the task is to detect genes, mutations, and diseases, the tag-set will be {B-Gene, I-Gene, B-Mutation, I-Mutation, B-Disease, I-Disease, O}.

The following sentence shows a sample input/output for a gene mention detection task:

```
The/O mutation/O of/O lymphocyte/B adaptor/I protein/I (/O
LNK/B or/O SH2B3/B )/O was/O detected/O in/O MPN/O
```

where B, I, and O are used for B-Gene, I-Gene, and O, since we are only considering gene mentions in this example. Based on the output tags, we can conclude there are three distinct gene mentions in this sentence: "lymphocyte adaptor protein", "LNK", and "SH2B3".

Conditional random field (CRF) [Lafferty et al., 2001a] is one of the most successful supervised methods for sequence tagging problems. Many popular NER systems, including some popular biomedical systems such as BANNER [Leaman et al., 2008], Gimli [Campos et al., 2013], and BANNER-ChemDNER [Munkhdalai et al., 2015], are based on CRF. However, CRF is a supervised method and can only make use of labeled data. Obtaining labeled data for biomedical NER tasks is considerably time-consuming and expensive given that highly trained human annotators are needed to produce such data.

Motivated by the success of Graph-based SSL in other sequence tagging tasks, we extended the algorithm of Subramanya et al. [2010] from part of speech tagging to named entity recognition and implemented GraphNER, a biomedical named entity recognition tool. GraphNER inputs the probabilistic output of CRF-based systems such as BANNER or BANNER-ChemDNER and improves their label assignments using a graph constructed over a given partially labeled corpus.

We compared our work to the initial best-performing methods in the BioCreative II gene mention (BC2GM) shared task, as well as more recent methods before GraphNER that report results on the corpus of that shared task. The best performing method in the BC2GM shared task was a semi-supervised method from IBM Watson Research Center [Ando, 2007] reporting an F-score of 87.21%. A more recent semi-supervised tool before GraphNER, BANNER-ChemDNER [Munkhdalai et al., 2015], takes advantage of abundant unlabeled data by using Brown clustering [Brown et al., 1992] and word-to-vector (word2vec) embeddings [Mikolov et al., 2013], leveraging these embeddings as features in its otherwise supervised machine learning core (CRF). Brown clustering and word2vec embeddings both capture the syntactic and semantic similarity of words. Brown clustering constructs a cluster hierarchy over the words by maximizing the mutual information of bi-grams, whereas word2vec embeddings are the hidden layer of a neural network, trained to predict each word by using the words in its context.

We also evaluated a strong neural network method for named entity recognition, based on a bi-directional long short-term memory with a CFR layer, LSTM-CRF [Lample et al., 2016], and benchmarked it on the BC2GM corpus. LSTM-CRF has been shown to outperform bi-directional LSTM and simplified LSTM (S-LSTM) [Lample et al., 2016]. It achieves comparable results to LSTM with a convolutional neural network layer (LSTM-CNN) [Chiu and Nichols, 2016] on the same benchmark corpus in the absence of lexicons.

3.2 Approach

We developed GraphNER, a graph-based SSL for named entity recognition, specifically for gene mention detection. Following the practice introduced by Subramanya et al. [2010], we first construct a graph where the vertices are 3-grams, and edge-weights encode corpus-level similarity of these 3-grams. We use this graph to push similar 3-grams towards taking similar labels. This is a semi-supervised method because the graph is constructed on labeled and unlabeled data. While this method is theoretically capable of using abundant unlabeled data, we chose a transductive approach: the only unlabeled data we use in graph construction is the test data. The rationale behind this decision is the time complexity of graph construction, as the scalability of graph construction for large datasets remains an open problem.

Algorithm 3.3 shows the train and test procedure of GraphNER. The training stage of the algorithm comprises an expansion of a base CRF model, where we scan the labeled data (D_l) and compute an average label distribution $X_{\text{ref}}(v)$ for any 3-grams v that appear in D_l (we call the set of such 3-grams V_l).

The testing procedure is more complex than the training stage. First, we run the CRF to extract the posteriors and transition probabilities. Then, we compute the average of these posteriors for each possible 3-gram v to form the initial belief about label distribution of v, which we will call $X(v)$. These distributions are then propagated on the graph to ensure similar 3-grams get similar distributions, as described in further detail in the Graph propagation section. Finally, we assign labels to all words w that appear in the context of (w_{-1}, w, w_1) in an input sentence S, using a mixture of graph and CRF results. That is, we combine the graph's belief, $P_s(S,i)$ (i indicating the

Algorithm 3.3 GraphNER

```
1: procedure TRAIN
2:     CRF_train($D_l$)
3:     $X_{\text{ref}}, V_l \leftarrow$ Set_ReferenceDistributions($D_l$)
4: procedure TEST
5:     $P_s, T_s \leftarrow$ CRF_Posteriors_And_Transitions($D_l \cup D_u$)
6:     X $\leftarrow$ Average($P_s$,$V$)
7:     X $\leftarrow$ Propagate($X, X_{\text{ref}}, \mu, \nu, \#iterations$)
8:     $P'_s \leftarrow$ Combine ($P_s, X, V, \alpha$)
9:     finalLabels $\leftarrow$ Viterbi($P'_s, T_s$)
```

labeled data:
drug/O response/O was/O significant/O in/O wilms/B tumor/I -/I 1/I positive/O patients/O ./O
we/O observed/O the/O following/O mutations/O in/O wilms/B tumor/I -/I 1/I ./O
we/O did/O not/O observe/O this/O mutation/O in/O the/O patient/O '/O s/O tumor/O -/O 1/O subclone/O ./O

unlabeled data:
wilm ' s tumor - 1 (wt1) gene was highly expressed .
we did not observe this mutation in the patient ' s tumor - 2 subclone .

Input: labeled and unlabeled data that includes the sentences shown above and a graph constructed over the whole partially labeled corpus. Part of the input graph is shown in (a).

Output: BIO labels for every word in all unlabeled sentences. We focus on "-" in the two unlabeled sentences shown above.

After training: Reference labels, the average label distributions based on the labeled data are associated with some vertices in the graph as shown in (b).

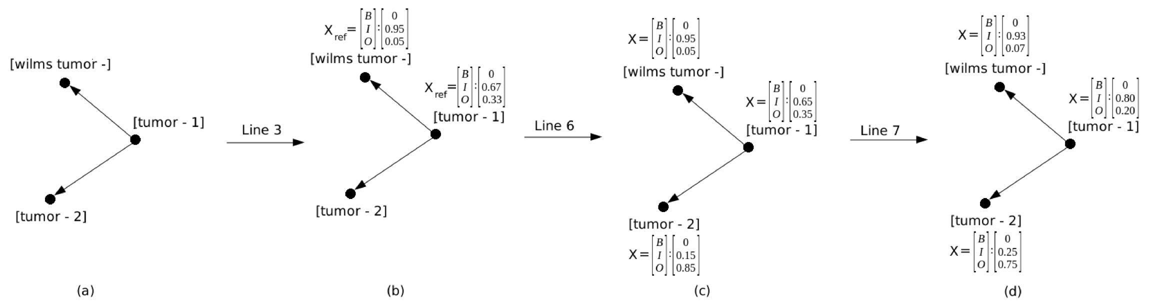
After line 5 in Algorithm 3.3: The following posterior probability distributions are extracted from CRF. For "-" in the first sentence we get $\underline{\text{(B,I,O)=(0,0.45,0.55)}}$ and for "-" in the second sentence we get $\underline{\text{(B,I,O)=(0,0.15,0.85)}}$. Label O is preferred for both instances of "-". In the second case, this preference is correct and much more enunciated. In the first case, however, O is not the right label. One reason that the CRF could make such a mistake is that it has seen the sequence <' s tumor - 1> before in the labeled data with annotation O for "-".

After line 6: The average of extracted posterior probabilities are put on the graph as illustrated in (c).

After line 7: As a result of graph propagation, the vertex [tumor - 1] has a label distribution that peaks at I (note the label distributions in (d)). The reason is that this vertex will have many neighbor vertices where I is preferred. Examples of such neighbors are [wilms tumor -] that is shown in the figure and [wilms tumour 1], [wilms ' tumor], and [wilms tumor ,] that are not shown due to lack of space.

After line 8: The posteriors extracted from CRF and the distributions on the graph after propagation are linearly combined with coefficients α and $1 - \alpha$ respectively. Let us pick the coefficient α to be 0.1 since smaller α values were consistently preferred in our cross validations. This will give us new distributions for "-" in the first and second sentences: $\underline{\text{(B,I,O)=(0,0.77,0.23)}}$ for the first "-" and $\underline{\text{(B,I,O)=(0,0.24,0.76)}}$ for the second "-".

After line 9: Labels of both instances of "-" are correct after the Viterbi algorithm chooses the most probable tag sequence for the unlabeled sentences.



[wilms tumor -]
[tumor - 1]
[tumor - 2]
(a)
Line 3
X_ref = [B I O] : [0 0.95 0.05]
[wilms tumor -]
X_ref = [B I O] : [0 0.67 0.33]
[tumor - 1]
[tumor - 2]
(b)
Line 6
X = [B I O] : [0 0.95 0.05]
[wilms tumor -]
X = [B I O] : [0 0.65 0.35]
[tumor - 1]
[tumor - 2]
X = [B I O] : [0 0.15 0.85]
(c)
Line 7
X = [B I O] : [0 0.93 0.07]
[wilms tumor -]
X = [B I O] : [0 0.80 0.20]
[tumor - 1]
[tumor - 2]
X = [B I O] : [0 0.25 0.75]
(d)

Figure 3.1: Illustration of Algorithm 3.3 with an example.

index that corresponds to a given word w in the sentence S), and the CRF's belief, $X(w_{-1}, w, w_1)$, about a label of w by computing $\alpha P_s(S,i)+(1-\alpha)X(w_{-1}, w, w_1)$, and get a final Viterbi decoding on the sequence of these values and transition probabilities of CRF. For an illustration of these procedures, see Figure 3.1.

Our algorithm differs from that of Subramanya et al. [2010] (detailed in Section 2.3.2) in that we have a transductive setting where we train and test once, whereas they have an inductive setting: they expand the labeled dataset by treating the output of Viterbi decoding (line 9) as correct and iterating over the train and test procedures, overwriting these labels until convergence or the 10th iteration. Note that we follow the same iterative graph propagation algorithm as them (elaborated in Section 2.2.2) and the hyper-parameter $\#iterations$ (line 7) refers to this iterative algorithm.

3.2.1 CRF Models used by GraphNER

We used two different CRF-based NER systems as the base model that is extended by the GraphNER model; that is in Algorithm 3.3, we used these tools in lieu of CRF. In order to extract posterior and transition probabilities, we have modified the source code of these tools. The modified version of these tools is included in our publicly available implementation.

BANNER

BANNER [Leaman et al., 2008] is a popular biomedical named entity recognition system that is frequently cited [Dai et al., 2015; Krallinger et al., 2015; Luo et al., 2016; Gonzalez et al., 2016; Hebbring et al., 2015], and is used for gene mention tagging [Li et al., 2015; Hakala et al., 2015; Leaman et al., 2015; Pyysalo et al., 2015; Li et al., 2015; Lee et al., 2014; Leaman et al., 2013]. The features used in our graph construction were extracted from BANNER.

BANNER-ChemDNER

BANNER-ChemDNER [Munkhdalai et al., 2015] is a tool built on BANNER and uses features extracted from large unlabeled datasets. As such, it is considered a semi-supervised version of BANNER.

3.2.2 Graph construction

The central idea in GraphNER is to have a graph that tells us what data points are similar so that we can assign similar labels to them. We followed a popular approach in graph-based semi-supervised learning for NLP applications (specifically, the ones that can be formulated as tagging problems), putting 3-grams as vertices, and representing them with a vector of feature values. That is, a vertex is represented as a vector of pointwise mutual information between the 3-gram associated with it and possible feature instances such as surrounding words. The edge weight between two vertices is the unnormalized cosine similarity (i.e. dot product) of their vector representations. We kept the graph sparse by keeping only k nearest neighbors for each vertex, which means the final graph is a directed one.

Different choices of feature sets change the vector representations, and consequently the edge weights and structure of the graph, leading to a different performance in GraphNER. We considered using all features extracted by BANNER (All-features), only lemmas of the words in a window of length 5 (Lexical-features), and features that have high mutual information with the tag assigned by BANNER (MI greater than some fixed threshold).

3.2.3 Datasets

BC2GM corpus

This dataset, introduced for the BioCreative II shared task in 2006, contains 15,000 training and 5,000 test sentences from published abstracts. Annotations are given by the starting and finishing character indices of genes in sentences. The space characters are ignored. Some sentences have alternative annotations presented in a separate file. This dataset is publicly available, and many studies have reported results on it, including the methods we compared against in this work [Campos et al., 2013; Ando, 2007; Rei et al., 2016].

AML corpus

This is a collection of 80 full-text articles related to acute myeloid leukemia (AML) clinical variant interpretation. The annotations are provided in the same format as the BC2GM corpus.

We divided the corpus into train and test sets by randomly selecting 22 full-text articles for the test set, and we placed the rest in the train set. The training set contains 10,504 sentences and the test set contains 3,952.

3.2.4 A note on time complexity

We can discuss the time complexity of GraphNER in three different phases: graph construction, model training, and model testing.

Time complexity of graph construction

Graph construction consists of a. constructing the feature vectors for vertices and b. computing the dot products of every pair and keeping only K nearest neighbors for each vertex.

In the first step, constructing feature vectors, we need to go through all 3-gram tokens in the corpus and try to extract all relevant features. The time complexity of this step is a linear function of the number of 3-gram tokens in the corpus (N). The constant in the linear function depends on not only the number of features (f) that we need to extract for each 3-gram token but also the difficulty of that feature extraction.

In the second step, we will get the dot product of all possible pairs of vertices (unique 3-grams). Therefore, the time complexity depends on the number of vertices (V) and the size of the feature vector(F). The feature vector can be large because there are as many elements in the feature vector as there are feature instances, which can be as many as Nf. The worst-case scenario of size Nf would happen if all the feature instances for any 3-gram token are unique. Computing the dot product between all pairs of vertices would have a time complexity of $O(V^2F)$. Keeping only K nearest neighbors adds another K or ($log(K)$ if we use a heap data structure) factor to the time complexity.

Overall, the time complexity of graph construction can be summarized as $O(Nf + V^2FK)$.

Added time complexity to train and test

As Algorithm 3.3 shows, the train and test contain training and testing CRF as well as other steps. The question is how much GraphNER adds to the time complexity of CRF.

In training, GraphNER needs to set the reference distributions (line 3 of Algorithm 3.3), which involves going through all 3-grams of the training set, keeping the number of occurrences and the number of any tag for all vertices (unique 3-grams), and finally loop over all vertices and divide the number of tags by the number of occurrences to get the distributions. The time complexity that this procedure adds is of $O(N_l + V_l)$ where N_l and V_l are the numbers of 3-gram tokens and unique 3-grams in the training set.

In testing, we do a similar averaging over all the posteriors (line 6 of Algorithm 3.3), adding a time complexity of $O(N + V)$ where N and V are the numbers of 3-gram tokens and unique 3-grams in both training and test sets. Then, graph propagation happens that consists of repeating over equation 2.9 for $V.\#Iterations$ times. Equation 2.9 itself is of $O(K)$, then the graph-propagation overall is of $O(VK\#Iterations)$. The next line (line 8 of Algorithm 3.3) involves going through all 3-gram tokens and doing a weighted sum, so the complexity of that is $O(N)$ and finally, the complexity of the last line, the Viterbi, is going to be also $O(N)$ because Viterbi has the complexity of $O(LQ^2)$ where L is the length of the sentence and Q is the number of tags which is only 3 in our case.

So, overall, the added time complexity of GraphNER is only $O(N_l + V_l)$ for training and $O(N + VK\#Iterations)$ for testing.

3.3 Results

We report the precision, recall, and F-Score obtained from the evaluation script of the Biocreative II gene mention task. The script compares detections with primary gene mentions and their alternatives and counts exact matches as true positives. Alternative annotations were present in the BC2GM corpus, but not in the AML corpus. The number of false negatives will be the number of primary gene mentions minus the number of true positives, and the number of false positives will be the number of detections minus the number of true positives.

As shown in Table 3.1, GraphNER improves both baselines on the BC2GM Corpus. A significance test with the SIGF tool [Padó, 2006b] (discussed in further detail in Section 3.3.1) revealed that these F-Score improvements were statistically significant.

Table 3.1: Results on the BC2GM corpus. Bold numbers indicate the best performance in each metric and * is used to show statistical significance. F-score is often used to measure the trade off between the precision and recall, and is the harmonic mean of these metrics.

Category	Method	Precision (%)	Recall (%)	F-Score (%)
Published in the literature	Ando (2007)	88.48	85.97	87.21
	Gimli (2013)	**90.22**	84.32	87.17
	BANNER-ChemDNER (2015)	88.02	86.08	87.04
	Rei et al. [2016]	-	-	**87.99**
Obtained from existing methods	LSTM-CRF	88.80	84.28	86.48
	Rei et al. [2016]	87.04	**88.72**	87.77
	BANNER	86.88	82.02	84.38
	BANNER-ChemDNER	87.51	85.49	86.49
GraphNER	CRF=BANNER	90.21	81.85	85.83*
	CRF=BANNER-ChemDNER	89.18	85.57	87.34*

Table 3.1 also shows that we were not able to exactly replicate the published BANNER-ChemDNER's performance on this task. While the authors have reported an F-score of 87.04%, we obtained a slightly lower performance, at 86.49%. Regardless, plugging BANNER-ChemDNER into GraphNER led to an F-score (87.34%) that is greater than the published F-score of BANNER-ChemDNER on BC2GM.

It is also worth mentioning that when applying the two neural-net-based methods [Lample et al., 2016; Rei et al., 2016] in the literature on BC2GM, we had to train them on a subset of the train set as they both need a dev-set. We divided the train set into a 12000-sentence train subset and a 3000-sentence dev-set. The fact that we did not have access to the exact train/dev sets that Rei et al. [2016] used explains the difference in our F-Scores.

The performance of GraphNER on the AML corpus is reported in Table 3.2. Performance of the baseline CRF-based supervised learning systems and GraphNER were substantially higher for the AML corpus relative to the BC2GM corpus. These performance differences were expected because there were multiple differences in the article curation and manual annotation procedures for the two corpora. The BC2GM corpus was curated broadly from articles in the field of biology, whereas the AML corpus was restricted to human clinical genetics articles. In the general field of biology, gene names may be used inconsistently with a variety of notation styles. Clinical genetics articles have a more standardized discourse about genes, and articles in this field preferentially use a gene nomen-

Table 3.2: Results on the AML corpus.

Category	Method	Precision (%)	Recall (%)	F-Score (%)
Baselines	BANNER	96.56	94.56	95.55
	BANNER-ChemDNER	97.29	96.00	96.64
GraphNER	CRF=BANNER	97.56*	94.46	95.98
	CRF=BANNER-ChemDNER	**97.68***	**96.08**	**96.87**

Table 3.3: Hyperparameters of GraphNER chosen by cross-validation.

Corpus	CRF Model	GraphNER hyperparameters (α, μ, ν,$\#iterations$)
AML	BANNER	$(0.02, 10^{-6}, 10^{-6}, 2)$
	BANNER-ChemDNER	$(0.02, 10^{-6}, 10^{-6}, 2)$
BC2GM	BANNER	$(0.02, 10^{-6}, 10^{-4}, 2)$
	BANNER-ChemDNER	$(0.02, 10^{-6}, 10^{-6}, 3)$

clature maintained by HGNC for human genes. This standardized nomenclature would simplify the manual annotation task for the annotators, and would likely improve the performance of the named entity recognition tools as well. BC2GM annotations were performed by undergraduate students with limited training and limited subject knowledge, whereas the AML corpus was curated and edited by subject experts in the clinical genetics domain. Due to the difference in expertise in the annotators for the two projects, we expected a higher error rate in the BC2GM manual annotations.

Nonetheless, GraphNER has improved both baselines on AML corpus by showing higher precision. As discussed in detail in Section 3.3.1, the improvements in precision were statistically significant.

We also applied the character-based bi-directional LSTM with attention mechanism [Rei et al., 2016], which was considered the state-of-the-art on BC2GM at the time, on the AML dataset. Since it needs a dev-set, we partitioned the AML train set to train/dev subsets (82%/18% in sentences). We also re-trained BANNER-ChemDNER and GraphNER with BANNER-ChemDNER on the new smaller train subset to have a fair comparison. Their system achieved an F-Score of 93.62 which was lower than both BANNER-ChemDNER (F-Score=94.32) and GraphNER with BANNER-ChemDNER (F-Score=94.54) when trained on the smaller train subset.

The hyper-parameters used to generate reported results for GraphNER were all chosen by cross-validation over different train:test splits. Table 3.3 shows the parameters used for each of the systems.

Finally, it is worth mentioning that all results reported in Table 3.1 and Table 3.2 were obtained with second-order CRF's and using java version 1.7. However, while we obtained different numbers for different CRF orders (1 or 2) or java versions (1.7 or 1.8), GraphNER always improved both baselines, and this improvement was consistently due to higher precision.

3.3.1 Significance testing of the results

We used SIGF [Padó, 2006b] to test for significant changes to precision, recall, and F-scores with the addition of GraphNER. *SIGF* is an implementation of an assumption-free significance test based on randomization [Yeh, 2000]. When testing the significance of the difference in the performance of two models m1 and m2, SIGF repeatedly constructs statistically identical models m3 and m4 by taking the predictions that are produced by m1 or m2 but not both of them, and randomly assigning

Table 3.4: Null hypotheses tested using SIGF [Padó, 2006b] and the corresponding p-values.

Crf Model	Measure	Corpus	p-value
BANNER	F-score	BC2GM	$< 10^{-4}$
BANNER_ChemDNER	F-score	BC2GM corpus	$< 10^{-4}$
BANNER	F-score	AML	0.018
BANNER	Recall	AML	0.72
BANNER	Precision	AML	0.0003
BANNER_ChemDNER	F-score	AML	0.035
BANNER_ChemDNER	Recall	AML	0.74
BANNER_ChemDNER	Precision	AML	0.003

those predictions to either m3 or m4. How often m3 and m4 produce results that are at least as different as results of m1 and m2 is interpreted as the p-value in the significance test.

We used SIGF with 10,000 repetitions to test the null hypotheses corresponding to rows in Table 3.4. Each row in Table 3.4 lists a CRF model K, a measure M, and a corpus C which corresponds to a null hypothesis following the pattern K and GraphNER with K have the same M on C. For example, the null hypothesis corresponding to the first row in Table 3.4 is "BANNER and GraphNER with BANNER have the same F-score on BC2GM corpus".

Bonferroni correction for multiple testing changes the first $\alpha = 0.05$ to $\alpha = 0.006$. The F-score improvement of GraphNER over both baselines (BANNER and BANNER_ChemDNER) is statistically significant while working with the BC2GM corpus. Although the F-score and recall improvements of GraphNER over BANNER and BANNER_ChemDNER were not statistically significant for the AML corpus, the improvements in precision were significant.

3.3.2 Effect of different vertex representations

Table 3.5 shows how GraphNER improves upon the performance of purely supervised models such as BANNER and BANNER-ChemDNER on the BC2GM corpus by using semi-supervised graph-propagation using different feature sets. The hyper-parameters used are shown in Table 3.3. It is interesting that while using all features led to the best results with both BANNER and BANNER-ChemDNER, GraphNER consistently improved the baselines even when only 40 ($MI > 0.01$) and 85 ($MI > 0.005$) features were used in graph construction.

Another variable in graph construction is K, the degree of the graph. While $K = 10$ was the default, we changed K to be 5 for one of the graphs (all features used for vector representation) and saw a small degradation in F-score (going from 87.34 to 87.32).

3.3.3 Added time and memory cost over supervised CRFs

In our tests, GraphNER has consistently improved both BANNER and BANNER-ChemDNER supervised CRF models, and this improvement is achieved with a small additional run time cost over the purely supervised models. Figure 3.2 shows the extra time GraphNER needs to train and test

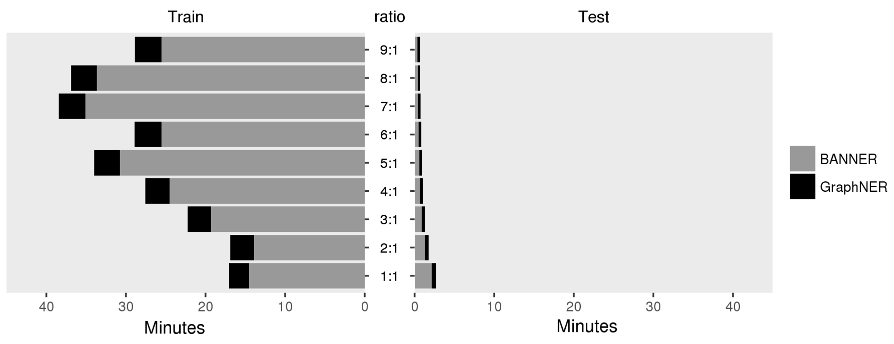


Figure 3.2: Time cost to train and test BANNER and GraphNER on the BC2GM dataset. The ratio indicates the relative sizes of train:test partitions.

over BANNER, using different ratios of train:test splits of the BC2GM corpus. All experiments were done in a GNU/Linux environment on a Dell Precision Tower 7910 with 16 Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz cores and 64GB of RAM.

We observed similar patterns when we experimented with the BANNER-ChemDNER as the supervised model and with the AML dataset. In all our experiments we used the all-features graph constructed over the relevant corpus and ran the train and test procedures over 10 instances of each train:test ratio.

During graph propagation, GraphNER loads the entire graph into memory, which represents the peak memory usage for the algorithm. Thus, the memory footprint of GraphNER can be estimated by the size of the graph description files. This is about 90 MB and 105 MB for the all-feature graphs constructed over AML and BC2GM corpora, respectively.

Table 3.5: Effect of choice of feature sets used in graph construction. The bold figures indicate best performance for GraphNER using BANNER and BANNER-ChemDNER models.

Method	CRF Model	Vector-Representation	K	F-Score (%)
BANNER	-	-	10	84.38
BANNER-ChemDNER	-	-	10	86.49
GraphNER	BANNER	All-features	10	**85.83**
	BANNER	Lexical-features	10	85.43
	BANNER	MI > 0.005	10	85.01
	BANNER	MI > 0.01	10	85.00
	BANNER-ChemDNER	All-features	10	**87.34**
	BANNER-ChemDNER	Lexical-features	10	87.23
	BANNER-ChemDNER	MI > 0.005	10	87.09
	BANNER-ChemDNER	MI > 0.01	10	87.12
	BANNER-ChemDNER	All-features	5	87.32

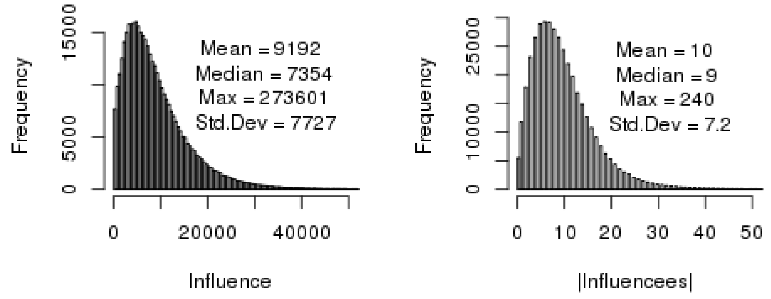


Figure 3.3: Histogram of influence and number of influencees for BC2GM all-features graphs.

3.3.4 Statistics of all-feature graphs

In the all-feature graphs, which led to the best results for both corpora, we note that the numbers of vertices (406,179 for BC2GM, and 348,683 for AML) are comparable. The percentage of labeled vertices is high in both graphs (77.2% for BC2GM, and 51.7% for AML). This is due to the fact that we are experimenting in a transductive setting, where the only unlabeled data is the test data. However, the percentage of labeled vertices, and especially the percentage of positively labeled vertices (vertices that appeared as the beginning or inside of a gene in the train set) show marked differences, though they are quite low in both graphs (8.5% in BC2GM, and 1.75% for AML). The low percentage of positively labeled vertices in both graphs explains the higher precision of GraphNER.

Both graphs are weakly connected, and by construction (because $K = 10$), the outgoing degree is 10 for all vertices. It follows that the number of edges is exactly 10 times the number of vertices in both graphs. The fact that any given vertex has exactly 10 nearest neighbors, however, does not mean that each vertex is the nearest neighbor to exactly 10 vertices.

The nearest neighbors of a vertex influence its label distribution in graph propagation. To formalize this, we can define Influencees(v) as the set of vertices that v influences, that is the set of vertices to which v is a nearest neighbor. Then we can define Influence(v) to be the sum of edge weights that connect v to Influencees(v):

$$\text{Influence}(v) = \sum_{k \in \text{Influencees}(v)} w_{k,v}.$$

Using these definitions, we can use $|\text{Influnecees}(v)|$ and Influence(v) as measures of a vertex's influence. Figure 3.3 shows the histogram of these measures over all vertices in the all-features BC2GM graph. As expected, the plots indicate that most vertices have low influences. We obtained similar histograms for the AML all-features graph.

3.3.5 Qualitative performance differences

GraphNER consistently improved the precision of the purely supervised CRF models, both BANNER and BANNER-ChemDNER, when trained on either the BC2GM corpus or AML corpus. To evaluate this outcome qualitatively, we performed a manual review of the false positive and false negative calls when using the corpus as the gold standard. To reduce the time of this task, we randomly sampled 280 errors from the 5000 BC2GM corpus errors. Due to the small number of total errors in the AML corpus, we reviewed all of the 454 AML corpus errors.

We categorized each false positive or false negative entity into one of two categories, either gene-related or spurious. Gene-related entities included actual genes, gene families, or specific protein domains. For example, "E3 ubiquitin", a gene family, was a gene-related false positive in BANNER that was corrected by GraphNER. Spurious entities were entirely erroneous annotations that did not thematically relate to genes or proteins. For example, "Ann Arbor" was a spurious false positive in BANNER that was also corrected by GraphNER.

Figure 3.4 shows an UpSet plot [Lex et al., 2014] of the intersections of false positive calls in GraphNER versus the BANNER-ChemDNER in the AML corpus. UpSet plots visualize combinatorial set intersections with a bar plot. The intersecting set for each bar is represented by the ball and stick model on the x axis. A chi-square two-sample test for equality of proportions with continuity correction found no significant difference in the relative proportion of false positives in gene-related entities in the supervised CRF model and the semi-supervised GraphNER model when trained and tested with the AML corpus (p=0.56). The difference in AML corpus precision noted in Table 3.2 was due to a quantitative difference in total annotations, rather than a difference in the quality of annotations. Conversely, a chi-square test of the relative proportion of gene-related entities was significant when both tools were trained and tested on the BC2GM corpus (p=0.029). Figure 3.5 shows substantial quantitative and proportional decreases in the number of spurious false positive calls when using GraphNER compared to BANNER-ChemDNER. GraphNER corrected several spurious annotations from the supervised BANNER and BANNER-ChemDNER CRF models, resulting in proportionally fewer spurious entities when trained on the BC2GM data.

In comparison to the AML corpus training results, a significantly higher proportion of false positive and false negative annotations from the BC2GM corpus training were in fact caused by a higher proportion of incorrect annotations in the gold standard corpus. For example, GraphNER correctly tagged "GRK6" as a gene, but our testing protocol counted this as a false positive due to the lack of an annotation in the BC2GM gold standard. The discrepancy in the proportion of false corpus annotations between the AML and BC2GM corpora was highly significant based on a chi-square test of proportions ($p < 2.2 \times 10^{-16}$). These results support the conclusion that GraphNER is more robust to training on sets with a higher rate of annotator errors, and GraphNER corrected spurious BC2GM annotations in BANNER and BANNER-ChemDNER. This advantage was less apparent when training on the AML corpus, which had fewer actual errors in its ground truth annotations.

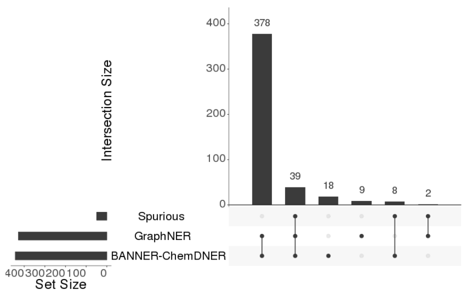


Figure 3.4: Upset plot of qualitative false positive differences between GraphNER and BANNER-ChemDNER trained on the AML corpus.

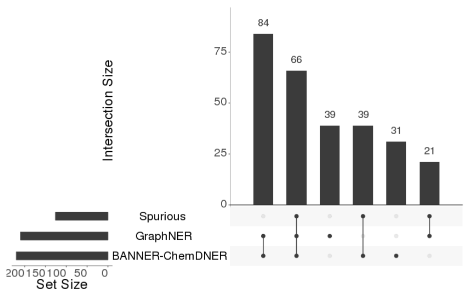


Figure 3.5: Upset plot of qualitative false positive differences between GraphNER and BANNER-ChemDNER trained on the BC2GM corpus.

3.4 Summary

We presented GraphNER, a semi-supervised tool for detecting gene mentions in biomedical literature. We show that our semi-supervised approach improves upon BANNER and BANNER-ChemDNER, two previous state-of-the-art CRF models for supervised gene mention detection. We benchmark this approach on two different biomedical text corpora: annotated abstracts of the BioCreative II gene mention shared task corpus and annotated full-text articles related to acute myeloid leukemia.

GraphNER outperformed the baselines regardless of the features used from the CRF models in the graph construction phase of our approach. The higher F-score produced by GraphNER was consistently due to higher precision over the base model. This is expected given the low percentage of positively labeled vertices.

We used GraphNER in a transductive setting where all the unlabeled data came from the test set (we consider the test set unlabeled since we do not know the true labels until we do the evaluation). Given the semi-supervised nature of GraphNER, we expect even higher performance when the tool is provided abundant unlabeled data. Though this would present an algorithmic challenge, as the time complexity of graph construction grows rapidly with increased dataset size, and becomes prohibitive for resources as large as the complete PubMed database. However, once the graph is constructed, we have shown that the time cost of semi-supervised learning with GraphNER is low in comparison to the CRF purely supervised model.

As well as GraphNER worked in its own turn, it could not catch up with the power of neural networks equipped with contextualized embeddings pre-trained on a large amount of unlabeled data. The next chapter presents our experiments running a deep learning package (AllenNLP) using ELMO [Peters et al., 2018b] contextualized embeddings and discusses later advances in the field.

Chapter 4

Contextualized Embeddings for Biomedical NER

This chapter focuses on an alternative semi-supervised approach to biomedical NER. Here, we report on a pipeline built on Embeddings from Language Models (ELMO) and a deep learning package for natural language processing (AllenNLP). We consider this a semi-supervised approach as large amounts of unlabelled data are used to train context-aware token embeddings. We trained ELMO on a dataset of biomedical papers and incorporated these context-aware token embeddings in the LSTM-CRF model used by AllenNLP for named entity recognition. We show these representations improve named entity recognition for different types of biomedical named entities. We also achieved a new state of the art in gene mention detection on the BioCreative II gene mention shared task, even though our results were surpassed by later models that we briefly describe in Section 4.6.

4.1 Introduction

The last decade witnessed substantial improvements in machine learning methods and their application to natural language processing tasks. Recently, Peters et al. [2018a] introduced ELMO (Embeddings from Language Models), a system for deep contextualized word representation, and showed how it can be used in existing task-specific deep neural networks. The method improves the state of the art over a variety of NLP tasks such as question answering, word sense disambiguation, sentiment analysis, and named entity recognition. The developers of the tool also provide an ELMO model pre-trained on the Billion-word Language Model (LM) dataset [Chelba et al., 2014] as an off-the-shelf tool for use in a wide variety of NLP tasks and domains.

In this chapter, we investigate the efficacy of off-the-shelf ELMO and the effect of an in-domain training set for ELMO in biomedical NER applications. We observe that:

1. off-the-shelf ELMO has room for improvement in domain-specific applications;

2. ELMO consistently improves biomedical NER when trained on in-domain data;

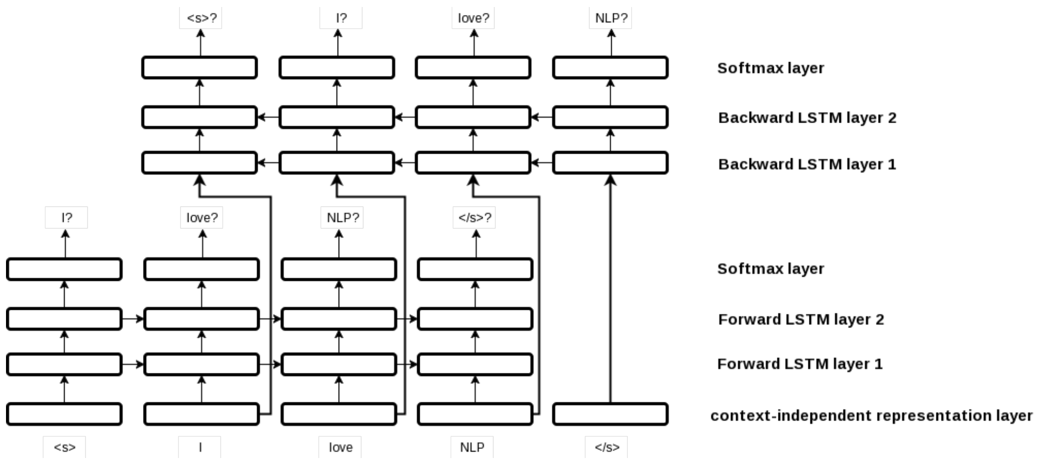


Figure 4.1: ELMO is a bidirectional LSTM for language modelling where the next or precedent tokens are predicted from the softmax layers over forward and backward LSTMs respectively.

3. such improvement can be achieved even when the in-domain training dataset is smaller than the Billion-word LM data;
4. the resulting model achieves the highest precision/recall/F1 scores so far on BC2GM.

We explain ELMO and AllenNER, the named entity recognizer we used, in Sections 4.2 and Section 4.3. Then, we describe our datasets in Section 4.4, and we move on to report the results in Section 4.5 and describe the later advances in biomedical NER in Section 4.6.

4.2 ELMO

ELMO [Peters et al., 2018a] is a system that produces context-aware embeddings for word tokens. Similar to traditional context-independent word embeddings such as GloVe [Pennington et al., 2014] and Word2Vec [Mikolov et al., 2013], ELMO representations can be used as input to a neural network for downstream tasks. However, ELMO is different from the traditional word embeddings in that it gives the representation of the word in the context of the specific given sentence; hence it is a context-aware word token representation as opposed to a word type representation.

It is trained using a language modeling objective function, where the objective is to predict the next word in the sequence; either sequentially left to right or right to left. As such, it can be viewed as learning a token level representation of words for a task that can be trained on unannotated data. These word representations can then be used for a task that is trained on labeled data. In our case, the task is biomedical named-entity recognition.

Figure 4.1 shows the architecture of ELMO as a recurrent language modelling network. The input to this system is a sequence of words $w_1 w_2 \dots w_i \dots w_n$. First, each word is converted to a context-independent embedding by a convolutional neural network (CNN) over its characters. These character-based representations are then fed into a two-layer bidirectional long short-term memory

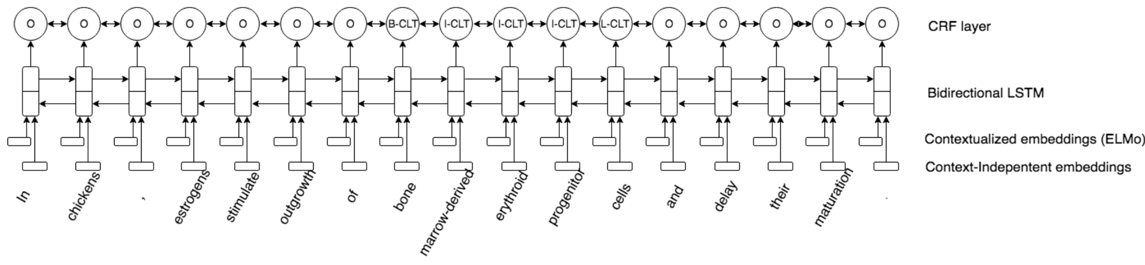


Figure 4.2: Architecture of LSTM-CRF Lample et al. [2016] with ELMO. Traditional word embeddings and ELMO representations are concatenated and fed into a bidirectional LSTM. A CRF layer on top of bidirectional LSTM takes local label dependencies into account. At training time the log likelihood of gold label sequences is maximized. At test time, Viterbi [Viterbi, 1967] algorithm is used to decode the complete label sequence. -CLT in the labels of the example indicate cell_type entity.

(LSTM) [Hochreiter and Schmidhuber, 1997] recurrent neural network. The output of the second layers of the forward and backward LSTMs are fed to a soft-max layer to predict w_{i+1} and w_{i-1}, respectively, at each position i.

Task-specific learned weights can be used later to combine all layers in the ELMO model at position i and form the task-specific "ELMO representation of w_i".

Peters et al. [2018a] showed that different layers in this deep recurrent model learn different aspects of a given token. The lower layers learn more syntactic features whereas higher layers learn the contextual aspects of the word. They linearly combined the layers using task-dependent weights, and their experiments show that for Named Entity Recognition tasks, the layers are combined with effectively the same weights.

4.3 Named Entity Recognition with AllenNLP

In our pipeline, we couple ELMO embeddings to AllenNLP [Gardner et al., 2017] for NER tasks.

AllenNLP uses a bidirectional two-layer LSTM-CRF [Lample et al., 2016] to perform NER as a sequence tagging task. Each word is tagged with an output that marks if it is at the beginning (B), in the middle (I), at the end (E or L), or outside (O) of an entity type. One-word entities are also marked (as S or U). For example, B-Gene and I-Gene stand for beginning and inside of a Gene, whereas B-DNA and E-DNA stand for beginning and end of a DNA entity type.

AllenNLP embeds the input words using a Convolutional Neural Network over characters. Rei et al. [2016] showed that word embeddings from character compositions outperform lookup embeddings such as word2vec, when used for named entity recognition.

AllenNLP combines the layers in ELMO Model using learned task-specific weights, concatenates the result for each token to context-independent word embeddings, and feeds the concatenation into the LSTM-CRF as illustrated in Figure 4.2.

4.4 Datasets

We collected a focused domain-specific subset of PubMed Central (PMC) documents and used them for training ELMO. This dataset is described in detail in Section 4.4.1. We report results on two benchmark datasets, which we describe in Sections 4.4.2 and 4.4.3.

4.4.1 ELMO Training Set

We downloaded a subset of PMC documents (available at ftp://ftp.ncbi.nlm.nih.gov/pub/pmc) in May 2018 and picked 3960 full-text documents that had a Medical Subject Heading (Mesh) term *cancer*. We ran StanfordNLP/CoreNLP toolkit [Manning et al., 2014a] on these documents for sentence splitting and tokenization. Tokens of each sentence were joined with a space character to form the sentences in the training set. This dataset contains about 21 million tokens and is substantially smaller than the One Billion Word Benchmark [Chelba et al., 2014] that Peters et al. [2018a] used for training ELMO but contains in-domain text that is more likely to benefit the biomedical text analysis of interest in this work.

4.4.2 BC2GM

BC2GM is the data set for the BioCreative II Gene Mention detection shared task [Smith et al., 2008]. This dataset contains 15000 training and 5000 test sentences, all from PubMed abstracts. Gold annotations give the gene mentions by providing the sentence ID, the start and end characters of the mention (ignoring all space characters), and the mention itself.

4.4.3 JNLPBA

JNLPBA [Kim et al., 2004] is the dataset for a shared task on biomedical entity detection. Its training set contains 2000 GENIA [Kim et al., 2003] abstracts, which the authors had collected by searching MEDLINE abstracts for Mesh terms *human*, *blood cells* and *transcription factors*. The test set contains 404 abstracts, half of which are from the same domain and the other half are from a super-domain of *blood cells* and *transcription factors*. The documents are annotated for protein, DNA, RNA, cell_line, and cell_type entity classes.

4.5 Results

Table 4.1 shows the leading results in the literature (top four rows) in comparison with our results (bottom three rows) on the BC2GM dataset.

In the year it was held, Ando [2007] had won the challenge with a semi-supervised system equipped with a lexicon and a combination of several classifiers. Gimli [Campos et al., 2013] is a supervised method based on conditional random fields (CRF) [Lafferty et al., 2001a] with hand-engineered features that was the state of the art for gene mention detection before Graph-NER [Sheikhshab et al., 2018b] obtained a higher F-score. GraphNER obtained the distributions

Model	Prec. (%)	Rec. (%)	F1 (%)
Ando (2007)	88.48	85.97	87.21
Gimli (2013)	90.22	84.32	87.17
GraphNER (2018)	89.18	85.57	87.34
Rei et al. (2016)	-	-	87.99
AllenNER with no ELMO (Baseline)	88.05	88.72	88.39
AllenNER + off-the-shelf ELMO	89.03	87.95	88.49
AllenNER + ELMO Trained In-Domain	**89.86**	**89.59**	**89.72***

Table 4.1: Leading results in the literature (up) in comparison with our results (down) on BC2GM dataset. Best performances are bold and statistical significance is shown by a star.

over labels from the CRF and propagated them on a graph of 3-grams similarities constructed over BC2GM.

Rei et al. [2016] set the previous state of the art on BC2GM by applying an LSTM-CRF based system with attention to characters. Our baseline, AllenNER (described in detail in Section 4.3) is similar to their system, except AllenNER uses a convolutional neural network (CNN) over characters instead of using the attention mechanism.

Our results, the lower part of Table 4.1, show that using the off-the-shelf ELMO, that is trained on the one billion word language model benchmark [Chelba et al., 2014], improves the precision at the expense of recall, modestly improving the F1 score. When ELMO is trained on approximately 21 million in-domain tokens both precision and recall are considerably improved resulting in a more than 1 percentage point improvement in the F1-score. A significance test using SIGF [Padó, 2006a] showed that this improvement is statistically significant ($p < 10^{-5}$), and the one from off-the-shelf ELMO is not ($p > 0.02$).

Table 4.2 shows our F1 scores on JNLPBA. It is evident from the table that using ELMO leads to salient improvements over the baseline if it is trained in-domain. The off-the-shelf ELMO has improved the performance for RNA and cell_line entity types but hurt the performance for protein, DNA, and cell_type. In-domain ELMO always obtains the best performance except for RNA entity type where it is competitive with off-the-shelf ELMO and considerably better than the baseline.

Statistical significance tests using SIGF [Padó, 2006a] showed that most differences in Table 4.2 are not statistically significant after Bonferroni correction for multiple testing. The only statistically significant improvements are those of in-domain ELMO for protein and cell_type mention detections over both off-the-shelf ELMO and baseline. This could be due to the fact that proteins and cell_types are more frequent in JNLPBA when compared to other entities. Still, it is interesting to note that the in-domain trained ELMO model consistently performed better than the alternative

Model	protein	DNA	RNA	cell_line	cell_type
AllenNER with no ELMO (Baseline)	70.47	70.87	63.94	57.17	73.55
AllenNER + off-the-shelf ELMO	69.96	70.56	**65.38**	59.70	73.21
AllenNER + ELMO Trained In-Domain	**75.08***	**73.13**	65.17	**61.15**	**75.87***

Table 4.2: F1-scores (%) for different entity types in JNLPBA dataset. Best performances are bold and statistical significance is shown by a star

Entity type	Training	Test
protein	30,269	5,067
DNA	9,533	1,056
RNA	951	118
cell_type	6,718	1,921
cell_line	3,830	500

Table 4.3: Frequencies of different entity types in training and test sets of JNLPBA

ELMO models in all but one NER task. Table 4.3 shows the frequencies of different entity types in training and test sets of JNLPBA.

Our results on JNLPBA were not the state of the art. Habibi et al. [2017] report F1 scores as high as 77.25% for protein and 63.31% for cell_line entity types when they use word embeddings trained on the union of (nearly 23 million) PubMed abstracts, (nearly 700,000) PMC full articles, and (approximately four million) English Wikipedia articles as input to an LSTM-CRF. Nevertheless, our results show the positive effect of using in-domain trained ELMO representations compared to a very strong baseline.

4.6 Beyond our work

After we published our work [Sheikhshab et al., 2018a], BERT [Devlin et al., 2019] was introduced as an alternative pre-trained contextual embedding and was shown to outperform ELMO on all the General Language Understanding Evaluation(GLUE) benchmark [Wang et al., 2019] tasks thanks to the larger corpus it was pre-trained on. Later, Lee et al. [2020] introduced BioBERT, which was different from BERT only in that it was pre-trained on PubMed abstracts after initialization to BERT. Lee et al. [2020] empirically showed that BioBERT outperforms BERT in biomedical NER, among other tasks confirming our observation that in-domain training data is beneficial. In parallel to BioBERT, Jin et al. [2019] introduced BioELMO, a domain-specific version of ELMO pre-trained on 10M PubMed abstracts, and confirmed the utility of in-domain training data for biomedical NER yet again.

To the best of our knowledge, the current state of the art in biomedical NER is an attention segmental recurrent neural network (ASRNN) [Lin et al., 2021], which represents characters, words, and variable-length segments with hierarchical attention. Lin et al. [2021] report an F1 score of 90.8 on BC2GM while reporting an F1 of 90.6 from BERT in the same table. According to their reported results, ASRNN achieves an F1 score of 74.0 on the JNLPBA dataset, when BERT achieves 73.7. Whether or not this improvement justifies their more complicated system is not entirely clear.

Finally, it is worth mentioning that Elangovan et al. [2021] have designed and run experiments to investigate how similarity between the train and test sets have affected results of several recent NLP tools and have included biomedical NER. They experimented with a finetuned BERT model on BC2GM, focusing on test instances in the four quartiles when instances are sorted according to their similarity with training instances. They defined instance similarity as cosine similarity of vector representations of instances where vectors were constructed based on unigram, bigram, and trigrams. Elangovan et al. [2021] found that the majority of test instances (74.1%) fell into the second quartile over which the finetuned BERT achieved an F1 score of 82.4. They also observed that focusing on instances in the first quartile (least similar ones to the training instances), led to a much lower F1 score of 74.5. It is interesting to see if the same drop in performance happens with BioBERT, BioELMO, or even GraphNER.

4.7 Summary

In this chapter, we showed that token level context-aware embeddings trained on an auxiliary task of language modeling using the ELMO toolkit can be used to consistently improve biomedical named entity recognition tasks, but only when the pre-trained embeddings are trained on in-domain biomedical data. Using this technique we achieved a new state of the art (at the time of our publication [Sheikhshab et al., 2018a]) on the BioCreative II dataset for gene mention detection. Later methods have surpassed our results but repeatedly confirmed our conclusion that in-domain training data is beneficial to the biomedical NER task.

This chapter concluded our work on semi-supervised biomedical named entity recognition. The next part focuses on our work on evaluating Word Sense Disambiguation.

Part II

Evaluating Word Sense Disambiguation

Chapter 5

Current State of Sense Disambiguation

Word sense disambiguation (WSD) is the task of picking the correct sense of a target word in a given context from a set of predefined senses. It is divided into two tasks based on whether or not only certain target words are considered for disambiguation. In lexical sampling, as opposed to WSD All-Words, the focus is on few target words. In this thesis, unless otherwise specified, WSD means the English All-Words WSD.

A sense inventory that lists predefined senses for all target words, is at the core of the WSD task. In English WSD, WordNet [Miller, 1995] is the go-to sense inventory. Recent WSD methods report results on the unified benchmark datasets of Raganato et al. [2017b] where the gold senses come from WordNet. In this chapter, we give an overview of the current evaluation framework and comment on aspects of it that are less than ideal.

We briefly describe WordNet in Section 5.1, the datasets in the benchmark in Section 5.2, the recent WSD methods and the progress of state-of-the-art on the benchmark datasets in Section 5.3, and our comments on the current state of affairs in Section 5.4.

5.1 An overview on WordNet

WordNet [Miller, 1995] is a large English lexical database that splits each word-POS pair into word senses and then groups synonyms of these word senses into synsets. Each synset has a description—its gloss, and may also have examples. Each WordNet synset is also labeled with a lexicographic category such as ANIMAL or ARTIFACT. There are few such categories (26 and 15for nouns and verbs) usually referred to as super-senses or coarse-grained WordNet sense tags. In this thesis, a WordNet sense will mean fine-grained WordNet sense.

WordNet is structured as a network of relationships over synsets where the relations between them include

1. is-A relationship between noun synsets formally known as hypernymy-hyponymy with the example of *mouse#4* being a hyponym of *electronic device#1* and *electronic device#1* being a hypernym of *mouse#4*;

2. part-whole relationship between noun synsets formally known as meronymy with the example of *arm#1* being a meronym of *human#1*;
3. specific-manner relationship between verbs formally known as troponymy with the example of *wolf#1* and *nibble#3* both being troponyms of *eat#1*;
4. opposit-meaning relationship between adjectives formally known as antonymy with the example of *ugly#1* begin an antonym of *beautiful#1*;
5. cross-POS relations that link the otherwise disjoint subnets of WordNet nouns, verbs, adjectives, and adverbs with the example of morphosemantics that links *beautiful#1* to *beautician#1*, *beauteous#1*, and *beautify#1,2,3* because they are semantically related and share a stem.

5.2 Unified Benchmark datasets of Raganato et al. [2017b]

Raganato et al. [2017b] took seven existing all-word English WSD datasets and unified them to have the same format, preprocessing, and WordNet sense inventory. They used the Stanford CoreNLP toolkit Manning et al. [2014b] for preprocessing and extracted the POS tags and Lemmas. To unify the sense inventory, they mapped all annotations that were according to previous versions of WordNet to version 3.0.

Here is a list of datasets in this unified framework.

- Senseval-2 (SE2) [Edmonds and Cotton, 2001]
- Senseval-3 task 1 (SE3) [Snyder and Palmer, 2004]
- SemEval-07 task 17 (Sem-07) [Pradhan et al., 2007]
- SemEval-13 task 12 (Sem-13) [Navigli et al., 2013]
- SemEval-15 task 13 (Sem-15) [Moro and Navigli, 2015]
- Semcor [Miller et al., 1994a]
- OMSTI (One Million Sense-Tagged Instances) [Taghipour and Ng, 2015a]

Table 5.1 shows the original sense inventory as well as some statistics for these datasets. The ambiguity of each dataset is computed by dividing the total number of candidate senses by the number of annotations.

It is worth mentioning that two of these datasets, SemEval-13 task 12 and SemEval-15 task 13, were annotated according to two different versions of BabelNet [Navigli and Ponzetto, 2012], a multilingual network over concepts automatically constructed over Wikipedia with regards to WordNet 3.0. As a result, these tasks were appropriate for not only WSD but also entity linking. However, if WSD is the only task, the annotations are already from WordNet 3.0.

	Evaluation Sets					Training Sets	
	SE2	SE3	Sem-07	Sem-13	Sem-15	Semcor	OMSTI
Original Senses	WN 1.7	WN 1.7.1	WN 2.1	WN 3.0	WN 3.0	WN 1.4	WN 3.0
#Docs	3	3	3	13	4	352	-
#Sents	242	352	135	306	138	37,176	813,798
#Tokens	5,766	5,541	3,201	8,391	2,604	802,443	30,441,386
#Annotations	2,282	1,850	455	1,644	1,022	226,036	911,134
#Sense types	1,335	1,167	375	827	659	33,362	3,730
#Word types	1,093	977	330	751	512	22,436	1,149
Ambiguity	5.4	6.8	8.5	4.9	5.5	6.8	8.9

Table 5.1: Datasets in the unified framework of Raganato et al. [2017b]

5.3 WSD State of the Art

Raganato et al. [2017a] reported results of many existing WSD methods on the benchmark datasets when introducing them for the first time. They included both supervised and knowledge base methods. They reported results of IMS [Zhong and Ng, 2010], IMS+embeddings [Taghipour and Ng, 2015c; Rothe and Schütze, 2015; Iacobacci et al., 2016], and Context2Vec [Melamud et al., 2016] as supervised methods and results of LESK [Lesk, 1986], UKB [Agirre and Soroa, 2009; Agirre et al., 2014], and Babelfy [Moro et al., 2014] as knowledge-based methods. They also presented two baselines, MFS, the most frequent sense in a train set, as a baseline for supervised methods, and WN 1^{st} sense, the first sense in WordNet, as a baseline for knowledge-based methods.

Their results showed that supervised methods performed better than knowledge-based methods in general. Later advancements in knowledge-based methods [Chaplot and Salakhutdinov, 2018; Wang et al., 2020] never caught up with their supervised competitors either.

For supervised methods Raganato et al. [2017a] showed that using the union of Semcor [Miller et al., 1994b] and OMSTI [Taghipour and Ng, 2015b] as training sets yields only slightly better results over using Semcor only. This might be the reason the later methods did not include OMSTI in their training sets.

Table 5.2 shows the progress of WSD methods on the benchmark datasets of Raganato et al. [2017b]. The rest of this section gives detailed backgrounds for the methods of Table 5.2 except for the baselines which we have already described.

5.3.1 Babelfy

Babelfy [Moro et al., 2014] is a graph-based method to jointly tackle both entity linking (EL) and WSD. It uses a knowledge graph, BabelNet [Navigli and Ponzetto, 2012], that includes WordNet among other resources. Babelfy uses vertices of the graph that are concepts or entities, as well as the presence of edges that encode relations between concepts/entities even though it discards the edge type.

Method	SE2	SE3	SE 2007	SE 2013	SE 2015
MFS baseline	65.6	66.0	54.5	63.8	67.1
WN 1^{st} sense baseline	66.8	66.2	55.2	63.0	67.8
Babelfy	67.0	63.5	51.6	66.4	70.3
IMS$_{-s}$+emb	73.3	69.6	61.1	66.7	70.4
WSD-TM [Chaplot and Salakhutdinov, 2018]	69.0	66.9	55.6	65.3	69.6
KEF [Wang et al., 2020]	69.6	66.1	56.9	68.4	72.3
Bi-LSTMatt+LEX [Raganato et al., 2017a]	72.0	69.4	63.7	66.4	72.4
supWSDemb [Papandrea et al., 2017]	73.8	70.8	64.2	67.2	71.5
GASext [Luo et al., 2018]	72.2	70.5	–	67.2	72.6
BERT [Hadiwinoto et al., 2019]	75.5	73.6	68.1	71.1	76.2
GlossBERT [Huang et al., 2019]	77.7	75.2	72.5	76.1	80.4
hypernyms+WNGC [Vial et al., 2019]	79.7	77.8	73.4	78.7	82.6
BEM [Blevins and Zettlemoyer, 2020]	79.4	77.4	74.5	79.7	81.7
EWISER+WNGC [Bevilacqua and Navigli, 2020]	80.8	79.0	75.2	80.7	81.8

Table 5.2: Progress over Datasets in the unified framework of Raganato et al. [2017b]. First section is from original paper, the second section is for knowledge-based methods, and the last section is for supervised methods.

Babelfy works in three steps. As the first step, it finds a signature, i.e. the set of related vertices, for each vertex. It won't simply use the adjacent vertices as the signature since there are unrelated concepts/entities that were mistakenly connected in the automatically-constructed graph as well as related concepts/entities that are indirectly connected. It will use random walks with restarts from a vertex to find all related vertices (therefore allowing to discover concepts/entities further down the road) while removing vertices that were not visited more often than a threshold. This step is done only once for the given knowledge graph.

The second and third steps are done given a particular text (for example a sentence). The second step is to find all the fragments of the sentence allowing for overlapping fragments, and finding candidate vertices for each fragment by lexical similarity. Since Babelfy is designed to do both EL and WSD jointly, it does not assume exact matches and allows for substrings/superstrings.

Given the fragments and their candidate meanings, the third step is to find the best fit for each fragment. This is done by making a subgraph over the candidate meanings of different fragments according to the vertex signatures and picking the densest subgraph. The intuition behind this heuristic is to keep the chosen meanings for different fragments in a way that yields a coherent meaning for the given text as a whole.

5.3.2 IMS$_{-s}$+emb

IMS (It Makes Sense) is a supervised WSD method that represents a word in the context as a vector of features (example features include surrounding words and their POS tags) and feeds them into a support vector machine as the classifier. Raganato et al. [2017a] trained skip-gram Word2Vec [Mikolov et al., 2013] on the English ukWaC corpus [Baroni et al., 2009] and used it

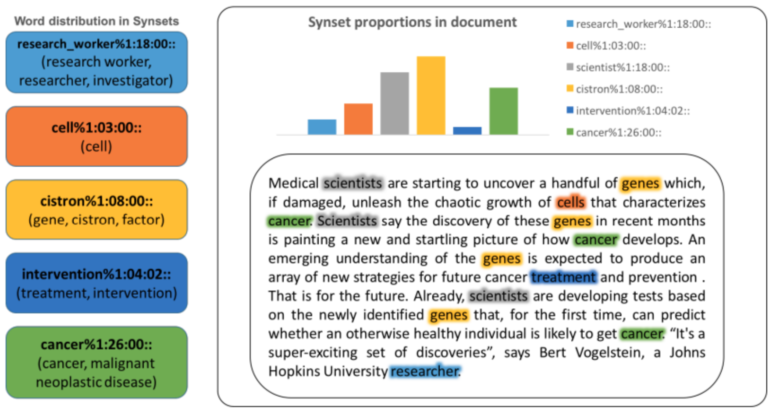

Word distribution in Synsets
research_worker%1:18:00:: (research worker, researcher, investigator)
cell%1:03:00:: (cell)
cistron%1:08:00:: (gene, cistron, factor)
intervention%1:04:02:: (treatment, intervention)
cancer%1:26:00:: (cancer, malignant neoplastic disease)
Synset proportions in document
research_worker%1:18:00::
cell%1:03:00::
scientist%1:18:00::
cistron%1:08:00::
intervention%1:04:02::
cancer%1:26:00::
Medical scientists are starting to uncover a handful of genes which, if damaged, unleash the chaotic growth of cells that characterizes cancer. Scientists say the discovery of these genes in recent months is painting a new and startling picture of how cancer develops. An emerging understanding of the genes is expected to produce an array of new strategies for future cancer treatment and prevention. That is for the future. Already, scientists are developing tests based on the newly identified genes that, for the first time, can predict whether an otherwise healthy individual is likely to get cancer. "It's a super-exciting set of discoveries", says Bert Vogelstein, a Johns Hopkins University researcher.

Figure 5.1: WSD-TM at work. Each document is a distribution over synsets, and each synset is a distribution over words. Figure taken from [Chaplot and Salakhutdinov, 2018].

as features in IMS implementation following suggestions of Iacobacci et al. [2016]. They observed that dropping the surrounding words (IMS$_{-s}$) will yield higher results.

5.3.3 WSD-TM

WSD-TM [Chaplot and Salakhutdinov, 2018] has a generative model similar to Latent Dirichlet Allocation(LDA) [Blei et al., 2003]. LDA models document generation as picking some topics for a document and sampling words from each topic. In other words, a document is modeled as a distribution over topics and a topic is modeled as a distribution over words. WSD-TM uses the same idea but replaces the concept of topic with synsets. It models a document as a distribution over synsets and a synset as a distribution over words. Figure 5.1 illustrates the method in the same way LDA is usually illustrated.

The main difference between topics in LDA and synsets in WSD-TM is that synsets correlate. Chaplot and Salakhutdinov [2018] use a logistic normal distribution for distribution over synsets instead of the Dirichlet distribution to address this difference. They will also take the similarity of synsets (measured based on the overlap of their definitions) as well as the frequency of a word in each synset into account when choosing a synset for a word.

5.3.4 KEF

KEF [Wang et al., 2020] is a comprehensive knowledge exploitation framework that makes use of WordNet glosses, examples, and relations, as well as Wikipedia documents to reach the state-of-the-art among knowledge-based WSD methods. Its workflow is shown in Figure 5.2.

KEF uses a TF-IDF-based simple information retrieval component to retrieves Wikipedia documents most similar to the document being disambiguated. It uses the union of retrieved documents and the British National Corpus to calculate vector representations of words using Latent Semantic Analysis(LSA) [Landauer and Dumais, 1997].

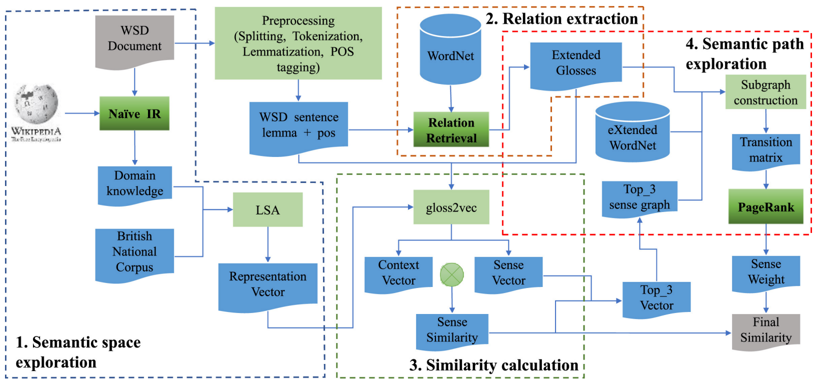


Figure 5.2: Workflow of KEF. Figure taken from [Wang et al., 2020].

It also gets a vector representation for WordNet senses by applying LSA on a bag-of-word representation of glosses and examples of not only the sense itself but all its hypernyms in WordNet. Similarities of the context vector and the sense vectors are calculated to rank the candidate senses for each word.

Finally, to take into account the effects of senses on each other, it forms a graph over all candidate senses based on WordNet connections and sense similarities it calculates based on the vector representations of each sense, and use a random walk to assign weights to each candidate sense.

5.3.5 Bi-LSTMatt+LEX

As illustrated in Figure 5.3, Raganato et al. [2017a] used BiLSTM with attention in a multitask setting to tackle the WSD task. WSD, POS prediction, and Coarse-grained semantic labels (LEX) prediction were the three tasks in their multi-task system. Lex prediction is the task of predicting one of 45 coarse-grained semantic categories that, as part of the WordNet, were manually associated with all WordNet synsets on the basis of both syntactic and logical groupings. These labels are saved in WordNet lexicographer files and can be found in WordNet documentation[1].

5.3.6 supWSDemb

Papandrea et al. [2017] released an extended and re-engineered implementation of IMS that was easy to extend and was designed for large datasets. Like IMS, supWSDemb uses an SVM classifier over surrounding words and their POS tags among other features. It also includes embeddings and features based on dependency parses in the publicly available implementation while making it easier to add user-defined features into the mix as well.

Papandrea et al. [2017] achieved the state-of-the-art by training over both Semcor and OMSTI but showed improvements over IMS$_{-s}$+emb even when Semcor was the sole training set.

[1]https://wordnet.princeton.edu/documentation/lexnames5wn

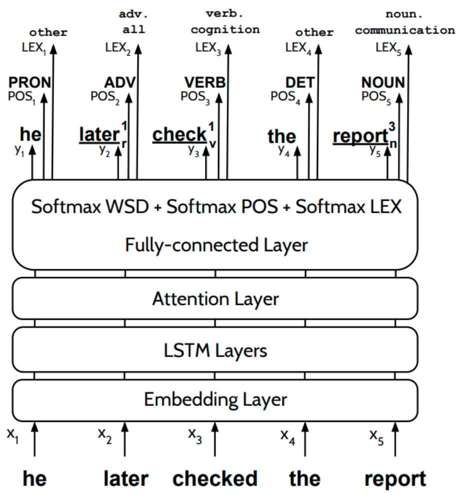


Figure 5.3: Architecture of Bi-LSTMatt+LEX. Figure taken from Raganato et al. [2017a].

5.3.7 GASext

[Luo et al., 2018] proposed a rather complex system that compares vector representations of context with vector representation of WordNet glosses to find the most suitable gloss. Their whole model is illustrated in Figure 5.4.

They used Bi-LSTMs to encode the context and the sense glosses and introduced a gloss extension system to take glosses of hypernyms and hyponyms into account as well. Then they used the attention mechanism to go over all gloss representations multiple times and maintain a memory module. In the first round, the attention will pick on the similarity of the context and each candidate gloss representation. In later rounds, the attention will pick on the similarity of memory and the

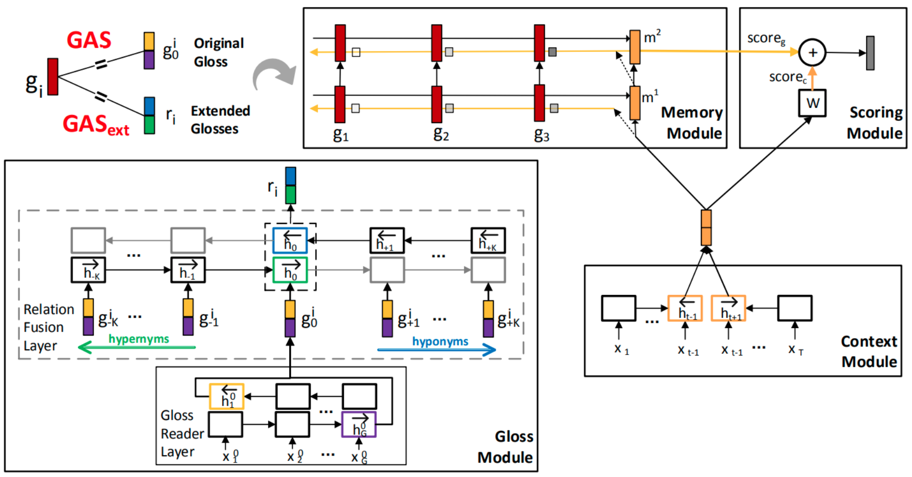


Figure 5.4: Architecture of GASext. Figure taken from [Luo et al., 2018].

gloss representations. The memory can be updated linearly or with concatenation. A scoring module finally ranks the candidate glosses according to the computed similarities in the memory module and picks the most similar gloss.

[Luo et al., 2018] reported better performance on the benchmark datasets over the previous methods in all their configurations but observed that the best result is obtained in presence of gloss extension and with memory updates through concatenation.

5.3.8 BERT

Like ELMO, BERT [Devlin et al., 2019] is a pre-trained contextualized word embedding trained on billions of words of text with the task of predicting a word given its target. Since the neural network architecture BERT uses is a transformer [Vaswani et al., 2017], which is not recurrent and can be trained much more efficiently and in a highly parallelized manner, it was possible to train it on much larger data.

Hadiwinoto et al. [2019] experimented with many strategies to exploit BERT for WSD. Following Peters et al. [2018b] and Melamud et al. [2016], Hadiwinoto et al. [2019] tried running the train and test contexts through BERT to get the context embedding. To predict a sense for a given test context, they found the nearest train context for the given lemma and POS and predicted the gold sense corresponding to the nearest train context as the sense for the test context.

Unlike Peters et al. [2018b] and Melamud et al. [2016] though, Hadiwinoto et al. [2019] observed improvements over previous work that utilized non-contextualized embeddings. They also pushed the idea further by trying other strategies, namely linear projection of hidden layers and using gated linear units(GLU) [Dauphin et al., 2017] and obtained the best results using GLU. They also showed the value of larger contexts by showing that extending the context by one surrounding sentence in each direction (left and right) consistently yields higher results. The BERT F-scores in Table 5.2 were obtained from the variation with GLU and extended context.

5.3.9 GlossBERT

Huang et al. [2019] introduced GlossBERT, a model that like knowledge-based methods and GASext uses the glosses of WordNet senses to match them with a given context. All it does is that it runs BERT not only on the given contexts, but also on the WordNet glosses, fine-tune BERT, and then performs a pair classification. When training, a (context,gloss) pair will have a gold label "Yes" if and only if the sense corresponding to that gloss is the gold sense for the given context. When testing, the probability of the label "Yes" is used to rank the candidate senses and pick the most probable one.

They experimented with three settings. The first two settings differ in what is being fed into the feed-forward classifier. In one setting, they used the representation of the first token in the sentence ([CLS]), and in the other, they used the average of representations of all the tokens in the target word. The third setting is different from the first one in that it uses a signal for where the target

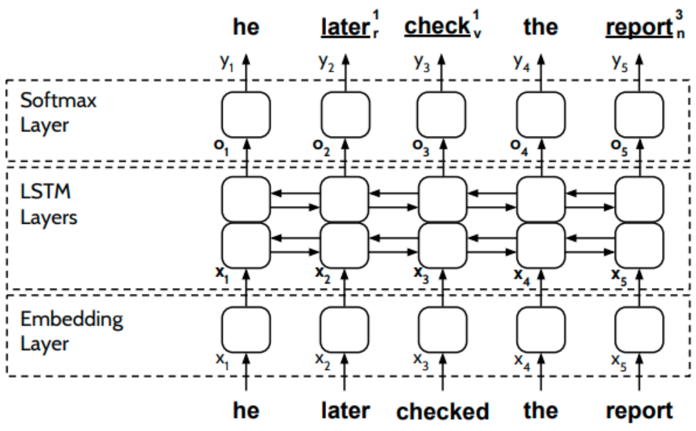


Figure 5.5: The Architecture of Raganato et al. [2017b] (Figure taken from the original paper.

word is as weak supervision. In this setting, the representation of [CLS] is fed to the classifier layer. Huang et al. [2019] observe that using [CLS] with weak supervision yields the best results.

5.3.10 hypernyms+WNGC

Vial et al. [2019] used the hypernymy hierarchy in WordNet synsets to map WordNet senses to higher synsets and train a deep neural network for Word Sense Disambiguation.

The architecture Vial et al. [2019] used was based on that of Raganato et al. [2017b], illustrated by Figure 5.5, except instead of predicting the word itself, in the absence of a label, Vial et al. [2019] always (even for stop words) predict a sense but use a special <skip> token to prevent the objective function and the evaluation script from paying attention to any unlabelled word. For the input embeddings, they experimented with Glove [Pennington et al., 2014], ELMO [Peters et al., 2018b] and BERT [Devlin et al., 2019] embeddings.

The main point in this work is a sense compression method based on the WordNet hypernymy hierarchy illustrated in Figure 5.6.

The sense compression method maps a sense to a clear synset higher up in the hierarchy. A clear synset for a given pair of lemma and POS is a synset that does not contain more than one sense for that pair of lemma and POS in its subtree. For example, we can see in Figure 5.6 that whole#2 is not a clear synset for the (mouse, Noun) because more than two senses (mouse#1 and mouse#4) exist in the subtree rooted at whole#2.

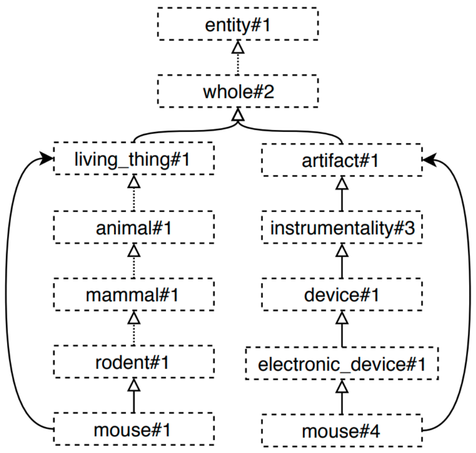


Figure 5.6: WordNet hypernymy hierarchy. Figure from [Vial et al., 2019].

However, there is more to the procedure. Vial et al. [2019] don't map a sense to the highest synset in the

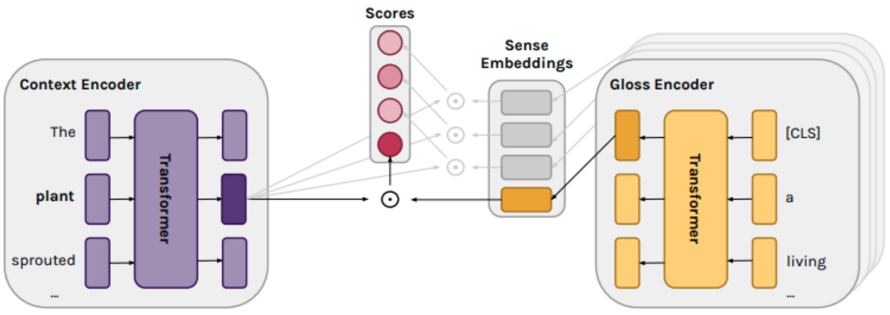


Figure 5.7: Architecture of BEM. Figure taken from [Blevins and Zettlemoyer, 2020].

hierarchy that is clear for that (lemma, POS). They follow a two-step procedure to intuitively maximize the number of senses being mapped to the same synset. Here is their two-step procedure:

1. For any sense, the highest *clear* synset in its hypernymy path is marked as a **Necessary** synset.
2. Each sense is mapped to the shallowest *necessary* synset in its hypernymy path.

Even though Vial et al. [2019] experimented with another sense compression method that clustered senses based on all relations in WordNet and not just hypernymy, they observed this two-step sense compression method yield better results both when they were training on SemCor only and when they were training on the union of SemCor and WordNet Gloss Corpus (WNGC) [Vial et al., 2018] which is a sense-annotated corpus of WordNet sense glosses. Their best results however were obtained when they trained on the union of the two datasets.

5.3.11 BEM

[Blevins and Zettlemoyer, 2020] continued the trend of using WordNet glosses and matching them against a given context. Similar to GlossBERT, they encode both contexts and glosses using BERT and compute scores based on the similarity of these representations. Unlike GlossBERT, they separate the context encoder and gloss encoder and train the gloss encoder independently of the contexts even though the two encoders are trained jointly with the objective that the nearest neighbor of a context in the senses should belong to the gold sense. The separation of the two encoders results in a more efficient model and higher performance as evident in Table 5.2. Figure 5.7 illustrates the architecture of BEM.

5.3.12 EWISER+WNGC

Bevilacqua and Navigli [2020] exploited not only the glosses but the internal structure of WordNet synsets through their structured logit mechanism. The idea is to produce scores on all WordNet synsets and taking the score of related synsets into account when assigning a final score to a particular sense.

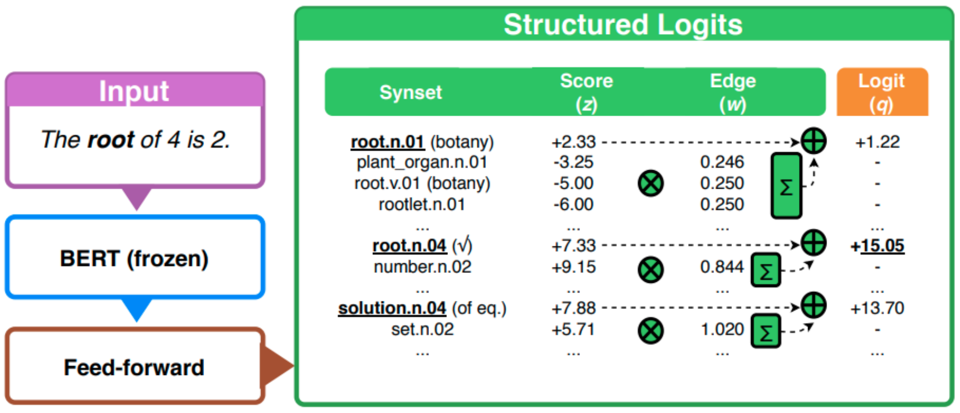


Figure 5.8: EWISER takes scores of other related senses into account when assigning a final score to a particular sense. Figure taken from [Bevilacqua and Navigli, 2020]r.

To achieve this goal Bevilacqua and Navigli [2020] compute an initial score z_s for sense s using a standard BERT+feedforward sense prediction architecture. Then they use the initial scores over all synsets and computer a second score q_s according to the following formula:

$$q_s = z_s + \sum_{s' \in V | <s', s> \in E} w(< s', s >) \cdot z_{s'}$$

where $< V, E, W >$ represent a graph over synsets(V) where related synsets are connected through edges(E) and have weights(W). Figure 5.8 illustrates this structured logit mechanism.

Bevilacqua and Navigli [2020] use WordNet relations to determine the edges and use $1/N$ where N is the number of incoming edges as initial weights in the graph which they fine-tune. As for the WordNet relation types, they experiment with different combinations and observe that considering hypernymy only yields the best results.

5.4 Comments on This Evaluation Approach

There are at least two problems with evaluating over the benchmark datasets of Raganato et al. [2017b]. 1) The differences between methods usually show inconsistency across the five datasets in the benchmark framework. And 2) All annotations are from WordNet senses, making it impossible to fairly evaluate methods that use other sense-inventories or avoid using sense inventories altogether. We elaborate on these problems in the next sections.

5.4.1 Little Consistency Across Benchmark Datasets

Looking back at Table 5.2 we can see several inconsistencies over the benchmark datasets. Here are some examples to name just a few:

- EWISER+WNGC achieves the best results over all datasets except for SE 2015;

System	SE2	SE3	SE07	SE13	SE15	Avg
Simple (1sent)	75.0	71.6	67.0	69.7	74.4	72.7
Simple (1sent+1sur)	75.9	73.4	**69.3**	70.4	75.1	73.7
LW (1sent)	75.0	71.6	66.7	69.9	74.2	72.7
LW (1sent+1sur)	**76.4**	**74.0**	69.0	70.1	75.0	73.9
GLU (1sent)	74.1	71.6	64.9	69.8	74.3	72.5
GLU (1sent+1sur)	75.5	73.6	68.1	**71.1**	**76.2**	**74.1**
GLU+LW (1sent)	74.0	70.9	65.7	68.8	73.6	71.8
GLU+LW (1sent+1sur)	75.5	73.4	68.5	71.0	**76.2**	74.0

Table 5.3: Ablation study from Hadiwinoto et al. [2019]

- Babelfy beats the baselines in SE2, SE2013, and SE2015 but not in SE2007 and SE3 and SE2007;

- BEM beats hypernyms+WNGC in SE2007 and SE2013, but not in the other three datasets;

The degree of inconsistency across the datasets is so high that it makes it difficult to decide which method is truly better unless performance in ALL (the union of all) datasets [Bevilacqua and Navigli, 2020; Luo et al., 2018] or the average of scores [Hadiwinoto et al., 2019] are considered. Not all recent papers have mentioned their performance on ALL datasets [Hadiwinoto et al., 2019; Papandrea et al., 2017]. None of the recent papers offered analysis on why their method was doing better on some of the datasets but not on others.

Not only this makes it difficult to choose the best model, but it also makes it difficult to make a case for certain features in ablation studies. Table 5.3 shows the ablation study from Hadiwinoto et al. [2019]. While it clearly shows the advantage of extending the context by surrounding sentences, it is harder to make the case for GLU to be the best combination method. Similarly, Table 5.4 shows the ablation study from Vial et al. [2019]. While it is evident that adding WNGC is clearly beneficial, it is harder to make the case for hypernyms as the best compression method.

Since Vial et al. [2019] report on SE07-7 as an additional dataset and observe pretty high results regardless of the method, it may need a brief discussion. Looking at the data and its description, it is evident that instances are annotated with many WordNet senses in a grouping attempt. However, since the evaluation script is considering a prediction correct if it equals any of the annotations, the dataset ends up getting high results regardless of the method. In fact, even the first sense baseline yields an F-score of 78.9 on this dataset. In any case, reporting results on this additional dataset

System	SE2	SE3	SE07	SE13	SE15	ALL	SE07-07
SemCor, baseline	77.2	76.5	70.1	74.7	77.4	76.0	87.7
SemCor, hypernyms	77.5	77.4	69.5	76.0	78.3	76.7	87.6
SemCor, all relations	76.6	76.9	69.0	73.8	75.4	75.4	86.7
SemCor+WNGC, baseline	**79.7**	76.1	**74.1**	78.6	80.4	78.3	90.4
SemCor+WNGC, hypernyms	**79.7**	77.8	73.4	**78.7**	**82.6**	**79.0**	90.4
SemCor+WNGC, all relations	79.4	**78.1**	71.4	77.8	81.4	78.5	**90.6**

Table 5.4: Ablation study from Vial et al. [2019]

did not help them motivate their compression methods as hypernyms show no improvement over baseline and the improvement achieved by all relations is not salient.

5.4.2 WordNet is Only One Choice

A sense inventory is at the core of the Word Sense Disambiguation (WSD) task in which the correct sense for a word is predicted given the context. However, the exclusive focus of recent papers on the benchmark datasets of Raganato et al. [2017b], all annotated with WordNet senses, seems to have re-defined the English WSD task as predicting WordNet senses.

Apart from a lack of choice in sense-inventories for low-resource languages, it is not ideal that WordNet fine-grained senses are considered the only sense inventory for WSD for at least the following reasons:

1. Even human annotators struggle with distinguishing between some word senses in WordNet. This was established during the OntoNotes [Hovy et al., 2006] project and most recently verified by Pilehvar and Camacho-Collados [2019] who reported a 57% average accuracy of human annotators on random samples of pruned instances in Word-in-Context(WiC) [Pilehvar and Camacho-Collados, 2019] (described in detail in Section 7.2) with especially confusing sense usages, such as sibling senses in WordNet hierarchy.

2. There is not enough training data for all the fine-grained senses, making training for those senses difficult. In fact, even the WordNet itself does not provide examples for all fine-grained senses.

3. Lack of a WordNet-independent evaluation set will make it impossible for approaches with different sense-inventories to be fairly compared with current methods.

4. Methods that entangle themselves with WordNet beyond using the sense inventory, such as methods that match word contexts with glosses of candidate senses [Huang et al., 2019; Blevins and Zettlemoyer, 2020], may not work well with other sense-inventories even if these other sense-inventories are groupings of WordNet fine-grained senses.

There have been isolated works working with other sense inventories. SemEval 2007 task 7 [Navigli et al., 2007] for example, attempted to address the well-established problem of low inter-annotator agreements for WordNet fine-grained senses by mapping them to coarse senses of Oxford English Dictionary (OED) [Catherine and Angus, 2005], effectively clustering the WordNet senses. To the best of our knowledge, the only recent work reporting results on this dataset though, is the work of Vial et al. [2019] who improved the state of the art F score on this dataset from the previous 83.6 [Yuan et al., 2016] to 90.6. Furthermore, Yuan et al. [2016] experimented with training and evaluation datasets annotated according to the New Oxford American Dictionary (NOAD) [Lindberg and Stevenson, 1999]. Even though they made the annotated dataset available, it was largely ignored by the community. We suspect this is due to the proprietary nature of the NOAD dictionary.

The community does have access to at least one public sense-inventory to use besides WordNet however—OntoNotes. To the best of our knowledge however, no all-word WSD task was annotated according to this alternative repository. The Lexical Sample part in SemEval 2007 task 17 however, was annotated according to OntoNotes.

OntoNotes

OntoNotes [Hovy et al., 2006], is a large dataset annotated for several semantics annotations, including word senses, that was developed with a target 90% inter-annotator agreement in mind. The six layers of annotations in OntoNotes are as follows:

- Parse tree annotations (ongoing Penn Treebank project [Marcus et al., 1993])
- Verb argument annotations (ongoing Propbank project [Kingsbury and Palmer, 2002])
- Word sense annotations
- Links to Omega ontology [Philpot et al., 2005]
- Coreference annotations
- Named Entity annotations

In the word sense annotation layer, Hovy et al. [2006] have regrouped the senses in the WordNet sense inventory according to the algorithm of Figure 5.9 to ensure the 90% inter-annotator agreement threshold.

There are only verb senses and noun senses in OntoNotes. The verb senses were developed as groups of WordNet senses according to subcategorization frames and semantic classes of arguments subject to the workflow of Figure 5.9. The noun senses were developed according to the same strategy starting from WordNet and other dictionaries by the same individual who grouped the WordNet verb senses.

Table 5.5 shows the number of documents, sentences, word-tokens, and sense annotations in the train, dev, and test partitions of the OntoNotes dataset. The partitions are the same partitions introduced in the github repository. It is also worth mentioning that the number of verb senses is 2702 and the number of noun senses is 2194.

Since OntoNotes sense annotations are groupings of WordNet senses, one can re-annotate the training and evaluating datasets according to this alternative sense-inventory and re-evaluate those

partition	#documents	#sentences	#word-tokens	#sense annotations
train	10539 (80%)	115812 (80%)	2200865 (80%)	229989 (79%)
development	1370 (10%)	15680 (11%)	304701 (11%)	31060 (11%)
test	1200 (9%)	12217 (9%)	230118 (8%)	29834 (10%)

Table 5.5: Size of OntoNotes

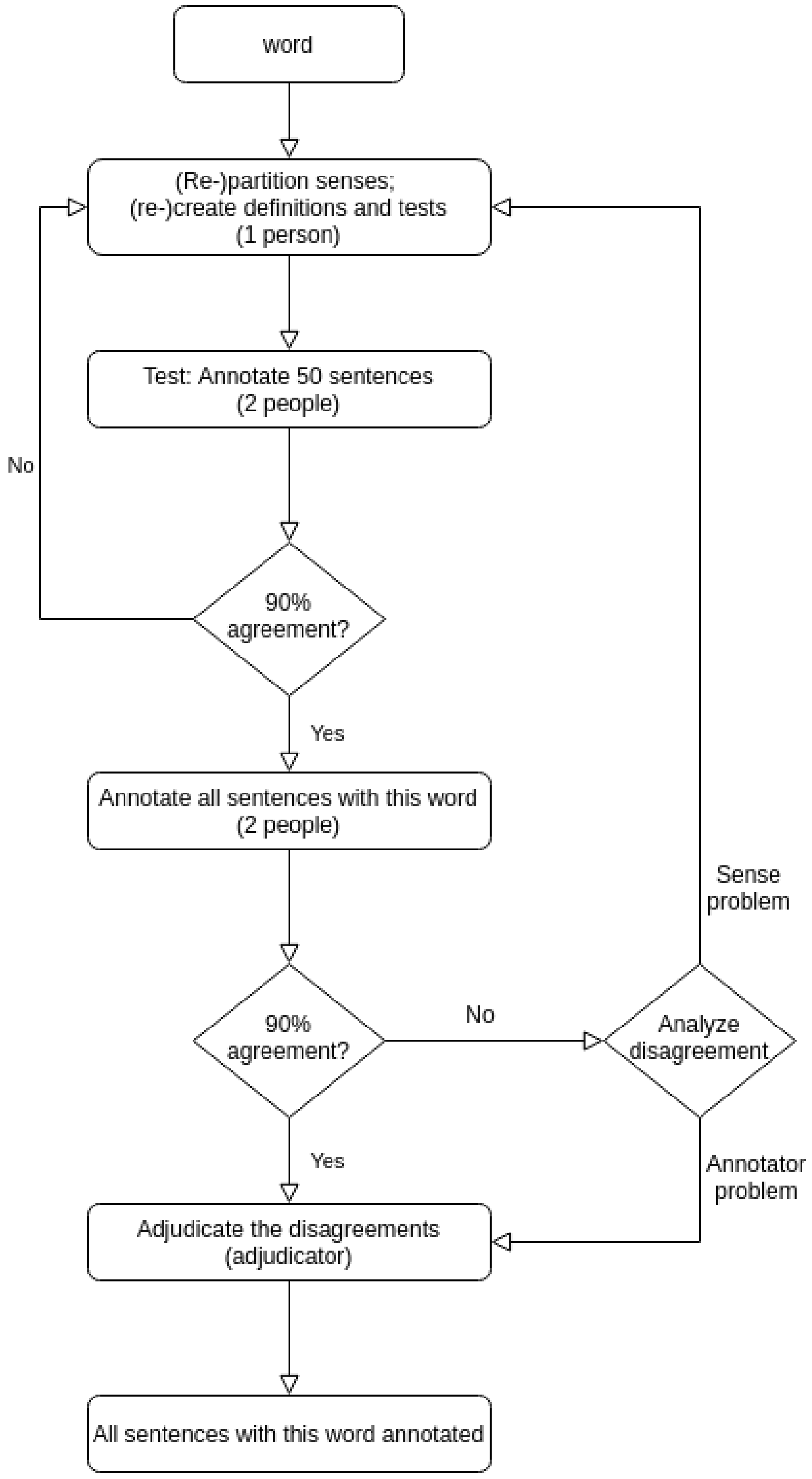

word
(Re-)partition senses;
(re-)create definitions and tests
(1 person)
Test: Annotate 50 sentences
(2 people)
No
90%
agreement?
Yes
Annotate all sentences with this word
(2 people)
Sense
problem
90%
agreement?
No
Analyze
disagreement
Yes
Annotator
problem
Adjudicate the disagreements
(adjudicator)
All sentences with this word annotated

Figure 5.9: OntoNotes algorithm for sense inventory creation

Category	Description	#mappings		
		Noun	Verb	Both
trivial	explicitly mentioned in sense inventory with WN3 version	5673(3.9%)	13581(54.2%)	19254
mapped	explicitly mentioned in sense inventory with other WN versions that we mapped to a WN3 sense	1349(0.9%)	0	1349
trivial+ mono	trivial that is also monosense in WordNet	493(0.3%)	1592(6.4%)	2085
none of above	not trivial, mapped, or mono with existing onotoNotes word-pos pair that includes a last 'none of the above' sense	751(0.5%)	180(0.7%)	931
none of above (made up)	not trivial, mapped, or mono with existing onotoNotes word-pos pair that does not include a last 'none of the above' sense.	131(0.09%)	25(0.1%)	156
none-existing	a polysense (in WordNet) word-pos pair where the word-pos does not exist in the ontoNotes sense inventory	138070(94.3%)	9677(38.6%)	147747

Table 5.6: OntoNotes coverage over WordNet senses in different categories. For mapping senses from previous versions of WordNet to WordNet3, we used wordNet sense map files.

WSD methods that are not too entangled with WordNet. We mapped WordNet senses to OntoNotes according to the categories of Table 5.6 and used the mappings to re-annotate the training and test sets.

Table 5.7 presents the results of training the code of Vial et al. [2019] on the re-annotated Semcor+WNGC and testing on the re-annotated benchmark datasets. Obviously, we needed to turn off the sense compression option as their compression methods work only on WordNet senses.

While the F-scores seem to be higher than the best of Table 5.2, it is important to note this is not an apple to apple comparison. Also, it is important to note that OntoNotes labels do not provide a 100% coverage on WordNet labels. Table 5.8 shows the extent of missed annotations (instances) in the mapping process.

SE2	SE3	SE 2007	SE 2013	SE 2015
85.86	83.17	81.00	78.27	79.89

Table 5.7: Results obtained by the code of Vial et al. [2019] when trained on OntoNotes-reannotated SemCor+WNGC and tested on OntoNotes-reannotated benchmark datasets.

SE2	SE3	SE 2007	SE 2013	SE 2015	SemCor	WNGC
1141(50%)	763(39%)	55(12%)	696(42%)	486(46%)	96027(42%)	440412(72%)

Table 5.8: Number and percentages of missed labels when re-annotating to OntoNotes labels for each training and evaluation dataset

5.5 Summary

In this chapter, we presented the background on WSD. We described the current evaluation framework and summarized the progress of the field on its benchmark datasets. We also commented on aspects of this evaluation approach that are not ideal.

The next chapter will focus on WSI, a related task where the meaning (sense) of a target word in a given context should be discovered as opposed to being picked from a pre-defined sense inventory. Later, in Chapter 7 we provide an evaluation framework that addresses some of the shortcomings of the current evaluation approaches for both WSD and WSI.

Chapter 6

Current State of Sense Induction

Word Sense Induction (WSI) is a task very similar to WSD. Both tasks aim to distinguish different meanings (senses) of a word in different contexts, but WSD does so with regard to a given sense inventory and thus is a classification task, while WSI determines what usages are referring to the same sense and as such is a clustering task. In this chapter, we provide an overview of the state-of-the-art WSI methods, describe the datasets used for WSI evaluation, and comment on less than ideal aspects of this evaluation framework to lay the ground for our proposed evaluation framework in the next chapter.

6.1 State of the Art in WSI

While recent WSD methods consistently report on the benchmark datasets of Raganato et al. [2017b] as described in Section 5.2, recent WSI methods are not so consistent. Table 6.1 shows the results that recent methods report on the most frequently used WSI datasets. It is evident from the empty cells in the table how inconsistently WSI methods report their results.

If it was not for a more recent paper expanding LSDP to BERT+DP [Amrami and Goldberg, 2019] that reports results on both datasets of SemEval-2010 [Manandhar et al., 2010] and SemEval2013-task13 [Jurgens and Klapaftis, 2013], it would have been impossible to pick a state-of-the-art method at all as Word2Sense [Panigrahi et al., 2019] was a previous state-of-the-art on SemEval2020 while LSDP [Amrami and Goldberg, 2018] held its place on SemEval2013.

MakeSense-2016 [Mu et al., 2017] and Task 2 of SemEval-2007 [Agirre and Soroa, 2007] are other WSI dataset. But for each of them, only one of the methods in Table 6.1 has reported results: BNP-HC on SemEval-2007, and Word2Sense on MakeSense-2016.

In spite of the inconsistency, it is evident from Table 6.1 that BERT+DP is the state-of-the-art in WSI. Before we describe this method, we briefly describe the SemEval2010 and SemEval2013 WSI datasets and the performance measures for them.

Method	SemEval2010			SemEval2013		
	F-S	VM	Avg	F-BC	F-NMI	Avg
BERT+DP [Amrami and Goldberg, 2019]	71.3	40.4	53.6	64.0	21.4	37.0
Word2Sense [Panigrahi et al., 2019]	59.38	6.80	33.09	-	-	-
AutoSense [Amplayo et al., 2019]	61.7	9.8	24.59	61.7	7.96	22.16
LSDP [Amrami and Goldberg, 2018]	-	-	-	57.5	11.3	25.4
SE-WSI-fix [Song et al., 2016]	55.1	9.8	23.24	-	-	-
MCC-S [Komninos and Manandhar, 2016]	68.9	67.5	17.1	55.6	7.62	20.58
STM+w2v [Wang et al., 2015]	-	-	-	55.4	7.14	19.89
BNP-HC [Chang et al., 2014]	23.1	21.4	22.23	-	-	-
LDA [Goyal and Hovy, 2014]	60.7	4.4	16.34	-	-	-
AI-KU [Başkaya et al., 2013]	-	-	-	39.0	6.5	15.92

Table 6.1: Recent progress in Word Sense Induction

6.2 SemEval2010 Task 14 for WSI and WSD

Task 14 of SemEval2010 [Manandhar et al., 2010] was a two-step task. In the first step, the systems were to induce senses for 100 target words (50 nouns and 50 verbs) based on unlabeled training instances. In the second step, the systems would tag each testing instance with a sense they had induced in the first step and had them evaluated. Table 6.2 shows statistics on the datasets of this task.

The Training set was curated automatically by sending queries to Yahoo! Search API that followed the pattern <*Target Word*> *AND* <*Relative Set*>. The relative sets were gathered from WordNet from senses that were in hypernymy, hyponymy, synonymy, meronymy, or holonymy with the target word. See Table 6.3 for examples of the search queries used to gather training instances for two senses of the word *failure*. The test set was sampled from OntoNotes [Hovy et al., 2006] dataset labeled with OntoNotes senses.

Task 14 of SemEval2010 was introduced with both supervised and unsupervised evaluation frameworks. In the unsupervised framework, V-measure [Rosenberg and Hirschberg, 2007] and paired F-scores were used to evaluate how the proposed clustering of a system compares to the gold standard clustering. And following the SemEval2007 task, the supervised framework mapped the proposed clusters to gold standard clusters using a relatively large split (80%) of the test set and evaluating on a held-out split (the remaining 20%).

	Training set	Testing set	Average Number of Senses
All	879807	8915	3.79
Nouns	716945	5285	4.46
Verbs	162862	3630	3.12

Table 6.2: Statistics of Training and testing set in SemEval2010 Task 14. The table is taken from [Manandhar et al., 2010].

Word Sense	Query
Sense 1	failure AND (loss OR nonconformity OR test OR surrender OR ”force play” OR ...)
Sense 2	failure AND (ruination OR flop OR bust OR stall OR ruin OR walloping OR ...)

Table 6.3: Example queries to get training instances in SemEval2010 task 14 for the target word *failure*. The table is taken from [Manandhar et al., 2010].

The primary goal of providing both supervised and unsupervised evaluation frameworks is claimed to be "to allow comparison of unsupervised word sense induction and disambiguation systems". The dataset, however, failed to attract WSD systems. We suspect that is due to the fact that the only permitted resources were the unlabeled training instances and NLP components for morphology and syntax when WSD systems by definition use some sense-inventory.

6.3 SemEval2013 Task 13 for WSI and WSD

Task 13 of SemEval2013 [Jurgens and Klapaftis, 2013] is similar to Task 14 of SemEval2010 in that it was introduced for both WSI and WSD methods, providing both supervised and unsupervised evaluation frameworks and yet failing to attract recent WSD methods.

It differs from Task 14 of SemEval2010 in two important aspects. Most importantly, it asks the participants to find not just one best sense for each instance, but all applicable senses weighted by their applicability. Table 6.4 gives examples of when a word is used in a way that more than one sense applies due to intentional or unintentional ambiguity.

Furthermore, Task 13 of SemEval2013 is not a two-step process. It provides a test set that the systems are supposed to tag with senses. Unlike in previous work, ukWaC corpus [Baroni et al., 2009] was provided to the participants to allow sense induction from a large corpus instead of from target-word specific collected sentences.

The test data was sampled from Open American National Corpus [Ide and Suderman, 2004], containing data from both spoken and written language, and was annotated by one of the authors. Inter-annotator agreements were computed by having the second author annotating a portion (10%) of the dataset. Table 6.5 provides statistics on this dataset.

For supervised evaluation, following the previous tasks, the induced senses are mapped to gold labels using distribution mapping technique of Jurgens [2012] and Jaccard Index, Positionally-weighted Kendall’s τ [Kumar and Vassilvitskii, 2010], and weighted normalized discounted cumulative gain (weighted NDCG based on DCG [Moffat and Zobel, 2008]) were used to respectively measure how well the systems detect applicable senses, rank them according to the applicability and their agreement with human annotators in applicability ratings.

For unsupervised evaluation, they introduce Fuzzy B-Cubed (F-BC) and Fuzzy Normalized Mutual Information (F-NMI) based on B-Cubed [Bagga and Baldwin, 1998] and previous use of Mutual Information for clustering evaluation [Danon et al., 2005]. The need for fuzzy measures comes from the fact that multiple senses(clusters) are considered applicable for each instance, making the cluster membership a fuzzy notion. Table 6.1 reports F-BC and F-NMI as those are the measures most frequently used by recent WSI methods on this dataset.

6.4 BERT+DP

BERT+DP [Amrami and Goldberg, 2019], the state-of-the-art in WSI systems, is based on LSDP [Amrami and Goldberg, 2018], which in turn is inspired by AI-KU [Başkaya et al., 2013]. All these methods use substitutes for the word they are trying to make sense of, from language models. AI-KU samples the substitutes from a 4-gram language model, LSDP samples them from pre-trained ELMO, and BERT+DP from pre-trained BERT. LSDP and BERT+DP also make use of dynamic patterns, a creative way to query the language models.

LSDP uses conjunctive symmetric patterns to make the language model take into account not only the context but also the word itself. That is, in order to find substitutes for "oranges" in "the doctor recommends **oranges** for your health", LSDP looks at the probabilities of the next word in both directions of ELMO based on the following two masked contexts: "[blank] and oranges for your health . " and "the doctor recommends oranges and [blank]".

BERT+DP however, found conjunctive patterns to be of little help but parenthetical pattern effective. Following the same example, BERT is given "the doctor recommends oranges (or even [MASK]) for your health ." to find substitutes for oranges.

Both LSDP and BERT+DP sample several substitutes for the target word in each instance to make several multi-hot feature vectors and then cluster them. Each cluster then is treated as a sense. While LSDP uses a fixed number of clusters, BERT+DP starts with a fixed number of clusters and then merges clusters with fewer than a minimum number of examples, assuming they are weak clusters. This approach may make it impossible to induce rare senses as they would likely end up being merged into other ones.

All other methods mentioned in Table 6.1 are based on increasingly complicated graphical models with the exception of SE-WSI-fix which learns sense embedding centroids from the corpus and

We all are relieved to lay aside our fight-or-flight reflexes and to commemorate our births from out of the **dark** centers of the women, to feel the complexity of our love and frustration with each other, to stretch our cognition to encompass the thoughts of every entity we know.
dark%3:00:01:: – devoid of or deficient in light or brightness; shadowed or black dark%3:00:00:: – secret
I **ask** because my practice has always been to allow about five minutes grace, then remove it.
ask%2:32:02:: – direct or put; seek an answer to ask%2:32:04:: – address a question to and expect an answer from

Table 6.4: Example instances with multiple senses due to intended double meanings (top) or contextual ambiguity (bottom). Senses are specified using their WordNet 3.1 sense keys. The table is taken from [Jurgens and Klapaftis, 2013].

	Genre	Instances	Tokens	Avg #senses/inst.
spoken	Face-to-face	52	1742	1.17
	Telephone	699	30,700	1.08
Written	Fiction	127	3438	1.15
	Journal	2403	69,479	1.13
	Letters	103	2238	1.31
	Non-fiction	477	11,780	1.10
	Technical	611	17,337	1.11
	Travel Guides	192	4490	1.11
	All	4664	141,204	1.12

Table 6.5: Statistics of the dataset in SemEval2013 Task 13. The table is taken from [Jurgens and Klapaftis, 2013].

assigns a sense to an unseen instance at test time according to the similarity of the context to the sense centroids.

6.5 Comments on This Evaluation Approach

To the best of our knowledge, no WSD task has been reporting results on any of the two datasets in this chapter. This can be due to the fact that the tasks discourage using other resources and WSD methods sometimes heavily rely on training data and many of them on WordNet glosses or structure.

WSI methods also frequently report results on only one of the two datasets which makes it impossible to readily compare them. Also, they frequently fail to report results according to the supervised evaluation metrics. This means WSD methods reporting those factors will not be able to readily compare with WSI methods, yet another reason for WSD methods to ignore these datasets. It may be worth noting that even if some WSI and WSD methods report results on the supervised evaluation metrics, since the WSI methods have to go through a cluster-to-label mapping step before the evaluation and errors in this step may cause errors in the final results, the comparison may not be completely fair.

6.6 Summary

In this chapter, we summarized the advances of WSI methods on two evaluation frameworks, described the evaluation frameworks and the state-of-the-art methods. We also provided comments on why, even though the evaluation frameworks claimed to aim for both WSI and WSD methods, they failed to attract many WSD methods and that they are not ideal for comparing WSD and WSI methods. In the next chapter, we provide an alternative evaluation approach for WSD and WSI methods that we argue can compare both WSD and WSI methods more fairly.

Chapter 7

Sense Inventory Independent Word Sense Evaluation

As mentioned in Section 5.4, the dependence of the benchmark evaluation datasets for WSD on WordNet makes it impossible to consider and compare alternative sense inventories such as OntoNotes or WSI methods. In this chapter, we propose using sense-inventory independent datasets such as WiC (Word-in-Context) to address this issue. Section 7.2 describe this dataset and presents results of select WSD/WSI methods on it.

The fact that WiC is sense inventory independent makes it a good candidate dataset to evaluate alternative sense inventories or WSI methods as well as WSD methods using WordNet sense inventory. The fact that it is closely related to WordNet, as it was automatically curated from examples of WordNet senses among other things, on the other hand, hurts its credibility in fairly evaluating methods that use WordNet in one way or another.

We address the problem by introducing more sense inventory independent datasets for WSD/WSI evaluation, Namely MASC-NOAD, LexSub, and Trofi, that have little to no meaningful ties to WordNet. The overlaps of all sense-inventory independent datasets with WordNet examples are discussed in Section 7.6. Sections 7.3, 7.4, and 7.5 fully describe and present results of select WSD/WSI methods on our datasets. Before we delve into datasets and results though, we describe the methods we will be evaluating in Section 7.1.

7.1 Select WSD/WSI Methods

To show how sense inventory independent datasets can be useful in WSD/WSI evaluation, we report the accuracy of several WSD/WSI methods on them.

The first two methods are our baselines, RandomWN3Sense, and MFS/FS. RandomWN3Sense takes a random WordNet sense for the target word in each instance. The fewer WordNet senses a (lemma, POS) pair has, the higher the RandomWN3Sense of predicting "T". For monosemous instances, for example, RandomWN3Sense will always predict "T". The other baseline, MFS/FS, always picks the same sense for a given (lemma, POS) and therefore always predicts "T".

We evaluate both Vial+hyper and Vial-hyper to see if the new evaluation framework can provide evidence on the efficacy of the sense compression method of Vial et al. [2019] that maps WordNet senses to certain hypernyms. We also evaluate BEM and GlossBERT, two WSD methods basing their classification based on the similarity of context to WordNet glosses, and compare them with Vial+hyper. It is important to note that, even though BEM and GlossBERT do not train on WNGC, the fact that they use gloss texts in their models, makes this a fair comparison. Finally, we evaluate EWISER, the state of the art in WSD according to the benchmark datasets of Raganato et al. [2017b] to see if it holds its status when evaluated on our framework.

Since WiC datasets are sense inventory independent, they can be used to evaluate any sense inventory. We used the code of Vial et al. [2019], turned off the sense compression option as it is WordNet specific, and trained the model on SemCor+WNGC again except they were relabelled to OntoNotes labels. Since our analysis showed that many errors made with OntoNotes labels were due to lack of coverage, we also trained another model with the same settings except that in the absence of an OntoNotes mapping, we backed off to the original WordNet sense. We report the results of both methods on all sense inventory independent datasets in this chapter.

Finally, sense inventory independent datasets can be used to evaluate WSI methods. We also report results on BERT+DP, the state of the art in the WSI field according to the traditional WSI evaluation datasets, to see how it compares with WSD methods. As a WSI method, BERT+DP has not been previously trained on a large dataset like WSD methods were. Instead, it takes instances of usages of a target word and clusters them. If two instances fall into the same cluster, BERT+DP prediction will be considered "T". Otherwise, the prediction is considered "F". We do not enforce the restriction BERT+DP uses to determine weak clusters and merge them into other clusters. BERT+DP considers a cluster weak if it has fewer than 2 instances. Merging such clusters into others may reduce the method's ability to recognize rare senses. In order to give BERT+DP a chance at competing with pre-trained WSD methods, instead of listing only instances from each evaluation dataset, we list instances from all evaluation datasets as well as SemCor and OMSTI [Taghipour and Ng, 2015a].

We have not included all the methods mentioned in Chapter 5 and 6 for such an exhaustive evaluation goes beyond the scope of this work. We hope to see the authors of those methods, especially the more recent ones, take the time to run their codes on our datasets and report the results.

7.2 WiC:The Word-in-Context Dataset

WiC [Pilehvar and Camacho-Collados, 2019] is a dataset for a binary classification task. It is made of pairs of sentences sharing a target word where the task is to determine whether or not the shared target word has the same meaning in both sentences. Table 7.1 shows some positive and negative samples from the dataset. WiC has 7,466 instances of nouns and verbs in context and is split into pre-determined train, dev, and test sets.

Sentence 1	Sentence 2	Gold Label
There's a lot of trash on the *bed* of the river	I keep a glass of water next to my *bed* when I sleep	F
Justify the margins	The end *justifies* the means	F
Air pollution	Open a window and let in some *air*	T
The expanded *window* will give us time to catch the thieves	You have a two-hour *window* of clear weather to finish working on the lawn	T

Table 7.1: Sample positive (T) and negative (F) pairs from the WiC dataset (target word in italics). Samples taken from [Pilehvar and Camacho-Collados, 2019].

Even though this task was introduced as an evaluation benchmark for contextualized embeddings, it can also be used to evaluate WSD techniques. And it is especially interesting as it does not limit the experiments to a specific sense inventory.

WiC was curated automatically by extracting example sentences for senses from WordNet, VerbNet [Schuler, 2006], and Wiktionary. In order to keep the dataset balanced and diverse, Pilehvar and Camacho-Collados [2019] decided to keep a maximum of three instances for each target word and to prevent repeated contextual sentences across the instances. They also pruned the dataset to remove pairs where the senses of the shared target word were not the same but were too similar in WordNet. Sibling senses and also senses belonging to the same SuperSense were considered too similar.

In order to estimate an upper bound on the dataset, they prepare four randomly chosen 100-instance subsets of the dataset for four expert annotators making sure that two of the annotators share 50 of the instances. They found the average accuracy of the four experts to be 80.0% (individual scores of 79%, 79%, 80%, and 82%). They also report an 80% inter-annotator agreement for the 50 instances shared by two annotators. Furthermore, they observed that having expert annotators annotate 100 random pruned instances, led to an average of 57% accuracy showing the importance of the pruning step.

7.2.1 WiC for Evaluating WSD/WSI

Table 7.2 shows how different WSD/WSI methods compare on WiC datasets. Since none of the methods saw instances of WiC-train or WiC-dev during training, these datasets are as suitable as WiC-test for their evaluation.

It is evident from Table 7.2 that

1. the compression method of Vial et al. [2019] results in consistent salient improvements;
2. the method of Vial et al. [2019] surpasses BEM, GlossBERT, and EWISER on all WiC datasets;
3. When lack of coverage in OntoNotes is addressed by backing off to WN3, Vial + OntoNotes surpasses most other methods and ties with vial+hyper as they both yield the same result on WiC-dev, and each win over the other in one of the two remaining WiC datasets;

Method	Trainset	WiC-train	WiC-dev	WiC-test
RandomWN3Sense	-	54.64	60.34	56.43
MFS/FS (All True)	-	50	50	50
Vial +hyper	Semcor+WNGC	71.67	**70.69**	**71.64**
Vial -hyper	Semcor+WNGC	65.24	65.99	61.5
Vial + OntoNotes	Semcor+WNGC	69.27	61.76	60.79
Vial + OntoNotes + WN3	Semcor+WNGC	**73.78**	**70.69**	70.29
BEM	Semcor	68.9	69.12	69.14
GlossBERT	Semcor	67.39	67.55	65.93
EWISER	Semcor+WNGC	68.79	66.14	65.36
BERT+DP	-	57.77($\pm$0.53)	58.97($\pm$1.83)	59.86*

Table 7.2: Accuracies of select WSD/WSI methods on WiC datasets. Note that all WiC datasets (including WiC-train and WiC-dev) were as unseen as WiC-test to all methods.

4. BERT+DP is not competitive with the other methods on WiC datasets.

While the first two conclusions are in agreement with previous results on benchmark WSD evaluation datasets, performances on WiC dataset strengthen the evidence that the compression method of Vial et al. [2019] is indeed effective. The other conclusions, the efficacy of OntoNotes sense-inventory, and that the best WSI method is not as good as some of the bests WSD methods were not possible to draw based on performances on the benchmark WSD evaluation datasets.

7.3 MASC-NOAD

Ide et al. [2008] introduced Manually Annotated Sub-Corpus (MASC), a collection of annotated texts from diverse genres that includes manual WordNet sense annotations among other semantic annotations.

Yuan et al. [2016] re-annotated MASC with labels from New Oxford American Dictionary (NOAD)[1] and made the annotations public[2]. However, the annotations did not receive much attention. We suspect that is due to the fact that NOAD is proprietary and that poses a serious problem for certain methods. For example, BEM and GlossBERT can not work with NOAD annotations as the glosses are not publicly available.

We used the NOAD annotations provided by Yuan et al. [2016] to assign a same-sense label to sentence pairs sharing a (lemma, POS) pair. Even though NOAD is not publicly available, and thus labels were not interpretable, it was still possible to compare NOAD labels and assign a "T" for pairs with the same label and an "F" for the ones with different labels. We constructed a balanced (1500 positive and 1499 negative instances) WiC-style dataset that we call MASC-NOAD from now on.

[1]Available at `https://www.oxfordreference.com/`

[2]Re-annotated MASC can be found at `https://research.google/tools/datasets/`

Method	Trainset	MASC-NOAD
RandomWN3Sense	-	54.52
MFS/FS (All True)	-	50.02
Vial +hyper	Semcor+WNGC	**70.29**
Vial -hyper	Semcor+WNGC	64.25
Vial + OntoNotes	Semcor+WNGC	62.42
Vial + OntoNotes + WN3	Semcor+WNGC	68.02
BEM	Semcor	69.12
GlossBERT	Semcor	68.22
EWISER	Semcor+WNGC	67.76
BERT+DP	-	66.31($\pm$0.50)

Table 7.3: Accuracies of select WSD/WSI methods on MASC-NOAD dataset

7.3.1 MASC-NOAD for Evaluating WSD/WSI

Table 7.3 shows the results of our select WSD/WSI methods on this dataset. The same conclusions we drew based on results on WiC datasets still hold:

1. the use of hypernyms in compressing WordNet senses proves useful yielding salient improvements;
2. Vial+hyper achieves the best results beating BEM, GlossBERT, and EWISER;
3. backing off to WordNet senses addresses the lack of coverage issue in using OntoNotes sense inventory;
4. BERT+DP is still behind the WSD methods, but with a smaller margin.

The only notable difference is that using OntoNotes sense inventory does not yield better results than BEM and GlossBERT on MASC-NOAD.

7.4 LexSub

Task 10 in SemEval-2007 [McCarthy and Navigli, 2007] was an English Lexical Substitution Task. The task is to give lexical substitutes for a target word in a given context. The Data is split into trial and test. Each target word appears in about 10 contexts and is annotated with a set of substitutes in each context.

While this is not a WSD task itself, it is a related task. If a target word in two different contexts can be substituted with the same set of words, then we argue, that means the word was used in the same sense. In other words, we can use the gold substitute sets as proxies for word sense. Table 7.4 shows the gold substitute sets for several instances of the word "bug" as a noun. All these samples are actual instances in the dataset of SemEval-2007-Task 10.

ID	Word in context	Gold substitutes
571	In any case , it deems the US President to have the power to authorise **bugs** on any person and any organisation	{listening device; tap; surveillance; spying device}
576	Let your child pick one **bug** to glue on the lid.	{insect}
577	Later she made **bug** soup in the birdbath.	{insect}
579	We wanted to give our client more than just a list of **bugs** and an invoice– we wanted to provide an audit trail of our work along with meaningful productivity metrics.	{error; defect; virus; code defect; fault}

Table 7.4: Samples from Task 10 in SemEval-2007 [McCarthy and Navigli, 2007] for the target (lemma, POS) of bug.n

We pair the contexts for each target (word, POS); compute the overlap of their gold substitute sets; and derive a WiC-style dataset from this data by considering a context pair positive if the Jaccard similarity of the corresponding gold substitute sets is more than 0.9 and negative if the similarity is below 0.1. We verified the appropriateness of these thresholds by taking a look at positive and negative pairs. We will call this dataset LexSub from now on. Of all the pairs we can get from the data in Table 7.4, all pairs except for one (576, 577) will be negative instances in LexSub whereas (576, 577) will be a positive instance of LexSub.

LexSub ends up having 229 positive instances as opposed to 3125 negative instances. We derive a LexSub_balanced dataset from LexSub by keeping all the 229 positive instances and randomly sampling the same number of negative instances.

7.4.1 LexSub for Evaluating WSD/WSI

Table 7.5 shows the results on both LexSub and LexSub_balanced datasets. Since LexSub consists of mostly negative instances, RandomWN3Sense does exceptionally well and MFS/FS do extremely bad. In the balanced dataset however, both baselines give results comparable to WiC and MASC-NOAD.

Method	Trainset	All	Balanced
RandomWN3Sense	-	**82.37**	56.7
MFS/FS (All True)	-	7.34	50
Vial +hyper	Semcor+WNGC	75.40	76.29
Vial -hyper	Semcor+WNGC	66.85	75.26
Vial + OntoNotes	Semcor+WNGC	45.26	67.01
Vial + OntoNotes + WN3	Semcor+WNGC	75.05	74.74
BEM	Semcor	75.05	**77.49**
GlossBERT	Semcor	70.18	69.42
EWISER	Semcor+WNGC	76.61	71.82
BERT+DP	-	69.03(±1.36)	72.18(±1.34)

Table 7.5: Accuracies of select WSD/WSI methods on LexSub datasets

As for the performances of WSD/WSI methods, we observe the following:

1. for both LexSub datasets Vial+hyper outperforms Vial-hyper;
2. backing off to WordNet senses helps when OntoNotes sense inventory is used;
3. In LexSub, Vial+hyper achieves the best performance but in LexSub_balanced BEM beats vial+hyper;
4. In neither of the datasets OntoNotes sense inventory (with backoff) falls too far behind;
5. BERT+DP is not performing as well as most WSD methods but is performing better than GlossBERT when evaluated on the balanced dataset.

7.5 Trofi

Birke and Sarkar [2006, 2007] have provided an example base for literal vs. non-literal use of 50 English verbs. Like Lexical Substitution, non-literal usage detection is not the same task as WSD. However, this data can be used to make an all-negative evaluation set for WSD methods. If a (lemma, POS) is used literally in one context and non-literally in another, the context pair can give us a negative WiC-style instance.

Birke and Sarkar [2006, 2007] take a clustering approach based on context similarity of target words to a set of seed literal and non-literal usage examples. They have human annotations for two sets of examples: 1) test cases, and 2) the cases sent to human annotators by an active learning component in their system. We took the human-annotated data and made the following three datasets (Trofi datasets henceforth).

1. All-Negative: 1468 samples of context pairs with one literal and one non-literal usage.
2. Balanced-small: The balanced version of All-Negative where 1400 positive examples were taken from verbs with either only one WordNet sense or two WordNet senses (one clearly literal and one clearly non-literal). Table 7.6 shows the fifty verbs in Trofi datasets along with the number of their WordNet senses.
3. Balanced-All: A balanced dataset of 1459 negative and 1446 positive examples from all examples pretending that same literality equals same sense. That is we pretend that two literal usages or two non-literal usages of a word are the same sense usages.

7.5.1 Trofi for Evaluating WSD/WSI

Table 7.7 shows the results on all three Trofi datasets. We can see that most observations we had for previous datasets do not hold for Trofi datasets. Balanced-All which used the same literality as

absorb(9)	assault(3)	attack(6)	besiege(3)	cool(3)	dance(3)	destroy(4)	die(11)	dissolve(11)	
drag(11)	drink(5)	drown(6)	eat(6)	escape(7)	evaporate(4)	examine(5)	fill(9)	fix(12)	flood(4)
flourish(3)	flow(7)	fly(14)	grab(6)	grasp(2)	kick(8)	kill(15)	knock(6)	lend(3)	melt(6)	miss(9)
pass(25)	plant(6)	play(35)	plow(3)	pour(6)	pump(8)	rain(1)	rest(11)	ride(14)	roll(18)	sleep(2)
smooth(3)	step(10)	stick(16)	strike(22)	stumble(4)	target(1)	touch(15)	vaporize(4)	wither(2)	

Table 7.6: The fifty verbs in Trofi datasets along with the number of their WordNet senses

a proxy for the same sense gets consistently low results. RandomWN3Sense is the best method not only for All-Negative but also for Balanced-small.

If we consider All-Negative only, two of our previous conclusions still hold. Vial+hyper outperforms Vial-hyper, and backing off to WordNet improves a system making use of OntoNotes sense inventory. Unlike before, BEM and GlossBERT surpass Vial+hyper, and BERT+DP works better than all WSD methods but GlossBERT.

Among the irregularities, one makes perfect sense. It is reasonable that RandomWN3Sense is doing so well for not only All-Negative but also Balanced-small. RandomWN3Sense predicts negative more often than positive assuming the number of WordNet senses is more than two. For the positive cases of Balanced-small though, RandomWN3Sense will always predict positive for verbs with only one WordNet sense and predict positive half of the time for the rest as they have exactly two WordNet senses. This explains its high performance on both datasets and shows that our efforts to bring it down by balancing the dataset have failed.

To investigate the reason for the low accuracies on the All-Negative dataset, we took a look at the pairs where all models agree. Table7.8 shows a sample of these pairs with the prediction all models agreed upon. Rows 1 to 6 provide examples where the prediction matches the gold label. Looking up the senses of each target word, we verified that for all these context pairs, each usage is indeed a clear instance of a different WordNet sense. Rows 7 to 12 provide examples where the prediction is different from the gold label. While these are only examples, they do provide some insight into at least a subset of the causes for error:

model	Trainset	Balanced-All	Balanced-small	All-Negative
RandomWN3Sense	-	49.6	**86.09**	**79.97**
MFS/FS (All True)	-	49.78	48.81	0
Vial +hyper	Semcor+WNGC	53.22	67.54	53.41
Vial -hyper	Semcor+WNGC	55.11	62.56	47.42
Vial + OntoNotes	Semcor+WNGC	54.01	46.44	34.76
Vial + OntoNotes + WN3	Semcor+WNGC	52.32	30.75	44.46
BEM	Semcor	53.76	75.84	56.40
GlossBERT	Semcor	**58.21**	78.70	61.10
EWISER	Semcor+WNGC	54.56	75.42	56.47
BERT+DP	-	58.40(±1.16)	45.56(±1.24)	58.84(±1.79)

Table 7.7: Accuracies of select WSD/WSI methods on Trofi datasets

	Context 1	Context 2	predictions
1	They won't let her use her own stove or refrigerator , so she cooks and **eats** meals at a neighbor's.	The chemicals , widely used in refrigerants and styrofoam, are suspected of **eating** away the Earth's protective ozone layer.	Different
2	The mold is then **cooled** and dried and the lid peeled off.	But after several months of talks with each other and with clients, their ardor has **cooled** considerably.	Different
3	The ship hadn't **attacked** U.S.-flag vessels or been involved in mine-laying.	It was one of the congressional spending programs President Reagan **attacked** in his State of the Union address.	Different
4	The teams can get as close to the carcasses as they want to , but can't **touch**.	The thaw in U.S.-Soviet relations has **touched** many areas of Soviet life.	Different
5	Actigall could be used to **dissolve** the fragments that remain from that procedure.	If not, Mr. Mitterrand would **dissolve** parliament and call new elections.	Different
6	Sometimes it **struck** without warning.	It **strikes** me as close to impossible to read Chaucer and not come away liking the poet.	Different
7	They began from scratch, amateurs learning to sing and **dance** and act, working up to 18 hours a day, almost every day.	You 're not going to be able to **dance** and diddle here .	Same
8	That is , they taught me how to get up in the morning when I wanted to **sleep** another few hours.	He takes a sip of brandy and adds an afterthought: "America should go to **sleep** for 50 years so we can catch up. "	Same
9	You can't **eat** these confections , but paper making satisfies the senses all the same.	They will **eat** crow for what they have been saying.	Same
10	They included fixtures required by the company and the salaries and expenses of four assemblers who **flew** in from Italy to set up the store.	Mexico verifies crop destruction by **flying** over sprayed sites; it calls on-ground inspections too risky.	Same
11	Pitching a baseball, it was agreed, is akin to **striking** a golf ball in that it is a complex act that only looks simple.	Mr. Bush avoided making any serious gaffes, while **striking** at his opponent in spirited fashion .	Same
12	The deity is Col. Lee Kuang-chien , the most senior KMT officer to **die** in the 1949 battle.	Thus , while government officials concede apartheid must **die**, they can't bring themselves to kill it.	Same

Table 7.8: Examples of Trofi-AllNegative where all models agreed on a prediction.

1. The usage of *dance* in Context 2 of row 7 and the usage of *sleep* in Context 2 of row 8 are both metaphoric, while *eat* in Context 2 of row 9 is part of an idiom. Whether or not idioms and metaphoric uses should be considered separate senses is not entirely clear.

2. The usage of *fly* in Context 2 of row 10 seems to be literal, which would indicate there could be errors in manual annotations of the original Trofi example base. It seems that in this context-pair, the predicted label is actually correct, and the gold label inherited the error from the original Trofi example base.

3. Looking up all the senses of the words *strike* and *die* in WordNet, it seems the usages of these words in Context2 of rows 11 and 12 are clear instances of different WordNet senses, indicating that these errors are clearly due to mistakes of the WSD/WSI methods.

7.6 Overlaps with WordNet Examples

Our motivation behind making other WiC-style datasets was to distance further from WordNet. WiC dataset was made automatically by taking examples from WordNet, VerbNet, and Wiktionary. The first part of Table 7.9 shows the huge overlap all partitions of WiC have with WordNet examples. This means any method using WordNet examples can not be fairly evaluated on WiC datasets.

In contrast, the datasets we provided have almost no overlap with WordNet examples. The only overlap was the sentence "Time flies like an arrow" that appears in MASC-NOAD.

To compute the overlaps, we retrieved all WordNet examples for all (lemma, POS) pairs in WordNet, retrieved the unique sentences in each of the datasets, and measured both character-level and token-level similarities between any dataset-specific unique sentence and all WordNet examples corresponding to its (lemma, POS) pair. We defined the similarity of two sequences (of characters or tokens) as:

$$sim(x, y) = \frac{2|c|}{|x| + |y|}, \tag{7.1}$$

where x and y are the two sequences, c is the longest common substring of the two, and $|s|$ denotes the length of the string s. We considered two sequences with similarity over 0.9 a match, and a similarity below 0.1 a mismatch. We verified a sample of matches and mismatches and also manually checked the cases where the lengths of the two sequences were smaller than three to make sure we don't miss a match because of a small difference such as missing a punctuation character.

The second part of Table 7.9 illustrates how distinct from WordNet examples our datasets are.

7.7 Putting It All Together

We propose using the datasets mentioned in this chapter as a sense-inventory independent evaluation framework to evaluate WSD and WSI systems, as well as the efficacy of different sense inventories. Since LexSub is unbalanced, Trofi-balanced failed to get a reasonably low response

Dataset	Number o f Unique Sentences	Overlap with WN3 examples
WiC train	6341	6075 (95.81%)
WiC dev	1276	1024(80.25%)
WiC test	2800	2278 (81.36%)
MASC-NOAD	5364	1 (0.02%)
LexSub	1697	0
LexSub Balanced	732	0
Trofi AllNegative	1688	0
Trofi Balanced	1816	0
Trofi AllWords	2695	0

Table 7.9: Overlap between unique sentences in each dataset and WordNet examples

Model	WiC			MASC-NOAD	LexSub balanced	Trofi All-Neg
	train	dev	test			
Vial +hyper	71.67	**70.69**	**71.64**	**70.29**	76.29	53.41
Vial + OntoNotes + WN3	**73.78**	**70.69**	70.29	68.02	74.74	44.46
BEM	68.9	69.12	69.14	69.12	**77.49**	56.40
GlossBERT	67.39	67.55	65.93	68.22	69.42	**61.10**
EWISER+WNGC	68.79	66.14	65.36	67.76	71.82	56.47
BERT+DP	57.77 (±0.53)	58.97 (±1.83)	59.86*	66.31 (±0.50)	72.18 (±1.34)	58.84 (±1.79)

Table 7.10: Accuracies of the most relevant WSD/WSI methods on inventory independent datasets

from RandomWN3Sense baseline, and Trofi-AllWords was not using a good proxy for meaning, we drop these datasets.

To keep the most relevant results in one place, we repeat the results scattered over dataset-specific tables into Table 7.10, dropping less relevant methods — the baselines, Vial-hyper, and Vial+OntoNotes. We also repeat the subset of Table 5.2 showing the performances of relevant methods in Table 7.11. Since evaluating alternative sense-inventories and WSI methods were not possible using the benchmark datasets, we do not see Vial+OntoNotes+WN3 and BERT+DP in Table 5.2.

It is evident from Table 7.10 and Table 5.2 that EWISER, the state of the art according to the benchmark datasets of Raganato et al. [2017b], does not hold its status when evaluated on our inventory-independent datasets. Also, the results on sense-inventory independent datasets strengthens the evidence for the superiority of Vial+hyper over BEM and GlossBERT.

Method	SE2	SE3	SE 2007	SE 2013	SE 2015
Vial+hyper	79.7	77.8	73.4	78.7	**82.6**
BEM	79.4	77.4	74.5	79.7	81.7
GlossBERT	77.7	75.2	72.5	76.1	80.4
EWISER+WNGC	**80.8**	**79.0**	**75.2**	**80.7**	81.8

Table 7.11: Results of the most relevant WSD methods on Datasets in the unified framework of Raganato et al. [2017b]. This table is a repeated subset of Table 5.2

7.8 Summary

In this chapter, we briefly described WiC dataset of Pilehvar and Camacho-Collados [2019] and showed that it can be used as a sense inventory independent WSD/WSI evaluation dataset since it has taken the gold sense annotation out of the equation by pairing contexts and annotating them with "T" for same sense and "F" for different sense usages.

Since WiC was curated automatically, mainly from examples of WordNet sense with over 80% overlap in all partitions, it is too close to WordNet. This raises the question if it can fairly evaluate methods that use WordNet in any way.

To address this concern, we transformed three annotated datasets for different tasks into the WiC format and reported results of the select WSD/WSI methods on all these inventory independent datasets.

We showed that using this evaluation framework makes it possible to go beyond evaluating WSD methods restricted to WordNet senses and evaluate WSI methods as well as the use of alternative sense inventories in WSD methods.

Chapter 8

Conclusion

In this thesis, we have presented our work on two different tasks — biomedical named entity recognition and evaluating word sense disambiguation and induction.

In Part I we introduced GraphNER, a graph-based semi-supervised tool for biomedical NER and showed its efficacy on two benchmark datasets. We also presented our work, training an LSTM-CRF for biomedical NER that took contextualized embeddings, in this case ELMo, as input along with the traditional context-independent embeddings. We showed that the contextualized embeddings will improve the performance on several biomedical named entity types but only when they are trained in-domain even if the in-domain training set is small. Later methods [**?**] confirmed this conclusion and further advanced the field by training contextual embeddings on large biomedical data.

In Part II we propose an evaluation framework for both word sense disambiguation and word sense induction. Even though both these fields are aiming to make sense of different usages (senses) of words, the community is not comparing the methods on any task despite the fact that frameworks for such comparisons have been proposed before [**?**]. The difference between our framework and these previous attempts lies in the fact that there is no need for an extra step to map the sense clusters of WSI methods to the senses listed in sense inventories of WSD methods.

We propose to compare methods of both WSD and WSI by evaluating them on WiC-style datasets where two contexts sharing a lemma are presented and the systems are expected to predict if the two usages of the shared lemma convey the same meaning. Since WiC [Pilehvar and Camacho-Collados, 2019] was automatically curated from WordNET examples, it may unfairly favor the methods that use WordNET in one way or another. To rectify this problem, we transformed three existing datasets annotated with 1) senses from Oxford English Dictionary, an alternative proprietary sense inventory; 2) gold substitute words; 3) literal vs. non-literal usages to get three effectively new datasets, MASC-NOAD, LexSub-balanced, and Trofi-AllNegative with no overlap with WordNet examples.

We show that our evaluation framework makes it possible to compare both WSD and WSI methods and to evaluate alternative sense inventories. We also show that results on our inventory-independent datasets provide more evidence for efficacy of the compression method of Vial et al. [2019] as well as a new ranking between recent WSD methods.

8.1 Future Directions

A first step in continuing our work in Part I can be to see if the performance of GraphNER, BioELMo, and/or BioBERT is drastically lower on test instances with the least similarity to train instances as Elangovan et al. [2021] showed is the case for BERT. Based on the answer to that question, one might decide to proceed with a second step of re-thinking the graph construction based on other features such as contextualized embeddings.

As a continuation to our work in Part II, we provide multiple examples for future directions as follows:

1. **Comparing more methods.** We have evaluated many recent WSD methods according to our sense-inventory independent datasets. For the sake of comparison, we also evaluated BERT+DP, a recent state-of-the-art WSI method according to the traditional evaluation datasets, but we certainly did not exhaustively compare all WSD/WSI methods. While our experiments were enough to make the points we wanted to make, it is a good idea to evaluate all the outstanding WSD and WSI methods as our evaluation framework may change the rankings of the previous methods.

2. **Expanding OntoNotes.** Our experiments showed that at least for certain datasets, OntoNotes was a very good alternative for the WordNet sense inventory. This dataset, however, focuses on verbs and nouns only and has low coverage. While we bypassed this limitation by backing off to WordNet, it may be a good idea to continue the OntoNotes project. We also spotted not-so-rare problems or suspicious characteristics in the latest version of OntoNotes. For example in play-n.xml, among many other sense grouping files, all 14 OntoNotes labels have listed exactly one sense (the first WordNet sense for play as a noun), effectively mapping that one sense to multiple OntoNotes senses and leaving all the other senses un-mapped. This is clearly an error, not only because sense groupings should not result in one sense mapping to several groups, but also since the glosses and examples provided for different labels in the file match perfectly with those of different WordNet senses for the word. Another frequent problem is that 32% of all OntoNotes sense grouping files (for example bubble-v.xml) report inter-annotator agreements lower than 90%. For example, bubble-v.xml reports a quite low inter-annotator agreement of 0.2. This contradicts how OntoNotes was supposedly constructed. Fixing and/or double-checking these issues could help improve its efficacy as an alternative sense inventory for WSD.

3. **Combining advantages of OntoNotes and mapping to hypernyms.** Our experiments confirm that the method of Vial et al. [2019] for mapping senses to hypernyms, effectively

changing the sense inventory, consistently yields good results. We also observed that using OntoNotes sense inventory instead of the WordNet senses within their architecture yields good results on most datasets. While mapping to hypernyms is helpful because it gathers more examples for each label, OntoNote's success can be due to more examples per label and/or the fact that it groups WordNet senses to increase human inter-annotator agreements. Therefore, it is unknown at the moment, if the contributions of the two approaches are orthogonal. It is also uncertain if the two approaches can be combined at all. To combine the two approaches, we should decide what the hypernym for a group of senses means. Assuming we define it as the lowest synset that is on the hypernymy paths of all senses in the group, it is interesting to see how often the group hypernym is so high that mapping the group to that hypernym is impossible or hurts more than it helps.

4. **Guiding BERT+DP with labeled instances.** Even though BERT+DP is an unsupervised clustering approach, it is possible to guide it towards a sense inventory such as WordNet, by adding examples from each sense in the inventory into the instances it clusters and force the examples to fall into separate clusters. Taking WordNet, for example, it is possible to use all its glosses from the WNGC dataset. Using examples from OntoNotes would be a better bet however, as we already know WordNet senses are too fine-tuned and suffer lower inter-annotator agreements.

5. **More evaluation datasets.** Even though we have introduced three datasets, there is always room for more. One idea for collecting sense-inventory independent datasets for WSD/WSI evaluation is to present context pairs sharing a target word to language teachers and ask them if they expect the language learners to understand the meaning of the target word in the second context (assuming they know all the other words) provided that they are given a full explanation on what the target word means in the first context and that they do not know and have not seen any other example showcasing the usage of the target word in any other context. While we expect the teachers to say "no" where the two contexts refer to different meanings, it is important to know they may say "yes" relying on the students' human ability to infer certain concepts or understand metaphors among other things. It can be acceptable to have such difficult cases in an evaluation dataset however, as we would be comparing the methods to an ideal system whatever their current limitations might be.

Bibliography

Eneko Agirre and Aitor Soroa. Semeval-2007 task 02: Evaluating word sense induction and discrimination systems. In *Proceedings of the fourth international workshop on semantic evaluations (semeval-2007)*, pages 7–12, 2007.

Eneko Agirre and Aitor Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 33–41, 2009.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. Random walks for knowledge-based word sense disambiguation. *Computational Linguistics*, 40(1):57–84, 2014.

Andrei Alexandrescu and Katrin Kirchhoff. Graph-based learning for statistical machine translation. In *NAACL 2009*, 2009.

Reinald Kim Amplayo, Seung-won Hwang, and Min Song. Autosense model for word sense induction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6212–6219, 2019.

Asaf Amrami and Yoav Goldberg. Word sense induction with neural biLM and symmetric patterns. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4860–4867, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1523. URL `https://www.aclweb.org/anthology/D18-1523`.

Asaf Amrami and Yoav Goldberg. Towards better substitution-based word sense induction. *arXiv preprint arXiv:1905.12598*, 2019.

Rie Kubota Ando. BioCreative II gene mention tagging system at IBM Watson. In *Proceedings of the Second BioCreative Challenge Evaluation Workshop*, volume 23, pages 101–103. Centro Nacional de Investigaciones Oncologicas (CNIO) Madrid, Spain, 2007.

Amit Bagga and Breck Baldwin. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer, 1998.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Language resources and evaluation*, 43(3):209–226, 2009.

Osman Başkaya, Enis Sert, Volkan Cirik, and Deniz Yuret. AI-KU: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In *Second Joint*

*Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 300–306, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/S13-2050`.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 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. Association for Computational Linguistics, 2010.

Michele Bevilacqua and Roberto Navigli. Breaking through the 80% glass ceiling: Raising the state of the art in word sense disambiguation by incorporating knowledge graph information. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2864, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.255. URL `https://www.aclweb.org/anthology/2020.acl-main.255`.

Julia Birke and Anoop Sarkar. A clustering approach for nearly unsupervised recognition of nonliteral language. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, 2006.

Julia Birke and Anoop Sarkar. Active learning for the identification of nonliteral language. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*, pages 21–28, 2007.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Terra Blevins and Luke Zettlemoyer. Moving down the long tail of word sense disambiguation with gloss informed bi-encoders. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1006–1017, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.95. URL `https://www.aclweb.org/anthology/2020.acl-main.95`.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

David Campos, Sérgio Matos, and José Luís Oliveira. Gimli: open source and high-performance biomedical name recognition. *BMC bioinformatics*, 14(1):54, 2013.

Soanes Catherine and Stevenson Angus. Oxford dictionary of english, 2005.

Baobao Chang, Wenzhe Pei, and Miaohong Chen. Inducing word sense with automatically learned hidden concepts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 355–364, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/C14-1035`.

Devendra Singh Chaplot and Ruslan Salakhutdinov. Knowledge-based word sense disambiguation using topic models, 2018. URL `https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17415`.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *INTERSPEECH*, 2014.

Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. In *TACL 2016.*, 2016.

Hong-Jie Dai, Po-Ting Lai, Yung-Chun Chang, and Richard Tzong-Han Tsai. Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. *Journal of cheminformatics*, 7(S1):1–10, 2015.

Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005(09):P09008, 2005.

Dipanjan Das and Slav Petrov. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 600–609. Association for Computational Linguistics, 2011.

Dipanjan Das and Noah A Smith. Graph-based lexicon expansion with sparsity-inducing penalties. In *Proceedings of the 2012 conference of the North American chapter of the Association for Computational Linguistics: human language technologies*, pages 677–687. Association for Computational Linguistics, 2012.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A Smith. Probabilistic frame-semantic parsing. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 948–956. Association for Computational Linguistics, 2010.

Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. In *International conference on machine learning*, pages 933–941. PMLR, 2017.

Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

Paramveer S Dhillon, Partha Talukdar, and Koby Crammer. Metric learning for graph-based domain adaptation. 2012.

Philip Edmonds and Scott Cotton. Senseval-2: overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, 2001.

Aparna Elangovan, Jiayuan He, and Karin Verspoor. Memorization vs. generalization: Quantifying data leakage in nlp performance evaluation. *EACL*, 2021.

Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. Background to framenet. *International journal of lexicography*, 16(3):235–250, 2003.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. Posterior regularization for structured latent variable models. *The Journal of Machine Learning Research*, 11:2001–2049, 2010.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 2017.

Graciela H Gonzalez, Tasnia Tahsin, Britton C Goodale, Anna C Greene, and Casey S Greene. Recent advances and emerging applications in text and data mining for biomedical discovery. *Briefings in bioinformatics*, 17(1):33–42, 2016.

Kartik Goyal and Eduard Hovy. Unsupervised word sense induction using distributional statistics. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1302–1310, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/C14-1123`.

Yves Grandvalet and Yoshua Bengio. Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems*, pages 529–536, 2004.

Andreas Griewank et al. On automatic differentiation. *Mathematical Programming: recent developments and applications*, 6(6):83–107, 1989.

Maryam Habibi, Leon Weber, Mariana Neves, David Luis Wiegandt, and Ulf Leser. Deep learning with word embeddings improves biomedical named entity recognition. *Bioinformatics*, 33(14): i37–i48, 2017.

Christian Hadiwinoto, Hwee Tou Ng, and Wee Chung Gan. Improved word sense disambiguation using pre-trained contextualized word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5297–5306, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1533. URL `https://www.aclweb.org/anthology/D19-1533`.

Kai Hakala, Sofie Van Landeghem, Tapio Salakoski, Yves Van de Peer, and Filip Ginter. Application of the evex resource to event extraction and network construction: Shared task entry and result analysis. *BMC bioinformatics*, 16(Suppl 16):S3, 2015.

Luheng He, Jennifer Gillenwater, and Ben Taskar. Graph-based posterior regularization for semi-supervised structured prediction. In *Proceedings of CoNLL*, page 38, 2013.

Scott J Hebbring, Majid Rastegar-Mojarad, Zhan Ye, John Mayer, Crystal Jacobson, and Simon Lin. Application of clinical text data for phenome-wide association studies (PheWASs). *Bioinformatics*, 31(12):1981–1987, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Eduard Hovy, Mitch Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, 2006.

Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3509–3514, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1355. URL `https://www.aclweb.org/anthology/D19-1355`.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. Embeddings for word sense disambiguation: An evaluation study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 897–907, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1085. URL `https://www.aclweb.org/anthology/P16-1085`.

Nancy Ide and Keith Suderman. The american national corpus first release. In *LREC*, 2004.

Nancy Ide, Collin Baker, Christiane Fellbaum, Charles Fillmore, and Rebecca Passonneau. MASC: the manually annotated sub-corpus of American English. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2008/pdf/617_paper.pdf`.

Feng Jiao, Shaojun Wang, Chi-Hoon Lee, Russell Greiner, and Dale Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 209–216. Association for Computational Linguistics, 2006.

Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. Probing biomedical embeddings from language models. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 82–89, Minneapolis, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-2011. URL `https://www.aclweb.org/anthology/W19-2011`.

Richard Johansson and Pierre Nugues. Lth: semantic structure extraction using nonprojective dependency trees. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 227–230. Association for Computational Linguistics, 2007.

John Judge, Aoife Cahill, and Josef Van Genabith. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 497–504. Association for Computational Linguistics, 2006.

David Jurgens. An evaluation of graded sense disambiguation using word sense induction. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 189–198, 2012.

David Jurgens and Ioannis Klapaftis. Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 290–299, 2013.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. Genia corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182, 2003.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Association for Computational Linguistics, 2004.

Paul R Kingsbury and Martha Palmer. From treebank to propbank. In *LREC*, pages 1989–1993. Citeseer, 2002.

Alexandros Komninos and Suresh Manandhar. Structured generative models of continuous features for word sense induction. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3577–3587, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL `https://www.aclweb.org/anthology/C16-1337`.

Martin Krallinger, Obdulia Rabal, Florian Leitner, Miguel Vazquez, David Salgado, Zhiyong Lu, Robert Leaman, Yanan Lu, Donghong Ji, Daniel M Lowe, et al. The chemdner corpus of chemicals and drugs and its annotation principles. *Journal of cheminformatics*, 7(S1):1–17, 2015.

Ravi Kumar and Sergei Vassilvitskii. Generalized distances between rankings. In *Proceedings of the 19th international conference on World wide web*, pages 571–580, 2010.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001a.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001b.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N16-1030`.

Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

Robert Leaman, Graciela Gonzalez, et al. Banner: an executable survey of advances in biomedical named entity recognition. In *Pacific Symposium on Biocomputing*, volume 13, pages 652–663. Citeseer, 2008.

Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917, 2013.

Robert Leaman, Chih-Hsuan Wei, and Zhiyong Lu. tmchem: a high performance approach for chemical named entity recognition and normalization. *J. Cheminformatics*, 7(S-1):S3, 2015.

Hee-Jin Lee, Tien Cuong Dang, Hyunju Lee, and Jong C Park. Oncosearch: cancer gene search engine with literature evidence. *Nucleic acids research*, page gku368, 2014.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, 1986.

Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. Upset: visualization of intersecting sets. *IEEE transactions on visualization and computer graphics*, 20(12):1983–1992, 2014.

Gang Li, Karen E Ross, Cecilia N Arighi, Yifan Peng, Cathy H Wu, and K Vijay-Shanker. mirtex: A text mining system for mirna-gene relation extraction. *PLoS Comput Biol*, 11(9):e1004391, 2015.

M Liberman and M Mandel. Pennbioie. *Linguistic Data Consortium, Philadelphia*, 2008.

Dekang Lin. Principle-based parsing without overgeneration. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 112–120. Association for Computational Linguistics, 1993.

Dekang Lin. Principar: an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th conference on Computational linguistics-Volume 1*, pages 482–488. Association for Computational Linguistics, 1994.

Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*, pages 768–774. Association for Computational Linguistics, 1998.

Jerry Chun-Wei Lin, Yinan Shao, Youcef Djenouri, and Unil Yun. Asrnn: a recurrent neural network with an attention model for sequence labeling. *Knowledge-Based Systems*, 212:106548, 2021.

Jianhua Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, 1991.

Christine A Lindberg and Angus Stevenson. *New Oxford American Dictionary*. Oxford University Press, 1999.

Shujie Liu, Chi-Ho Li, Mu Li, and Ming Zhou. Learning translation consensus with structured label propagation. In *ACL 2012*, 2012.

Fuli Luo, Tianyu Liu, Qiaolin Xia, Baobao Chang, and Zhifang Sui. Incorporating glosses into neural word sense disambiguation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2473–2482, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1230. URL `https://www.aclweb.org/anthology/P18-1230`.

Yuan Luo, Özlem Uzuner, and Peter Szolovits. Bridging semantics and syntax with graph algorithms—state-of-the-art of extracting biomedical relations. *Briefings in bioinformatics*, page bbw001, 2016.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. Semeval-2010 task 14: Word sense induction &disambiguation. In *Proceedings of the 5th international workshop on semantic evaluation*, pages 63–68, 2010.

Gideon S Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 109–112. Association for Computational Linguistics, 2007.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014a. URL `http://www.aclweb.org/anthology/P/P14/P14-5010`.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014b.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. 1993.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics, 1994.

Diana McCarthy and Roberto Navigli. SemEval-2007 task 10: English lexical substitution task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 48–53, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/S07-1009`.

Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61, 2016.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *ICLR*, 2013.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

George A Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. Using a semantic concordance for sense identification. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994a.

George A Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. Using a semantic concordance for sense identification. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*, 1994b.

Alistair Moffat and Justin Zobel. Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems (TOIS)*, 27(1):1–27, 2008.

Andrea Moro and Roberto Navigli. Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 288–297, 2015.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. Entity linking meets word sense disambiguation: a unified approach. *Transactions of the Association for Computational Linguistics*, 2: 231–244, 2014.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. Geometry of polysemy. In *Proceedings of the 5th International Conference on Learning Representations. OpenReview.net.*, 2017.

Tsendsuren Munkhdalai, Meijing Li, Khuyagbaatar Batsuren, Hyeon Park, Nak Choi, and Keun Ho Ryu. Incorporating domain knowledge in chemical and biomedical named entity recognition with word representations. *J. Cheminformatics*, 7(S-1):S9, 2015.

Roberto Navigli and Simone Paolo Ponzetto. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193: 217–250, 2012.

Roberto Navigli, Kenneth C. Litkowski, and Orin Hargraves. SemEval-2007 task 07: Coarse-grained English all-words task. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 30–35, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/S07-1006`.

Roberto Navigli, David Jurgens, and Daniele Vannella. Semeval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, 2013.

Sebastian Padó. *User's guide to `sigf`: Significance testing by approximate randomisation*, 2006a.

Sebastian Padó. *User's guide to `sigf`: Significance testing by approximate randomisation*, 2006b.

Abhishek Panigrahi, Harsha Vardhan Simhadri, and Chiranjib Bhattacharyya. Word2sense: sparse interpretable word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5692–5705, 2019.

Simone Papandrea, Alessandro Raganato, and Claudio Delli Bovi. SupWSD: A flexible toolkit for supervised word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 103–108, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-2018. URL `https://www.aclweb.org/anthology/D17-2018`.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, 2018a. Association for Computational Linguistics. URL `"http://aclweb.org/anthology/N18-1202"`.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018b.

Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2089–2096, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2012/pdf/274_Paper.pdf`.

Andrew Philpot, Eduard Hovy, and Patrick Pantel. The omega ontology. In *Proceedings of OntoLex 2005-Ontologies and Lexical Resources*, 2005.

Mohammad Taher Pilehvar and Jose Camacho-Collados. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1128. URL `https://www.aclweb.org/anthology/N19-1128`.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. Semeval-2007 task-17: English lexical sample, srl and all words. In *Proceedings of the fourth international workshop on semantic evaluations (SemEval-2007)*, pages 87–92, 2007.

Sampo Pyysalo, Tomoko Ohta, Rafal Rak, Andrew Rowley, Hong-Woo Chun, Sung-Jae Jung, Sung-Pil Choi, Jun'ichi Tsujii, and Sophia Ananiadou. Overview of the cancer genetics and pathway curation tasks of bionlp shared task 2013. *BMC bioinformatics*, 16(Suppl 10):S2, 2015.

Alessandro Raganato, Claudio Delli Bovi, and Roberto Navigli. Neural sequence learning models for word sense disambiguation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1156–1167, 2017a.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. Word sense disambiguation: A unified evaluation framework and empirical comparison. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 99–110, 2017b.

Marek Rei, Gamal Crichton, and Sampo Pyysalo. Attending to characters in neural sequence labeling models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 309–318, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1030.

Andrew Rosenberg and Julia Hirschberg. 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 (EMNLP-CoNLL)*, pages 410–420, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D07-1043.

Sascha Rothe and Hinrich Schütze. AutoExtend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1173. URL https://www.aclweb.org/anthology/P15-1173.

Avneesh Saluja, Hany Hassan, Kristina Toutanova, and Chris Quirk. Graph-based semi-supervised learning of translation models from monolingual data. In *ACL 2014*, 2014.

Karin Kipper Schuler. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD thesis, University of Pennsylvania, 2006. URL http://verbs.colorado.edu/~kipper/Papers/dissertation.pdf.

Henry J Scudder. Probability of error of some adaptive pattern-recognition machines. *Information Theory, IEEE Transactions on*, 11(3):363–371, 1965.

Golnar Sheikhshab, Elizabeth Starks, Aly Karsan, Anoop Sarkar, and Inanc Birol. Graph-based semi-supervised gene mention tagging. In *Proceedings of the 15th Workshop on Biomedical Natural Language Processing*, pages 27–35, 2016.

Golnar Sheikhshab, Inanc Birol, and Anoop Sarkar. In-domain context-aware token embeddings improve biomedical named entity recognition. In *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis*, pages 160–164, 2018a.

Golnar Sheikhshab, Elizabeth Starks, Readman Chiu, Aly Karsan, Anoop Sarkar, and Inanc Birol. Graphner: Using corpus level similarities and graph propagation for named entity recognition. In *Proceedings of the 2018 IEEE International Parallel and Distributed Processing Symposium Workshops*, pages 229–238, 2018b.

Larry Smith, Lorraine K Tanabe, Rie Johnson nee Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, et al. Overview of biocreative ii gene mention recognition. *Genome biology*, 9(2):S2, 2008.

Benjamin Snyder and Martha Palmer. The english all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, 2004.

Linfeng Song, Zhiguo Wang, Haitao Mi, and Daniel Gildea. Sense embedding learning for word sense induction. In *Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics*, pages 85–90, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/S16-2009. URL `https://www.aclweb.org/anthology/S16-2009`.

Karl Stratos and Michael Collins. Simple semi-supervised pos tagging. 2005.

Amarnag Subramanya and Jeff A Bilmes. Entropic graph regularization in non-parametric semi-supervised classification. In *Advances in Neural Information Processing Systems*, pages 1803–1811, 2009.

Amarnag Subramanya, Slav Petrov, and Fernando Pereira. Efficient graph-based semi-supervised learning of structured tagging models. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 167–176. Association for Computational Linguistics, 2010.

Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: Principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159, 2012. doi: 10.2200/S00433ED1V01Y201207DMK005.

Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Machine Learning*, 4(4):267–373, 2011.

Jun Suzuki and Hideki Isozaki. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*, pages 665–673, 2008.

Kaveh Taghipour and Hwee Tou Ng. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the nineteenth conference on computational natural language learning*, pages 338–344, 2015a.

Kaveh Taghipour and Hwee Tou Ng. One million sense-tagged instances for word sense disambiguation and induction. In *Proceedings of the nineteenth conference on computational natural language learning*, pages 338–344, 2015b.

Kaveh Taghipour and Hwee Tou Ng. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 314–323, 2015c.

Partha Pratim Talukdar and Fernando Pereira. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1473–1481. Association for Computational Linguistics, 2010.

Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP 2008*, 2008.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. Bilingual lexicon extraction from comparable corpora using label propagation. In *EMNLP-CoNLL 2012*, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Loïc Vial, Benjamin Lecouteux, and Didier Schwab. UFSAC: Unification of sense annotated corpora and tools. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). URL `https://www.aclweb.org/anthology/L18-1166`.

Loïc Vial, Benjamin Lecouteux Didier, and Schwab. Sense vocabulary compression through the semantic knowledge of wordnet for neural word sense disambiguation. In *Wordnet Conference*, page 108, 2019.

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

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ICLR*, 2019.

Jing Wang, Mohit Bansal, Kevin Gimpel, Brian D. Ziebart, and Clement T. Yu. A sense-topic model for word sense induction with unsupervised data enrichment. *Transactions of the Association for Computational Linguistics*, 3:59–71, 2015. doi: 10.1162/tacl_a_00122. URL `https://www.aclweb.org/anthology/Q15-1005`.

Yinglin Wang, Ming Wang, and Hamido Fujita. Word sense disambiguation: A comprehensive knowledge exploitation framework. *Knowledge-Based Systems*, 190:105030, 2020. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2019.105030. URL `http://www.sciencedirect.com/science/article/pii/S0950705119304344`.

Leilei Yang, Guiquan Liu, Qi Liu, Lei Zhang, and Enhong Chen. Analyzing sequence data based on conditional random fields with co-training. In *Computational Intelligence and Security (CIS), 2012 Eighth International Conference on*, pages 94–98. IEEE, 2012.

Alexander Yeh. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 947–953. Association for Computational Linguistics, 2000.

Dayu Yuan, Julian Richardson, Ryan Doherty, Colin Evans, and Eric Altendorf. Semi-supervised word sense disambiguation with neural models. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1374–1385, Osaka, Japan, December 2016. The COLING 2016 Organizing Committee. URL `https://www.aclweb.org/anthology/C16-1130`.

Zhi Zhong and Hwee Tou Ng. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 system demonstrations*, pages 78–83, 2010.

Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.