Learning a semantic database from unstructured text

by Keshav Dhandhania

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
May 23, 2014
[JUNE 2014]

Signature redacted

Author: .....
Department of Electrical Engineering and Computer Science
May 23, 2014

Signature redacted

Certified by: .....
Prof. Tommi Jaakkola
Thesis Supervisor

Signature redacted

Accepted by: .....
Prof. Albert R. Meyer
Chairman, Masters of Engineering Thesis Committee

Learning a semantic database from unstructured text

by Keshav Dhandhania

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
May 23, 2014

Abstract

In this paper, we aim to learn a semantic database given a text corpus. Specifically, we focus on predicting whether or not a pair of entities are related by the hypernym relation, also known as the 'is-a' or 'type-of' relation. We learn a neural network model for this task. The model is given as input a description of the words and the context from the text corpus in which a pair of nouns (entities) occur. In particular, among other things the description includes pre-trained embeddings of the words. We show that the model is able to predict hypernym noun pairs even though the dataset includes many incorrectly labeled noun pairs. Finally, we suggest ways to improve the dataset and the method.

Acknowledgments

Firstly, I would like to thank my parents for their constant encouragement and support throughout my life.

Secondly, I would like to thank my thesis supervisor, Tommi Jaakkola, for his expertise and time. The pointers he provided led me to explore directions I would not have explored otherwise. His input helped me make decisions with clarity and thoughtfulness, instead of running experiments for each possible choice and evaluating empirically. Having him examine the model from different aspects and answering his questions deepened my understanding of my own work. I also greatly appreciate his high level and detailed feedback on the thesis write-up. Finally, and most importantly, I would like to thank him for guiding the scope of the thesis to be within reach yet challenging.

Finally, I would like to thank MIT, MIT EECS and MIT CSAIL for providing an environment which makes all of this possible.

Contents

1 Introduction **5**

2 Glossary **6**

3 Problem statement **8**

4 Previous work **9**

5 Methodology **10**
5.1 Description of features 10
5.1.1 Extending the corpus 11
5.1.2 Evidence 11
5.2 Description of neural network model 13
5.2.1 Notation 14
5.2.2 Neural network architecture 15

6 Dataset **19**

7 Experiments **23**

8 Results **24**

9 Discussion **27**
9.1 Dataset errors 27
9.2 Baseline model errors 28
9.3 Neural network model errors 29

10 Future work **32**
10.1 Dataset errors 33
10.2 Recursive neural network 33

11 Conclusion **35**

1 Introduction

Artificial intelligence and machine learning methods are increasingly used to solve high level applications today. Such applications include Jeopardy playing IBM Watson [9] and high accuracy automatic speech recognition [12] and object recognition [13]. In-order to perform and/or assist in higher-level tasks, our computers need a better understanding of the world we live in. They need to understand the entities that surround us, and how these entities are related to each other. In other words, they need to have access to a comprehensive, semantic database.

There are a number of semantic databases which exist currently and are available for artificial intelligence and machine learning methods to use. These include Freebase [2], WordNet [8], ConceptNet [10] and YAGO [20]. However, the construction of these semantic databases have so far relied on extensive manual effort. For example, Freebase and WordNet have both been created manually. Freebase is crowd-sourced, whereas WordNet is constructed by experts. Even when the construction process is not entirely manual, it relies on the presence of structured data in some form. YAGO and ConceptNet fall into this category. YAGO relies on the structured infoboxes on Wikipedia and ConceptNet compiles many existing semantic databases into one.

The fact that humans are heavily involved in the process of creating a semantic database makes the cost of building a comprehensive database

prohibitive. Moreover, once built, most semantic databases still require significant human intervention for maintenance, and for adapting them to a particular task. Hence, it would be highly desirable to have automated mechanisms for constructing or extending semantic databases using only unstructured information. In this thesis, we take a step towards achieving this goal.

The rest of this thesis is structured as follows. In section 2 we introduce some notation that will be used throughout the document. In section 3 we state our problem more formally. In section 4 we discuss previous work on tasks similar to ours. In section 5 we describe our approach to the problem. This section consists of two major parts. In the first half, section 5.1, we describe formally how features used in our model are extracted from the text corpus. In the second half, section 5.2, we describe the model itself. Section 6 and section 7 include a description of the dataset and details of our experiments. We present the results in section 8 and discuss them in section 9. In section 10 we propose possible future work to remedy the main categories of errors made by our model. Finally, we conclude in section 11.

2 Glossary

In this section we introduce some terminology that will be used throughout the rest of the document.

entity: Entities are the objects between which hypernym relations are de-

fined. In general, entities appear in English text as noun phrases. However, for most of this document, an entity will be equivalent to an English noun word. As such, we use the words *noun* and *entity* interchangeably apart from when the difference is clear based on context.

hypernym relation: The hypernym relation is also known as the 'is-a', the 'type-of' or the 'kind of' relation. For example, a horse *is an* animal, and Albert Einstein *is a* physicist. Hence, the entity pairs (animal, horse) and (physicist, Einstein) are hypernyms, or satisfy the hypernym relation. Note that the order in which the entities occur matters. We will refer to the broader entity as the *parent entity*, and the more specific entity as the *child entity*. So for example, for the hypernym pair (animal, horse), animal is the parent entity, and horse is the child entity. Some example of non-hypernym entity pairs include (horse, animal), (bird, horse) and (vegetable, banana), i.e. an animal is not (necessarily) a horse, a horse is not a bird, and a banana is not a vegetable.

embeddings: Embeddings are distributed vector representations of words. For most of the document, it will help to keep in mind that there is exactly one embedding for each English word. Embeddings are learned on natural language processing tasks such as part-of-speech tagging, chunking, named entity recognition, parsing, semantic role labeling and language modeling [1, 6, 15, 24]. They have been shown to capture syntactic and semantic information about the words [14].

3 Problem statement

The central problem we focus on in this thesis is automatically inferring a semantic database from an unstructured text corpus. Although a full semantic database would involve a number of different relations, we restrict our focus to the hypernym relation, also known as the 'is-a' relation or the 'kind of' relation. The hypernym relation is defined in detail in section 2.

The problem setup is as follows. We will be given a text corpus and a set of labeled entity pairs. Each entity pair consists of the parent noun and the child noun. And the label is either hypernym or non-hypernym. We are also given a set of unlabeled entity pairs, for which the model needs to make predictions. In particular, the model predicts a number between 0 and 1, where the number represents the confidence that the pair is a hypernym relation. Note that the unlabeled entity pairs include entities for which there are no labeled entity pairs available. As such, the model must be capable of handling new or unseen entities at prediction time.

Finally, note that the hypernym relation is transitive. For example, a horse is an animal, and an animal is an organism together imply that a horse is an organism. Hence, if a model uses features of the entities themselves (such as the embeddings of the entity nouns), then splitting strategy for the train and test dataset should be explicitly designed to avoid the possibility of models simply memorizing the labeled entity pairs and exploiting the transitivity property of the hypernym relation.

4 Previous work

Previous work on hypernym prediction can be divided into two broad categories. The first approach focuses on identifying syntactic patterns which are good indicators of the hypernym relation. Previous work along this line does not make use of word embeddings. The second approach only utilizes the embeddings of the entities in order to make predictions. In particular, it does not look at a text corpus. Previous work that combines word embeddings and the text corpus uses it for semantic tasks different from relation extraction, such as semantic role labeling. We describe each of these three lines of work in detail in the following three paragraphs.

The use of syntactic patterns was first made popular by Hearst [11]. She noted that 'is-a' relationships are indicated in text by a number of syntactic patterns such as "X is a Y", "Y, such as X" and "X and other Y". Her work used a fixed set of a dozen such syntactic patterns in-order to predict hypernym relations. Snow et al. [17] built upon Hearst's work by applying regression on hundreds of thousands of such syntactic patterns instead of only a hand-full of them. Each of their syntactic patterns corresponded to a shortest path linking two single-word noun entities in the dependency parse tree of a sentence.

Mikolov et. al [14] analyze the information captured by word embeddings by showing regularities and patterns in the learned word embeddings, such as $women - men \approx queen - king$. They use these regularities to answer

syntactic analogy questions. They restricted themselves to using methods based solely on the offsets of word embeddings. Chen et. al [3] train a neural tensor network for predicting relations between entities. Their method handles a number of different relations by learning a separate neural tensor network for each relation. However, since they modify the embeddings of the entity words during training, their method cannot handle unseen entities in the test dataset.

Lastly, Collobert et. al [5,6] learned models jointly on a number of different tasks, including semantic role labeling. For this task, their neural network model relied on the embeddings of the words and the context. However, as previously mentioned, they did not attempt semantic tasks such as predicting relations between entities.

5 Methodology

In this section we describe our approach to hypernym prediction. The section comprises of two major parts. The first half describes the pre-processing we do on the text corpus in order to extract relevant information that will be given as input to our neural network model. In the second half we describe the neural network model itself.

5.1 Description of features

Our dataset comprises of a set of labeled entity pairs. Each entity pair is a pair of nouns (p, c), and the label of the entity pair is given by $l = 1$ if p

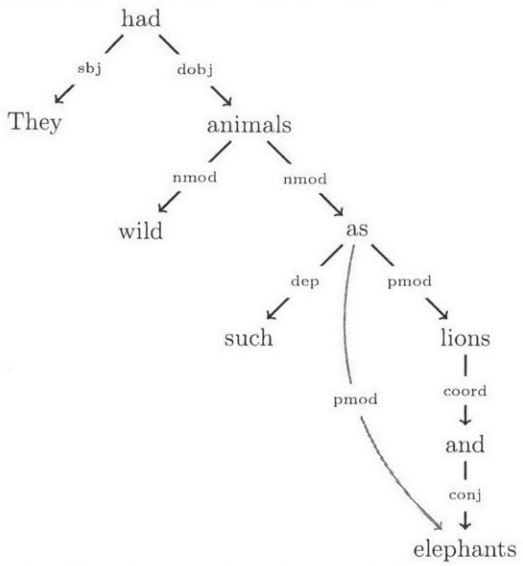
is a hypernym of c and $l = 0$ otherwise. In order to predict whether or not the noun pair (p, c) is a hypernym, we will make use of the text corpus and pre-trained word embeddings. In this subsection, we describe how we get all the evidence about a noun pair from the text corpus. Each piece of evidence includes descriptions of the words and the context in which the noun pair appears. All the evidence is then be used by the neural network for training and prediction.

5.1.1 Extending the corpus

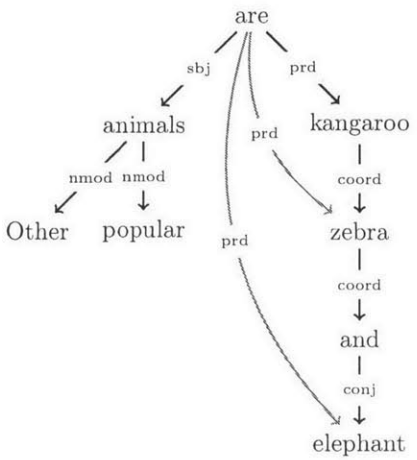
We begin by running a part-of-speech tagger and a dependency parser on the entire corpus. The dependency trees of the sentences have labeled arcs and the trees are augmented by adding *conjunction links*, i.e. new dependency arcs connecting the parent of a head conjunct to all the other conjuncts. The arc-label of the conjunction link is the same as the arc-label of the dependency between the head conjunct and its parent. Figure 1 shows the dependency structures of two sentences with conjunction links. From now on, we will refer to the original dependency parse tree by the phrase *dependency parse tree*, and to the dependency parse tree plus conjunction links by the phrase *dependency parse structure*.

5.1.2 Evidence

Let S be a sentence in which both p and c appear as nouns, in any order. Let P denote the path from p to c in the dependency parse tree of S. If there exists a conjunction link that spans multiple arcs of the path P, then the path P is shortened by replacing the arcs with the conjunction link. We



They had wild animals such as lions and elephants.



Other popular animals are kangaroo, zebra and elephant.

Figure 1: The dependency parse structures. Conjunction links shown in gray.

only look at paths P with length $k \leq L$ for some fixed L.

Let a *satellite link* be a link from p or c to their dependents or a special NULL word, i.e. if p has n dependents, then there are $n+1$ possible satellite links from p, and similarly for c.

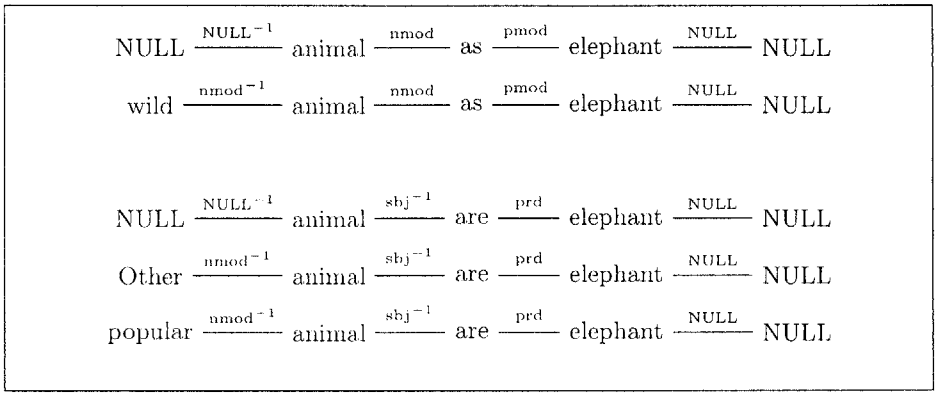


Figure 2: All the evidence corresponding to noun pair (animal, elephant) obtained from the sentences in figure 1

We are finally ready to define the evidence about a noun pair (p, c). *Evidence* E about a noun pair (p, c) consists of a path P from p to c in the dependency parse structure of sentence S, and one satellite link from p and c. The satellite links included in evidence E are not to words already included in the path P. We get new pieces of evidence from different sentences as well as from different pairs of satellite links appearing in the same sentence S. Figure 2 illustrates all the evidence we get for the noun pair (animal, elephant) from the sentences in figure 1.

5.2 Description of neural network model

We learn a neural network for predicting whether or not a noun pair is a hypernym. The neural network is trained with back-propagation. The cost

function is given by

$$-\sum_{i \in D} l_i \log f(S_i) + (1 - l_i) \log(1 - f(S_i))$$

where l_i is the label of the ith noun pair, S_i is all the evidence associated with the ith noun pair, and f is the function represented by the neural network.

The neural network has a total of four layers. The first two layers process each piece of evidence about the noun pair separately from others, whereas the last two layers combine all the available evidence about the noun pair.

Before describing the architecture of the neural network in detail, we introduce some notation. Thereafter, we describe the layers one by one.

5.2.1 Notation

n is the number of pieces of evidence corresponding to the noun pair (p, c) and E_i denotes the ith piece. When we say, $w \in E_i$, we are referring to all the words in the ith piece of evidence E_i, and by $a \in E_i$ we are referring to the arcs in E_i. Each word w in evidence E_i has two other properties, its part-of-speech tag and its location, denoted by $loc(w)$ and $POS(w)$, respectively. Each arc a in E_i also has the location property, denoted by $loc(a)$. Note that it is possible for the same English word or arc-label to appear in different locations across different pieces of evidence or even in the same E_i. The same holds for the part-of-speech tags of words. Hence loc and

POS are functions which depend on the context in which the word or arc occurs. We denote the length of E_i by $len(E_i)$, where length is defined as the number of arcs between p and c in E_i.

$emb(w)$ denotes the pre-trained word embedding of word w and is a vector of length d. In particular, it is a strict function of the English word w itself, and does not depend on the context or its part-of-speech tag. $emb_2(w_1, w_2)$ denotes a vector of length $(2d)^2$ consisting of all the second-order terms from the outer product of vector $[emb(w_1); emb(w_2)]$ with itself.

5.2.2 Neural network architecture

We are now ready to describe the neural network architecture. It may help to refer to figure 3 while going through the description. The network has a total of 4 layers. The first two layers operate on each piece of evidence independently. After the transformations corresponding to the first two layers of the network, the hidden units form a matrix of dimension sz_2 x n, where each evidence E_i results in the ith column of the matrix, a vector of length sz_2.

Layer one

The first layer combines input from the words and arcs. It has three main components. The first component gets as input the word embeddings $emb(w)$ of the words w in evidence E. The weights which operate on these embeddings depend on the length of E, the location of w and the part-of-speech tag of w. The second component looks at the arcs in evidence E.

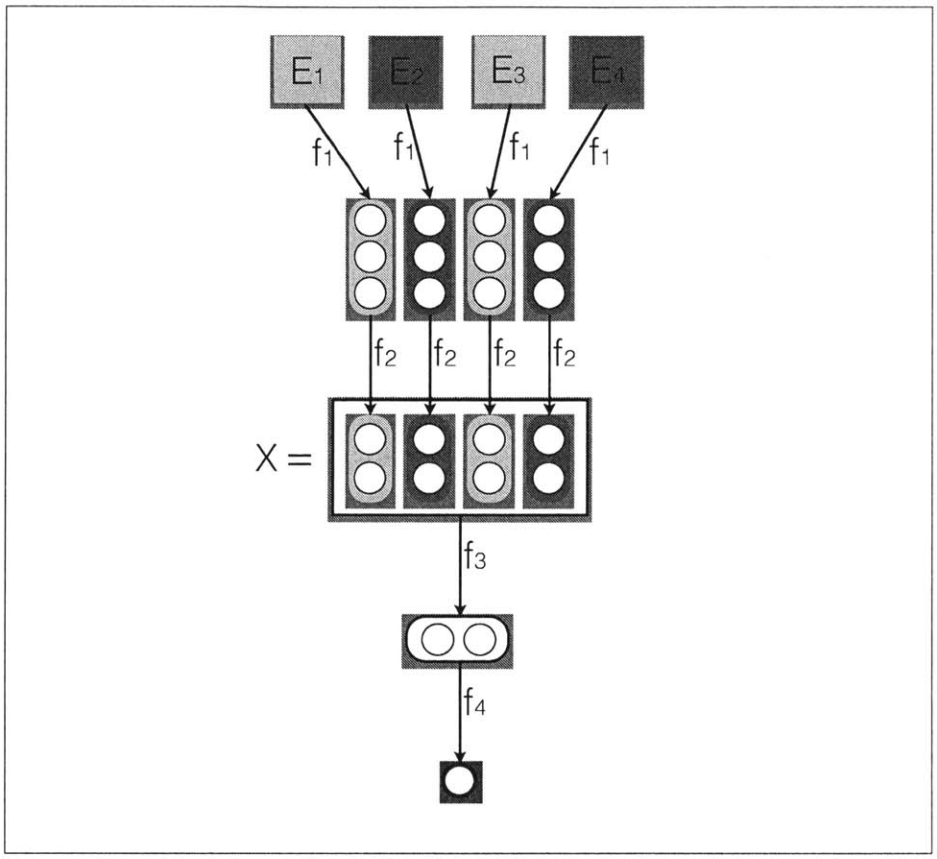


Figure 3: Overall network architecture with $n = 4$, $sz_1 = 3$ and $sz_2 = 2$. Functions f_1, f_2, f_3 and f_4 are defined in section 5.2.2

The weights for a particular arc a depend on the length of E, the location of a and the arc-label itself. The third component gets as input the second order features from the embeddings of the noun pair (p, c). The weights that operate on this feature depend only on the length of E. More concretely, the three components of the first layer, and the first layer itself are given by

the following functions g_W, g_A, g_N and f_1 respectively

$$g_W(E) = \sum_{w \in E} W_W(len(E), loc(w), POS(w)) \cdot emb(w)$$

$$g_A(E) = \sum_{a \in E} W_A(len(E), loc(a), a)$$

$$g_N(E) = W_N(len(E)) \cdot emb_2(p, c)$$

$$f_1(E) = sig(b_1 + g_W(E) + g_A(E) + g_N(E))$$

where $W_W(\cdot, \cdot, \cdot)$ is a weight matrix with dimensions sz_1 x d, $W_A(\cdot, \cdot, \cdot)$ is a weight vector of length sz_1, $W_N(\cdot)$ is a weight matrix with dimensions sz_1 x $(2d)^2$, sig is the sigmoid function and b_1 is a bias vector of length sz_1. Here we use $W(\cdot)$ as a shorthand for saying that $W(z)$ satisfies the property for each valid value of parameter z.

Illustration of layer one

The functions g_W, g_A and g_N are illustrated below for the second piece of evidence in figure 2.

$$\begin{aligned} g_W(E) = {} & W_W(2,\ 1,\ \text{adj}) \cdot emb(\text{wild}) + W_W(2,\ 2,\ \text{noun}) \cdot emb(\text{animal}) + \\ & W_W(2,\ 3,\ \text{prep}) \cdot emb(\text{as}) + W_W(2,\ 4,\ \text{noun}) \cdot emb(\text{elephant}) + \\ & W_W(2,\ 5,\ \text{NULL}) \cdot emb(\text{NULL}) \\ g_A(E) = {} & W_A(2,\ 1,\ \text{nmod}^{-1}) + W_A(2,\ 2,\ \text{nmod}) + W_A(2,\ 3,\ \text{pmod}) + \\ & W_A(2,\ 4,\ \text{NULL}) \\ g_N(E) = {} & W_N(2) \cdot emb_2(\text{animal, elephant}) \end{aligned}$$

Layer two

The second layer of the network is a simple sigmoid layer, but is applied to each evidence separately. It is represented by the function $f_2(x) = sig(b_2 + W_2 x)$, where b_2 is a bias vector of length sz_2 and W_2 is a weight matrix with dimensions sz_2 x sz_1.

$f_2(f_1(E))$ gives a vector of length sz_2, but the entire input X into layer 3 is the matrix formed by arranging all the vectors as columns of a matrix, where each evidence E_i of noun pair (p, c) contributes column i. Hence, the matrix X has dimension sz_2 x n.

Layer three

The third layer is a max-pooling layer where the max-pooling is happening across all the evidence, i.e. $f_3(X) = rowMax(X)$. The result is a vector with element j being the maximum value of row j of X.

Layer four

Lastly, the fourth layer is also a sigmoid layer, i.e. $f_4(x) = sig(b_4 + W_4 x)$, where b_4 is a scalar and W_4 is a weight vector of length sz_2. Parameter b_4 is not learned, and is rather kept fixed at a large negative value, giving incentive to the elements of W_4 to be positive. We do this to capture the intuition that a noun pair is a non-hypernym by default, and features mostly have information that provide evidence about the noun pair being a hypernym.

Summary

Overall, the entire neural network has parameters $W_W(\cdot,\cdot,\cdot)$, $W_A(\cdot,\cdot,\cdot)$, $W_N(\cdot)$, W_2, W_4, b_1 and b_2. We use pre-trained word embeddings and do not modify them during learning. Figure 3 is a diagrammatic representation of the neural network.

6 Dataset

We use Simple Wikipedia as our text corpus, WordNet [8] to obtain hypernym and non-hypernym noun pairs and SENNA [4, 5] for the pre-trained word embeddings. These are described in turn.

Text corpus

Our text corpus is Simple Wikipedia, which is a version of Wikipedia where authors are encouraged to use simpler English, both in terms of grammar and vocabulary. The Simple Wikipedia dump[1] has Wikimedia style mark-up in its content. After extracting a plain text version from the dump[2] we use the Stanford CoreNLP library[3] [22,23] for lemmatization and part-of-speech tagging, and the Ensemble Malt parser for dependency parsing[4] [21]. All sentences with length greater than 100 tokens are ignored, since these are suspected to be errors made during extraction of plain text from the Simple Wikipedia dump.

[1] version dated Sep 2, 2013 available at `http://dumps.wikimedia.org/simplewiki/20130902/`
[2] using extractor available at `http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`
[3] available at `http://nlp.stanford.edu/software/corenlp.shtml`
[4] available at `http://www.surdeanu.info/mihai/ensemble/`

In the version we used, Simple Wikipedia had 96,244 articles with a total of roughly 892,000 sentences and an average of 17.1 tokens per sentence (including punctuation).

Hypernyms and non-hypernyms

We obtain our hypernyms and non-hypernyms from the WordNet lexical database. Although WordNet includes a lot of other information, for our application we only need synsets and hypernym relations. Synsets in WordNet represent cognitive synonyms, and hypernym relations exist between ordered pairs of synsets (p, c) denoting c 'is a' p. Since the hypernym relation is transitive, all the hypernym relations together form a directed acyclic graph. A noun pair (p, c) is given a hypernym label in our dataset if p is a strict ancestor of c as per the WordNet hypernym directed acyclic graph. Otherwise the noun pair is labeled a non-hypernym.

Since the text corpus is not connected to WordNet in any way, we need to identify synset mentions in the text corpus. We allow a noun to appear in a noun pair if the noun either has exactly one synset associated with it in WordNet, or if there are more than one synsets associated with it, but exactly one of the synsets has a non-zero frequency count in WordNet. Finally, a noun can appear in a noun pair only if we have an embedding available for the noun. How we obtain word embeddings is described at the end of this section.

For a noun pair to be considered as a candidate for being included in the dataset, each of its nouns must satisfy the criteria specified in the above paragraph and the noun pair must have at least one evidence (as defined in section 5.1) associated with it. Any hypernym noun pair that is a candidate is included in the dataset. A non-hypernym noun pair (p, c) is included in the dataset only if the noun p appears in some hypernym noun pair (p, x) and the noun c appears in some hypernym noun pair (x, c).

Finally, the dataset is split into train and test datasets. To do this, a fraction f of all the nouns that appear in the parent position of the noun pair are chosen at random to be test-only, and similarly for nouns in the child position. Any noun pair involving a test-only parent noun at the parent position or a test-only child noun at the child position is put in the test dataset, and the remaining noun pairs form the train dataset. We use this mechanism for splitting the dataset in order to have a large number of noun pairs in the test dataset with unseen nouns. Our mechanism for splitting guarantees that for the test noun pair, at least one noun is unseen. Due to this property, it is also impossible for a model to make predictions solely based on the transitivity of the hypernym relation and the labels of noun pairs in the train dataset. Table 1 gives some statistics on the dataset.

Dataset type	Number of noun pairs	Number of hypernym noun pairs	Percent of hypernym noun pairs
Entire dataset	52242	4439	8.50%
Train dataset	40008	3530	8.82%
Test dataset	12234	909	7.43%
Test dataset with only parent noun unseen	5965	342	5.73%
Test dataset with only child noun unseen	5425	475	8.76%
Test dataset with both nouns unseen	844	92	10.90%

Table 1: Dataset statistics

Word embeddings

We obtain the word embeddings from SENNA[5]. The word embeddings are learned via joint training on various natural language processing tasks such as language modeling, chunking, part-of-speech tagging, named entity recognition and semantic role labeling. For the language modeling task, the authors use Wikipedia as the text corpus. The embeddings have dimension $d = 50$.

When learning the word embeddings, the authors converted all words to lowercase and mapped all digits to zero. We do the same normalization when looking for the embedding of a particular word.

[5]available at `http://ml.nec-labs.com/senna/`

7 Experiments

In this section, we detail the parameters of our model and also add more details about training the neural network.

As mentioned in the previous section, the embeddings have dimension $d = 50$. We restrict the maximum length of the path P to be $L = 4$. The only other parameters involved in the architecture of the neural network are sz_1 and sz_2. These determine the sizes of the hidden layers and dimensions of the weight and bias parameters being learned. We use $sz_1 = 50$ and $sz_2 = 50$ in all of our experiments.

We initialize all the weights and biases by drawing them uniformly in the range $(-0.05, 0.05)$. The bias of the fourth layer b_4 is fixed at -3. We use a learning rate of 10^{-3}. Training is done using minibatched back-propagation [16]. We divided the train dataset into 50 minibatches each with roughly 800 noun pairs out of which approximately 70 were hypernym noun pairs.

For our update rule, we use the diagonal variant of AdaGrad [7], i.e. we keep track of the mean squared gradient for each weight and bias parameter, and divide the gradient by the root mean squared gradient when updating the parameter. Mathematically, the equations for updating a weight w with

mean squared gradient u are given by,

$$u_{i+1} := \alpha \cdot u_i + (1 - \alpha) \cdot \left\langle \frac{\partial C}{\partial w}|_{w_i} \right\rangle^2_{D_i}$$

$$w_{i+1} := w_i - \epsilon \cdot \frac{\left\langle \frac{\partial C}{\partial w}|_{w_i} \right\rangle_{D_i}}{\sqrt{u_{i+1}}}$$

where i is the time step, ϵ is the learning rate, and $\left\langle \frac{\partial C}{\partial w}|_{w_i} \right\rangle_{D_i}$ is the partial derivative of the cost C with respect to w evaluated at w_i for the minibatch D_i. We set $\alpha = 0.9$.

It took about 20 epochs for the network to train before it started overfitting. The training time was about half a day for our implementation in R programming language using 4 cores on a 3.2GHz machine. There is significant scope for optimization both within R or by moving to a different programming language. Our implementation was not optimized.

For our results, we use maximum classification f-score on the test dataset where the maximization is over all possible thresholds.

8 Results

Table 2 includes the results from our experiments. For comparison, we implemented a baseline model. The baseline model gets as input only the embeddings of the nouns, and does not use the corpus at all. The baseline model is a simple logistic regression model. We get two different sets of baseline results. One when the model only gets as input the embeddings of

the noun words and a bias term, i.e. $[emb(p); emb(c); 1]$. And second when the model also gets as input the second order features from $emb_2(p, c)$, i.e. input is $[emb_2(p, c); emb(p); emb(c); 1]$.

	f-score on all noun pairs	f-score on noun pairs with both nouns unseen	f-score on noun pairs with parent noun unseen	f-score on noun pairs with child noun unseen
Baseline (only first order noun features)	36.93	43.44	28.35	43.38
Baseline (all noun features)	53.30	54.73	39.05	65.99
Our model (no noun features)	41.99	42.04	37.82	45.75
Our model (only first order noun features)	49.27	49.20	43.11	55.69
Our model (all noun features)	58.70	54.35	46.70	68.50

Table 2: F-score on the test dataset using the baseline model and the neural network model.

We run three different variations of our model. In the first version, we do not use any noun features and the model relies solely on the text corpus. This changes the equations for g_W and g_N in the first layer of our neural network to be

$$g_W(E) = \sum_{w \in E \setminus \{p,c\}} W_W(len(E), loc(w), POS(w)) \cdot emb(w)$$

$$g_N(E) = 0$$

In the second version, we use only first order noun features, i.e. $g_N(E) = 0$ and everything else is unchanged. The third version has all equations as described in section 5.2.

Parent noun	Child noun
invertebrate	nestling
metropolis	euphrates
geographer	hipparchus
crocodilian	tuatara
invertebrate	bullfrog
warship	warship
airliner	airliner
planetoid	perihelion
rifle	howitzer
warship	speedboat
highwayman	highwayman
judaism	anti-semitism
invertebrate	hawksbill
canoe	betelgeuse
utensil	jug
organism	chert
highway	underpass
geographer	guyot
artefact	burial
coin	centavo

Parent noun	Child noun
scientist	delius
leader	himmler
steroid	mifepristone
metal	satin
novelist	sartre
drug	nicotine
meat	meatball
illness	insomnia
explosive	nitroglycerin
mathematician	bose
measles	chickenpox
infection	lymphoma
fungus	bullfrog
disease	adenoma
pup	meerkat
airport	airfield
district	muslin
food	bistro
chemical	thc
human	midget

Table 3: Non-hypernym noun pairs which were given the highest hypernym score by the baseline model (left) and by the neural network model (right).

Table 3 lists for both the baseline model and our neural network model their top 20 false positives, i.e. non-hypernym noun pairs which were given highest hypernym score by each model. The results are from the models that get all noun embeddings features as input including the second-order features, i.e. second version for the baseline model and third version for the neural network model.

9 Discussion

In table 5 and table 6 we attempt to categorize the errors made by the baseline model and the neural network model shown in table 3. The categories of errors for the baseline model are *dataset errors*, *related noun errors*, *synonym errors* and *miscellaneous errors*. The categories of errors for the neural network model are *dataset errors*, *parse errors*, *entity errors* and *miscellaneous errors*. We describe and discuss each of the categories of errors in this section, starting with dataset errors, then analyzing the errors made by the baseline model, and finally analyzing the errors made by the neural network model. The objective is to highlight the kind of errors made by the neural network model, how they differ from the baseline model, and to suggest future work to remedy such errors.

9.1 Dataset errors

In section 6 we describe the process by which we obtain hypernym and non-hypernym labels for each noun pair. In particular, notice that WordNet only includes hypernym relations connecting pairs of synsets, and it does not include any explicit non-hypernym relations. Hence, we assume that any two synsets which are not related to each other explicitly by the hypernym relation (direct or inherited) are non-hypernyms.

This assumption fails to hold in quite a number of cases. In table 5 and table 6 we see that 3 out of the top 20 false positives of the baseline model, and 9 out of the top 20 false positives of the neural network model are actu-

Parent noun	Child noun	Sentence in Simple Wikipedia
leader	himmler	Heinrich Himmler (7 October 1900 – 23 May 1945) was the leader of Germany's SS and Gestapo organisation.
steroid	mifepristone	Mifepristone is a synthetic steroid that is used as a drug.
novelist	sartre	Jean-Paul Charles Aymard Sartre (21 June 1905 – 15 April 1980) was a French existentialist philosopher, novelist, playwright, screenwriter, and critic.
drug	nicotine	Nicotine is a drug found in tobacco cigarettes, cigars, pipe tobacco, and chewing tobacco.
illness	insomnia	It can help with about 81 different illnesses including cancer, bronchitis, insomnia, edema, colds, etc..
explosive	nitroglycerin	Mercury fulminate, picric acid, lead azide, nitroglycerine and iodine nitride are examples of primary explosives.
mathematician	bose	Professor Satyendra Nath Bose (1 January 1894 – 4 February 1974) was an Indian mathematician and physicist.
chemical	thc	The cannabis plant's flowers contain a chemical or drug known as THC.
human	midget	Midgets are perfectly proportioned humans while dwarves have a large head and misshapen limbs and torsos.

Table 4: The noun pairs in table 6 for which the neural network model gets support from Simple Wikipedia, but WordNet does not include the hypernym relation.

ally hypernym noun pairs. Table 4 lists the sentences in Simple Wikipedia which contain evidence about the noun pairs being hypernyms.

9.2 Baseline model errors

Apart from the dataset errors, a lot of the errors made by the baseline model are of the form where the parent and child nouns are semantically close to each other, but are not related by the hypernym relation.

Parent noun	Child noun	Category
invertebrate	nestling	related noun error
metropolis	euphrates	related noun error
geographer	hipparchus	dataset error
crocodilian	tuatara	related noun error
invertebrate	bullfrog	related noun error
warship	warship	synonym error
airliner	airliner	synonym error
planetoid	perihelion	related noun error
rifle	howitzer	related noun error
warship	speedboat	related noun error
highwayman	highwayman	synonym error
judaism	anti-semitism	related noun error
invertebrate	hawksbill	related noun error
canoe	betelgeuse	misc. error
utensil	jug	dataset error
organism	chert	misc. error
highway	underpass	related noun error
geographer	guyot	dataset error
artefact	burial	misc. error
coin	centavo	related noun error

Table 5: Categorization of the top 20 false positives of the baseline model.

For example, *howitzer* is a cannon, not a *rifle*, and *hawksbill* is a turtle (and hence a vertebrate), not an *invertebrate*. *tuatara* and *crocodilian* are both diapsid reptiles, and *perihelion* and *planetoid* are both concepts related to the solar system. The algorithm also makes errors for the special cases where the parent noun is the same as the child noun.

9.3 Neural network model errors

There are two noticeable patterns of errors made by the neural network model beyond dataset errors – namely, parse errors and entity errors. We discuss these one by one.

Parent noun	Child noun	Category
scientist	delius	entity error
leader	himmler	dataset error
steroid	mifepristone	dataset error
metal	satin	parse error
novelist	sartre	dataset error
drug	nicotine	dataset error
meat	meatball	misc. error
illness	insomnia	dataset error
explosive	nitroglycerin	dataset error
mathematician	bose	dataset error
measles	chickenpox	parse error
infection	lymphoma	misc. error
fungus	bullfrog	misc. error
disease	adenoma	parse error
pup	meerkat	entity error
airport	airfield	entity error
district	muslin	parse error
food	bistro	misc. error
chemical	thc	dataset error
human	midget	dataset error

Table 6: Categorization of the top 20 false positives of the neural network model.

The neural network model gets as input evidence derived from the dependency parse structure of the sentences in the text corpus. However, for 4 out of the top 20 false positives, the dependency parser gives inaccurate output. Given the specific errors in dependency parse structure (see figure 4), it makes sense that the neural network model predicts the noun pairs to be hypernyms. Figure 4 shows the correct and incorrect parses of two Simple Wikipedia sentences which resulted in false positives.

Lastly, we have entity errors caused by the presence of synsets whose mentions in the text corpus span multiple words. Because of the way we con-

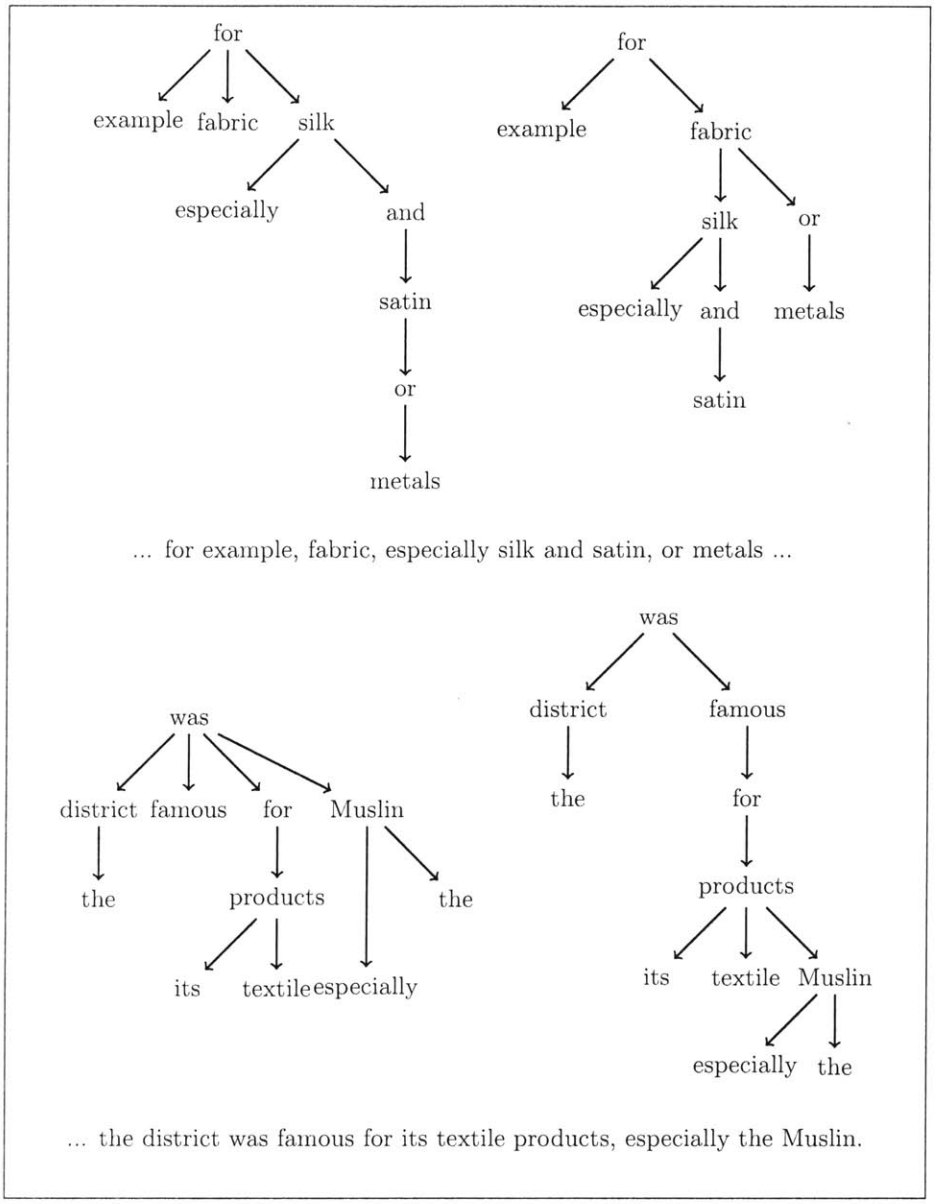


Figure 4: Incorrect (left) and correct (right) dependency parse trees (without conjunction links) for 2 out of the 4 noun pairs listed in table 6 whose sentences are parsed incorrectly by the dependency parser.

struct each evidence, we handle entity mentions spanning multiple words particularly badly. The neural network model assumes that the evidence is about the single word noun entity, but the sentence is actually referring to an entity spanning multiple words. Table 7 lists sentences from Simple Wikipedia for which the neural network model makes entity errors. In the first example, Wikipedia is referring to political scientist Christina Delius, whereas WordNet resolves Delius to composer Frederick Delius. The second sentence is about baby meerkats, not meerkats and the third sentence is about the Los Alamitos Army Airfield, not an airfield.

Parent noun	Child noun	Sentence in Simple Wikipedia
scientist	delius	On 9 August 1982 he married the political scientist Christina Delius (born 1956).
pup	meerkat	Baby meerkats, called "pups" are sometimes also eaten by snakes.
airport	airfield	Los Alamitos Army Airfield is a military airport.

Table 7: Noun pairs in table 6 for which the neural network model makes mistakes due to entity mentions spanning multiple words.

10 Future work

In this section, we list some possible directions for future work based on the analysis of errors in the previous section. First, we explore possible remedies for dataset errors. Thereafter, we propose a recursive neural network model for the task, and give some intuition about how recursive neural network models might deal with parse errors and entity errors better than the model

we implemented.

10.1 Dataset errors

In the previous section, we noted that our method for inferring hypernym and non-hypernym labels for the noun pairs was not robust. As such, it is desirable to have a dataset available to us which includes explicit non-hypernyms. The process of creating this dataset need not be entirely manual in that predictions made by the model presented in this paper could be used as a guide in terms of which noun pairs should have their labels verified. After correcting a significant number of incorrectly labeled noun pairs, the model could be trained again on the new dataset, and the process could be repeated.

10.2 Recursive neural network

We will first describe the recursive neural network (RNN) model by Socher et. al [18, 19] used for parsing. Thereafter, we will explain how the RNN model can be used for hypernym classification, and give intuition about how it could remedy entity and parse errors.

A RNN model comprises of a neural network which takes as input features from two embeddings, and gives as output one embedding and a scalar score. The output embedding has the same dimension as each of the input embeddings. While parsing a sentence, the neural network is given as input features from the embeddings of each adjacent pair of words in the sentence. The pair of words with the highest output score are combined into a phrase and

the embedding output by the neural network is taken to be the embedding for the phrase. Since the phrase embedding has the same dimension as the original word embeddings, the above process can be repeated for adjacent words and phrases to determine which word or phrase should be combined next. This process is applied recursively until exactly one embedding is left representing the entire sentence. The order in which the words and phrases were combined give the parse structure for the sentence.

Now, for the task of hypernym classification, we could use the RNN trained on parsing to get the embeddings for the two noun phrases occurring in a sentence and the phrase separating the two noun phrases. We could then use these three embeddings to predict whether or not the first noun phrase is a hypernym of the second noun phrase. If there are multiple sentences in which the same two noun phrases occur, then we could use a max-pooling method similar to ours to combine information over multiple evidences. Since the method automatically handles noun phrases, this should reduce entity errors.

Finally, parse errors could be handled by doing joint training of the RNN on hypernym classification and parsing. We believe that this would reduce parse errors since it seems that to correct the parse errors such as those shown in figure 4, the parsing model needs semantic information present in the embeddings. This semantic information could be learned while jointly training on the hypernym classification task.

11 Conclusion

We started out with the motivation of automatic learning of semantic databases from unstructured text, and narrowed the problem to that of predicting hypernym relations. We learned a neural network model for predicting, given a text corpus, whether or not two nouns are related by the hypernym relation. We analyzed the errors made by the model and saw three patterns, namely dataset errors, parse errors and entity errors. Specifically, we showed that a significant fraction of the 'errors' made by the model were actually errors in the dataset itself. Lastly, we proposed possible future work to remedy each kind of error.

References

[1] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003.

[2] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, New York, NY, USA, 2008. ACM.

[3] Danqi Chen, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *CoRR*, abs/1301.3618, 2013.

[4] R. Collobert. Deep learning for efficient discriminative parsing. In *AISTATS*, 2011.

[5] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

[6] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM.

[7] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011.

[8] Christiane Fellbaum, editor. *WordNet: an electronic lexical database*. MIT Press, 1998.

[9] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79, 2010.

[10] C. Havasi, R. Speer, and J. Alonso. Conceptnet 3: a flexible, multilingual semantic network for common sense knowledge. In *Recent Advances in Natural Language Processing*, Borovets, Bulgaria, September 2007.

[11] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[12] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *NIPS*, pages 1106–1114, 2012.

[14] Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751. The Association for Computational Linguistics, 2013.

[15] Andriy Mnih and Geoffrey E. Hinton. A scalable hierarchical distributed language model. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Lon Bottou, editors, *NIPS*, pages 1081–1088. Curran Associates, Inc., 2008.

[16] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Neurocomputing: Foundations of research. chapter Learning Representations by Back-propagating Errors, pages 696–699. MIT Press, Cambridge, MA, USA, 1988.

[17] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Advances in Neural Information Processing Systems (NIPS 2004)*, November 2004. This is a draft version from the NIPS preproceedings; the final version will be published by April 2005.

[18] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing With Compositional Vector Grammars. In *ACL*. 2013.

[19] Richard Socher, Christopher D. Manning, and Andrew Y. Ng. Learning continuous phrase representations and syntactic parsing with recursive neural networks, 2010.

[20] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[21] Mihai Surdeanu and Christopher D. Manning. Ensemble models for dependency parsing: Cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 649–652, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[22] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 173–180, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[23] Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *In Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 63–70, 2000.

[24] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 384–394, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.