Saarland University
Language Science and Technology
Master's Thesis
Prof. Dr. Dietrich Klakow
Prof. Dr. Iryna Gurevych

Relation Extraction Using Liberalism love and beloved

Ehsan Khoddammohammadi

Acknowledgements

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

Saarbrücken, 4th December 2013

Ehsan Khoddammohammadi

Abstract

Keywords: Blah, Blah

Contents

1 Introduction

Nowadays searching through Internet is the first step we take if we want to get answer to our question. We convert our questions to sequences of keywords and search engines are trying to lead us to web pages where might contain the answer to our questions, they return us a set of related documents which are sorted by their popularity and similarity of their text to our query. In this way, users themselves are responsible to find the desired knowledge from documents. The next generation search engines should go further than current approaches in understanding the meaning of a query and the underlying semantics of documents on the web. The next natural improvement is to return a piece of information which directly approaches to answer the user question. For this reason, the elements of a query, concepts or entities and their relations should be identified and documents (which might have the answer) should be mapped to the same space of entities and relations in order to find the the desired information in the question.

Relation Extraction is the task of detecting and classifying semantic relationship between named entities (NE). The goal of this task is to find a triple of binary relations and their arguments [2]. For instance, we want to induce a relation like *bornIn* with its arguments which could be for example like this: *(bornIn, Beata Nyari, Budapest)*. Applications of such task is numerous in natural language processing; Question/Answering, machine translation and text summarization are systems that benefit from relation extraction [2].

In this task, we are trying to have a model to find paraphrases which means that we are interested to have all the similar semantically similar relations under one umbrella. So it is desired to have all different surface realizations of one relation like *isGivenBirth, isBorn, isFrom* in a same set, namely *bornIn*.

From a classic method, DIRT [24], to very famous frameworks , TextRunner[4] or distant supervision[27], and more recent works e.g. PATTY[30], all are examples of several

different family of approaches. These methods could be categorized from different perspectives (1) amount of annotated data they need (2) if they can only handle a predefined enumeration of entities and re- lations or are open to any number of relations (3) organization of semantic interpretation and (4) underlying family of methods they use.

In this proposal, we suggest a new method which learns informative features from data using deep learning methods and then use the learnt features to develop a relation extraction system in two schema (1) Weakly supervised and (2) Unsupervised.

In next chapters we review recent or influential related works and finally we will describe our new model as an improvement to current approaches.

2 Machine Learning

Machine learning is playing a fundamental role in natural language processing and computational linguistics. The impact is so significant that one of key note speakers of ACL 2012, Mark Johnson, predicted that in 50 years from now NLP/CL will not exist as a research field and will be emerged as two different fields:(1) machine learning and (2) logic. This chapter is dedicated to subset of models in machine learning that can play a role of bridge between logic and machine learning. This chapter provides the building material for a new direction of research so called *Representation Learning* which enables us to transfer information from logical forms such as predicate-argument form to vector space (beside many other advantages which we will discuss). Most of current models in machine learning are designed to work in vector space therefore it is an important achievement to be able to work with type of information which is in other forms or representations by first transferring them to vector space. Working with logical forms in vector space helps us to use many available knowledge bases and lexicons with available machine learning models which is crucial for relation discovery. I will skip discussing classical machine learning problems to emphasis more on new relevant topics. I will first briefly review artificial neural networks and then motivate and define the task of representation learning. In next chapters, we will need the materials provided in this chapter to understand the undergoing research about relation discovery.

2.1 Artificial Neural Networks

Before diving into different methods of representation learning, it would be useful to briefly review the architecture, applications and learning algorithms of artificial neural networks (ANN). ANNs can be used for variety of applications in machine learning: classification, clustering, dimensionality reduction,... . What makes ANN important for us is their ability to do many of mentioned tasks jointly. Its layer-wised architecture very

well fits to the idea of multi-task learning. Different layers of an ANN can have different objectives and can share information with other layers through parameter sharing. First I discuss about elements and architecture of ANNs and then we will very briefly review *backpropagation* and *stochastic gradient descent* as we need them to understand the mechanism of learning in ANNs.

Artificial neural networks have old history in the domain of machine learning. They are inspired from the mechanism humans learn via their neural networks. The basic element of this network is a neuron which takes a weighted input signals and by applying an activation function on this input, the neuron will output a signal. Neurons are usually arranged in layers and receive input from the neurons from the previous layer and send their outputs to the next layer. The combination of different layers and using different activation function with different degree of non-linearity is enabling ANNs to learn non-linear functions. A common mathematical model for neuron in ANNs is shown in (2.1) and visually in Figure 2.1.

$$\begin{aligned} x^{l-1} &: \text{input vector of size } n \text{ and layer } l-1 \\ w^{l,l-1} &: \text{weight vector of size } n \text{ from layer } l-1 \text{ to layer } l \\ y_j^l &: \text{output value of neuron } j \text{ at layer } l \\ y_j^l &= f(\sum_{i=0}^{n} w_{i,j} x_i^{l-1}) \end{aligned} \tag{2.1}$$

Activation functions can be usually either of functions in the list below:

linear function $f(x) = ax + b$ where $a, b \in \mathbb{R}$

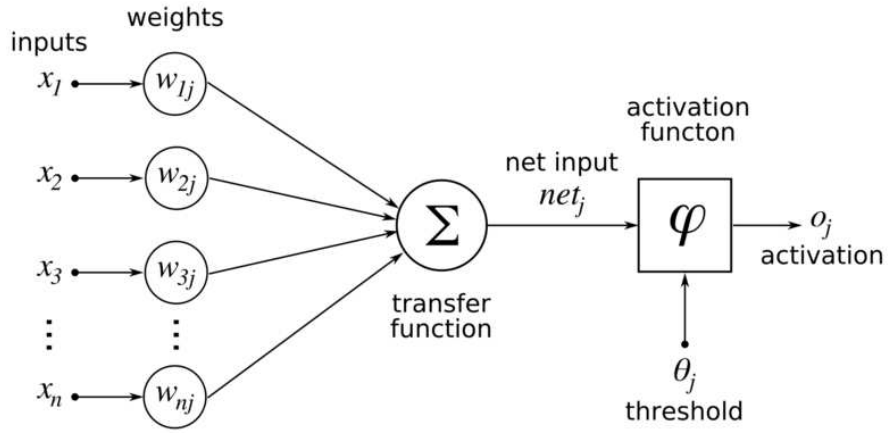
step function $f(x) = 1$ if $x > \theta$ else $f(x) = 0$ where $\theta \in \mathbb{R}$ is a constant threshold

tangent hyperbolic $f(x) = tanh(x)$

Log-sigmoid $f(x) = \frac{1}{1+e^{-x}}$

The type of ANNs matters mostly for us in this research are networks which neurons are arranged in ordered layers and there is no feedback from a deeper layer to the layers in the back. The first layer is usually for unsupervised pre-training of objects, then a layer for adding non-linearity comes and finally we have an axillary classification or ranking task. In Figure 2.2 we can see a neural architecture proposed by Bengio et al.

Figure 2.1: A model of neuron



[5] which works as language model. This model takes n-grams as input and predict the probability of the next word. The first layer is hot-one representation of sequence words, $w_{t-n+1}, \ldots, w_{t-2}, w_{t-1}$ and then the layer for unsupervised pre-training of words comes. Learned features of words will be combined together with *tanh* layer and finally we will have a *softmax* layer which outputs probability of all words as the next word, w_t, in the sequence. Later we will see more details about different layers of this ANN but the main question that arises here is how to learn parameters of such a network in order to make good predictions and induce meaningful features for words.

There are two main approaches for learning parameters of a single supervised layer. By supervised layer we mean that true outputs/predictions are available for training the parameter of the layer. :

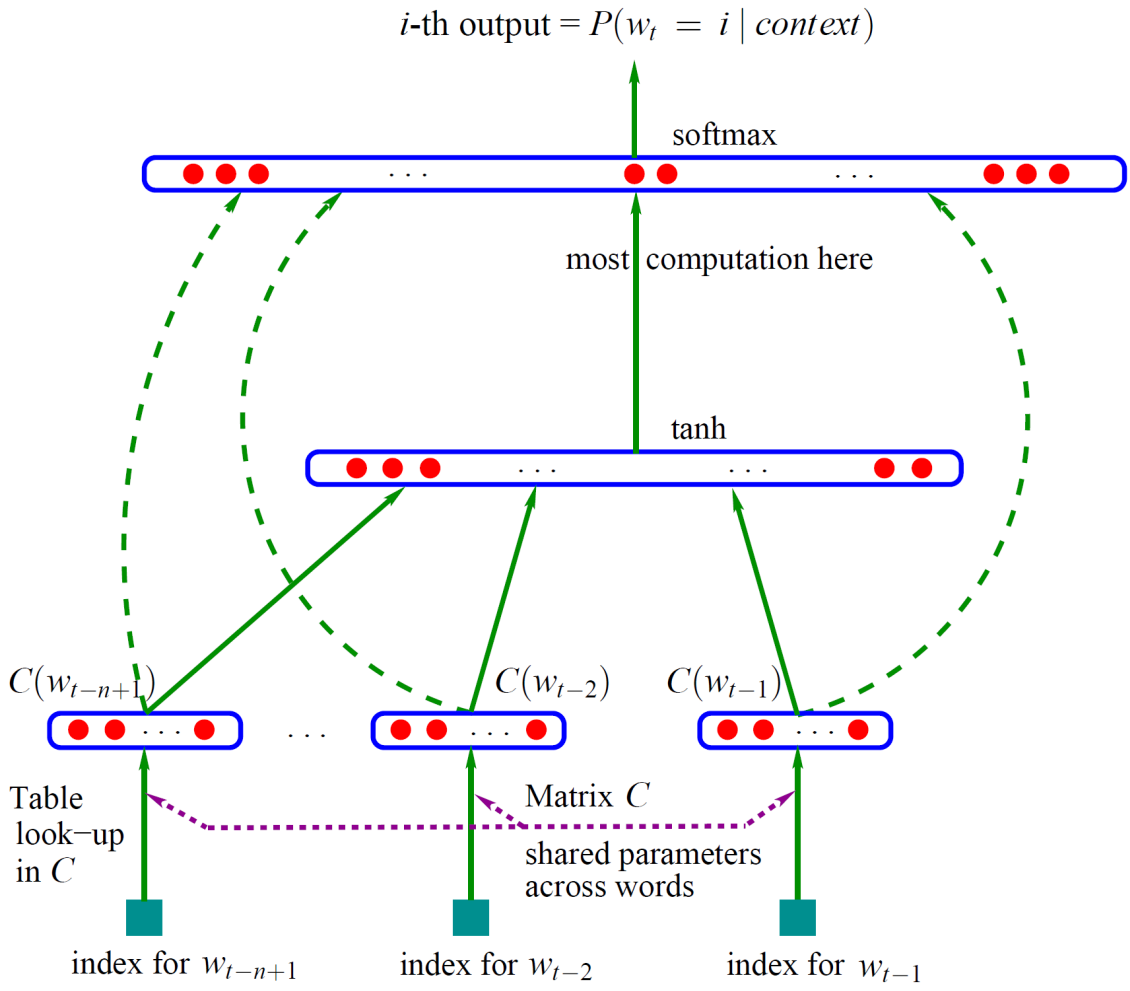
Online learning

To optimize the parameters, we iterate over all points in the training dataset and calculate the layer predictions for input. In each iteration, we update the weights in order to decrease the difference between the predicted output and the true output for each point. This forms an objective function (error function) which can be minimized by gradient descent, by calculating error derivatives and update weights to decrease them. Since each data point can change the parameters, we call this algorithm an online learning algorithm.

Batch learning

In this approach, unlike the previous approach, first we calculate the predicted out-

Figure 2.2: A Neural Language Model proposed by Bengio et al.



puts of whole dataset and form the error function and try to minimize the error derivative for whole dataset. Since we update the parameters after observing all points in the training dataset we call this approach batch learning.

Stochastic Gradient Descent

The idea of SGD is to make best out of both previous ideas. Neither updating after each datapoint nor after all points, SGD makes small batches in each iteration, each usually contains only 100 datapoints, and calculate error derivatives and update parameters for each batch.

Since we now know how to train a single supervised layer, other layers of a network can be trained using *backpropagation*. Parameters of the network will be randomly initialized and then by choosing one of the approaches above we can start by learning the last layer, since we have the output for that. After updating the last layer, it works as the output of the layer before it and we can use the same strategy to train its parameters. Likewise, we continue training of other layers. The criteria for stopping the algorithm can be either maximum number of iterations or when parameters don't change from after some iterations.

After introducing ANNs we will focus more on the idea of unsupervised pre-training in the next section. I show the generalization of this idea using other methods and motivates it as a very crucial step in many NLP tasks.

2.2 Representation Learning

In this chapter, we will define and justify the task of *Representation Learning* and we will see different families of methods for inducing word representation and its application in NLP.

In machine learning specially in industry, most of the labor is dedicated to *Feature Engineering*. Extracting informative features is the crucial part of most supervised methods and it is done mostly manually. While many different applications share common learning models and classifiers, the difference in performance of competing methods mostly goes to the data representation and hand-crafted features that they use. This observation reveals an important weakness in current models, namely their inability to extract and organize discriminative features from data. Representation learning is an umbrella term for a family of unsupervised methods to learn features from data. Most of recent works on the application of this idea in NLP focus on inducing word representations. *Word representation* is a mathematical object, usually a vector, which each dimension in this vector represents a grammatical or semantical feature to identify this word and is induced automatically from data [42]. Recently, it has been shown in [42] and [16] that using induced features can be helpful to improve state-of-the-art methods in different NLP tasks. It seems that relation extraction can also benefit from such features since similar tasks like semantic role labeling has been shown to benefit from induced word representations. In chapter 4 and chapter 5 I will show two applications of it, (1) direct usage for relation

discovery and (2) word feature generation which can be used in current relation discovery systems as well as other NLP applications. In the next two sections, two major families of representation learning methods will be shortly reviewed.

2.2.1 Distributional Representation

In distributional semantics, the meaning of a word is expressed by the context that it appears in it [20]. Features that are used to represent the meaning of a word are other words in its neighborhood as it is so called the context. In some approaches like LDA and latent semantic analysis (LSA), the context is defined in the scope of a document rather than a window around a word. To represent word meanings in via distributional approach, one should start from count matrix (or zero-one co-occurrence matrix) which each row represents a word and each column is a context. The representation can be limited to raw usage of the very same matrix or some transforms like *tf-idf* will be applied first. A further analysis over this matrix to extract more meaningful features is applying dimensionality reduction methods or clustering models to induce latent distributional representations. A similar clustering method to k-means is used in [23] to represent phrase and word meanings and brown clustering algorithm [14] has been shown to have impact on near to state-of-the-art NLP tasks [42].

2.2.2 Distributed Representation

Distributed representation has been introduced in the literature for the first time in [5] where Bengio et al. introduced a first language model based on deep learning methods[6]. Deep learning is learning through several layers of neural networks which each layer is responsible to learn a different concept and each concept is built over other more abstract concepts. In the deep learning society, any word representation that is induced with a neural network is called *Word Embedding*. In contrast to raw count matrix in distributional representations, word embeddings are low-dimensional, dense and real-valued vectors. The term, **'Distributed'**, in this context refers to the fact that exponential number of objects (clusters) can be modeled by word embeddings. Here we will see two famous models to induce for such representations. One family will use n-grams to learn word representation jointly with a language model and the other family learns the embedding from structured resources. In [15], Weston and Collobert use a non-probabilistic and

discriminative model to jointly learn word embeddings and a language model that can separate plausible n-grams from noisy ones. For each word in a n-gram, they combine the word embeddings and use it as positive example. They put noise in the n-gram to make negative examples and then train a neural network to learn to classify positive labels from negative ones. The parameters of neural network (neural language model) and word embedding values will be learned jointly by an optimization method called *Stochastic Gradient Descent* [13].

A hierarchical distributed language model (HLBL) proposed by Mnih and Hinton in [28] is another influential work on word embeddings. In this model a probabilistic linear neural network(LBL) will be trained to combine word embeddings in first $n-1$ words of a n-gram to predict the n_{th} word.

Weston-Collobert model and HLBL by Mnih and Hinton are evaluated in [42] in two NLP tasks: chunking and named entity recognition. With using word embeddings from these models combined with hand-crafted features, the performance of both tasks are shown to be improved.

2.2.3 Representation Learning of Knowledge Bases

Bordes et al. in [12] and [11] have attempted to use a neural distributed model to induce word representations from lexical resources such as WordNet [18] and knowledge bases (KB) like Freebase [10] . In Freebase for example, each named entity is related to another entity by an instance of a specific type of relation. In [12], each entity is represented as a vector and each relation is decomposed to two matrices. Each of these matrices transform left and right-hand-side entities to a semantic space. Similarity of transformed entities indicates that the relation holds between the entities. A prediction task is defined to evaluate the embeddings. Given a relation and one of the entities, the task is to predict the missing entity. The high accuracy (99.2%) of the model on prediciton of training data shows that learned representation highly captures attributes of the entities and relations in Freebase.

Two major models are proposed in [12] and [11] to learn features in continuous vector space from a Knowledge Bases(KB) which information is usually represented in form of triples of (e_i, r_k, e_j) where e_i and e_j are i_{th} and j_{th} entities related by a binary relation of type r_k. The purpose of the models is to induce a vector space and associate each entity

or relation to an embedding vector or a matrix. The dimensions of such an embedding vector are supposed to reflect a set of informative features of entities and relations.

In the first model, **structured embeddings(SE)**, entities are modeled as d-dimensional vectors. An associated vector to the i_{th} entity, e_i, is $E_i \in \mathbb{R}^d$. Each relation r_k is decomposed to two operators each represented as $d \times d$ matrix, $R_k = (R_k^{left}, R_k^{right})$. These operators transform the left and right entities to a new space induced by each relation and by using a p-norm measure (L1 norm in this work) they associate a similarity value or a score to each triple. This similarity value is being calculated by Equation (2.2).

$$Sim(E_i, E_j, R) = ||R_k^{left} E_i - R_k^{right} E_j||_1 \tag{2.2}$$

The similarity between transformed entities works as a score to measure the strength of a relation holds between two entities.

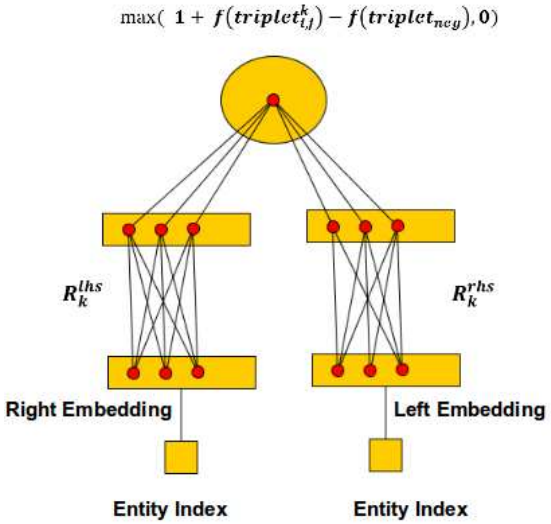
Using the idea of contrastive learning , the model will be trained to increase similarity of embeddings for a positive triple (a triple which exists in the KB) or lowering its rank among other training samples and decrease the similarity of embeddings when the relation doesn't hold (negative triple) or raising its rank . For each positive triple, two negative triples will be generated by randomly alternating the right entity or left entity with other entities. Inspired from large margin methods a constraint is introduced on the model that forces negative triples to have lower associated similarity value than correspondent positive triples by a large margin. In Figure 2.3 a schematic view of model is presented.

The second model, **Semantic Matching Energy using Bilinear layers(SME-Bil)**, is using a different representation for relations,weighted bilinear transformation of embeddings and dot product similarity function instead of L1 norm. In this model, each relation is represented by a d-dimensional vector R_k same as entities. For triple (e_i, r_k, e_j) , the model combines the weighted transformation of each entity embedding with the weighted embedding of relation using element-wise vector product. as it is shown in Equation (2.3).

$$E'_{left} = (W_i E_i) \odot (W_k R_k) + b_{left} \tag{2.3}$$

W_i and W_k are $d \times d$ weight matrices and b_{left} is a d-dimensional bias vector. The same equation holds for transforming the right entity embeddings to E'_{right}. Finally, the associated score for the triple can be calculated by dot product of E'_{left} and E'_{right} which is shown in Equation 2.4. Figure 2.4 is used by Bordes et al. to sketch SME-Bil model.

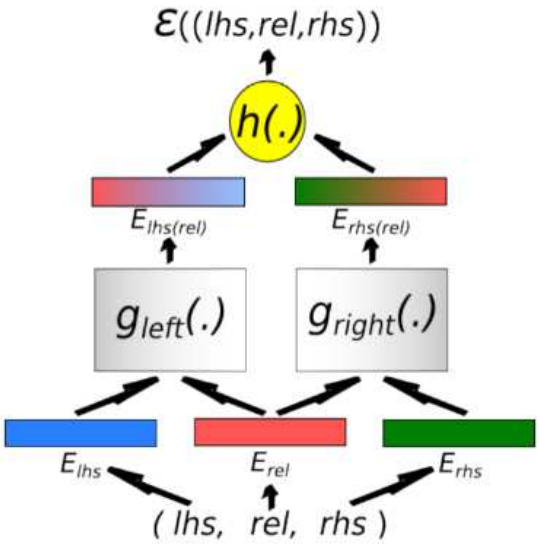
Figure 2.3: Neural Distributed Model to Learn Structured Embeddings (SE)



Similar constraints to the first model are also applied to this model and both models can be trained by stochastic gradient descent (SGD) which we introduced in section 2.1.

$$Sim(E_i, E_j, R_k) = -E'_{left} E'_{right} \tag{2.4}$$

Figure 2.4: Neural Distributed Model to Learn Structured Embeddings (SME)



3 Relation Discovery

3.1 Distant Supervision

Labeling data with entity-relation annotation is tedious, specially in large scale domains such as web. Mintz et al. [27] used the idea of *distant supervision* and combined it with a large knowledge base, Freebase, to extract relations from text. They assumed that for each pair of entity related to each other in Freebase, any piece of text contained both of the entities is most probably expressing the same relation. Therefore, they started to make an automatically annotated dataset of relations. From around 1.2 million Wikipedia, they collected contexts that a pair of related entities have been mentioned in them. This dataset of 1.8 million instances of 102 relation type is considered as positive labels. To learn a classifier, they also needed negative labels which were created with random sampling of entities that don't happened to be related in Freebase. The number of negative examples should be at least twice a number of positive examples. After this phase, a multi-class logistic regression classifier was trained to extract relation (separate sentences which express a relation from those which do not).

A variety of features were extracted to make the dataset and each type of features is shown to have different impact in final performance of the model. For example it is mentioned in [27] that the method is benefiting a lot from syntactic features specially in ambiguous relations or where the entities are near in the dependency tree but far in the sentences. Word representations where we will be covered in chapter 2.2, are capable of abstracting and learning both semantic and syntactic features from various resources. This will suggest us to use word embeddings combined with authors' hand-crafted features with the hope of improving their results. This idea will be elaborated more in section 5.

3.2 Unsupervised Relation Extraction

In this section we will review four major works on unsupervised relation discovery. I will discuss their formulation of the task, their models and the features they incorporate, the model constraints and finally we will take a look at how good a model is performing.

3.2.1 DIRT

Lin and Pantel in [24] proposed an unsupervised method for finding paraphrases from text which proved to be very influential and their method, **DIRT**, has become a classic work in literature. In addition to this paper we will also review two major improvements to DIRT proposed in [8] and [32]. DIRT tries to cluster semantically equivalent relations based on the similarity of their arguments. Relations (or inference rules) that DIRT can discover are mostly limited to paraphrases which are a subset of possible types of inference rules.

In this section, first, the key idea of DIRT will be elaborated more and then we will review the algorithm and analyze the observations. Based on these observations, two other methods, ISP [32] and LEDIR [8], which try to address a subset of DIRT's problems, will be discussed.

3.2.1.1 Extended Distributional Hypothesis

Synonymy of words and other semantic relations in word level have been studied very well in literature. For example Pereira et al. in [33] cluster similar words which convey a similar meaning. The key assumption in this work and related ones is so called **Distributional Hypothesis** [20] . It tells that words which usually occur in similar context have similar meaning. This idea can be generalized to phrases or predicates-arguments. A set of relations or phrases that appear in a similar context are semantically equivalent. Lin and Pantel have extended this idea with giving a slightly different notion of context. The context that they define in their work is the dependency path between a predicate of relation and its arguments. As the Extended Distributional Hypothesis states: If two paths tend to occur in similar contexts, the meaning of the path tend to be similar

In this sense, the task of finding paraphrases or semantically equivalent relations can be formulated as *finding paths with similar meaning*. An algorithm to find such a similar path is a topic of the next part.

3.2.1.2 Model Description

DIRT starts with dependency parsing of sentences and continues with pruning of these dependency trees. It amounts to several conditions on dependency path such as that only nouns will be kept as arguments. Additionally, all the function words will be filtered to have the dependency relations that only connect two content words. Dependency relations with less than a certain number of occurrence will be removed to make the task feasible. After the pruning phase, a set of binary relations with a list of nouns associated to each of their slots is generated. The similarity of relations can be expressed based on similarity of associated list of slot words. Lin and Pantel have used *pointwise mutual information (pmi)* two measure similarity of slots. It measures the independency of two random variables or the amount of information they contain for each other. The similarity function for relations is a geometric average of similarity of correspondent slots.

Instead of clustering of relations which seems to be a natural next phase, they maintain a databases of triples (relation and its slots).For each relation queried they measure the similarity of relations which share at least one common slot word and then report top-40 of them.

3.2.1.3 Evaluation and Analysis

Based on what Lin and Pantel have reported in [24] they extracted 231,000 unique paths from a newspaper corpus. They have used a manually generated paraphrases to evaluate their method. For six questions from TREC-8 they have generated top-40 paraphrases and then evaluate them manually to report percentage of correctly found paraphrases. This approach has a very obvious flaw which is that number of returned paraphrases is fixed. Clustering has this advantage that each cluster of relations can have arbitrary number of paraphrases. Average accuracy for these six questions is 50.3% and no paraphrases have been found for one of the questions. The recall can't be measured easily since many correctly found paraphrases by DIRT are not discovered by humans.

Beside the problem of having fixed number of returned paraphrases, there are three other major problems after observing the results of DIRT which two of them will be addressed by Pantel et al. and will be discussed in the next parts. These problems are:

1. Antonymy of relations can not be captured with DIRT and antonym relations will be returned as similar relations. After more than a decade, it is still an open problem
2. DIRT results are very noisy in a sense that it doesn't induce the type of arguments.
3. DIRT results or paraphrases are always considered as bi-directional. Relation *A* implies relation *B* and vice versa. This assumption doesn't hold for many instances and needs to be regarded in the model.

In the next two sections ISP and LEDIR methods will be covered as proposed improvements to DIRT to filter erroneous relations.

3.2.1.4 Inference Selectional Preferences

Selectional preference of a predicate is the constraint that it puts over the class of words can be used as its arguments. For example: *x drives y* has a selectional preference on the second argument for motorized vehicles with more than two wheels and *x rides y* has a preference for vehicles or animals which rider is positioned astride [26] .

In order to model this type of constraint on discovery of paraphrases, Pantel et al. [32] have proposed that relations which are similar but are not sharing a same class of arguments should be filtered out. For each relation, they compute a distribution of possible semantic classes for its arguments. For two equivalent relations, they will be kept if this distribution is similar for both.Yao et al. [43] also incorporate selectional preferences to their model and which we will see in detailed in the last section.

Two models are proposed to compute this distribution. The first one, the distribution is computed jointly for both of arguments and the second one two different distributions are computed for each argument. The computation is just a simple count.

The question that naturally arises is that how one can obtain these semantic classes? The answer is that they are obtainable from lexical resources such as WordNet or they can be induced by clustering words. The authors have used both methods. As the first approach, they have used top level nodes in WordNet and all their successors together as semantic classes. They have also used CBC clustering algorithm [31] to induce semantic classes. In [14] and [23] a similar approach but with different clustering algorithms has been chosen and the later is the current state-of-the-art method.

After computing selectional preferences for each relation, those relations which don't have similar selectional preference distribution will be considered as nonequivalent. The plausibility of an inference rule which contains two relations is discussed in detailed in [8] and [22] .

Inducing word clusters and using independent selectional preferences distributions has been shown to outperform other models and have the highest increase of accuracy compare to DIRT.

3.2.1.5 Direction of Inference Rules

So far all the similar relations are considered to be paraphrases and they have bi-directional semantic equivalency relation.

Since this assumption doesn't hold for all the inference rules that are discovered by DIRT, Pantel et al. pushed their model another step further to find the direction of implications among relations, in addition to just find equivalent relations.

They have used the results from DIRT and ISP to find out which direction holds for an implication over two relations. having $relation_i \iff relation_j$ they want to examine if it holds or either of $relation_i \implies relation_j$ or $relation_j \implies relation_i$ is the correct rule.

the key assumption in [8] is that the direction is most likely from a specific relation to a more general relation. By specific relation they mean a relation that has a narrow set of semantic classes and selectional preferences and a general relation is a relation with broader semantic classes. Number of acceptable semantic classes for each relation is a good measure for such purpose. If this number is greater for the antecedent then the direction is right otherwise it should be flipped. A similar approach has been used in [22] which additionally confirm the results of this work.

In the next section we will review a method which operates in web-scale and extract relations with bootstrapping.

3.2.2 TextRunner

TextRunner is the first system that addresses open domain relation extraction in web-scale [44]. In this section we will describe the architecture of TextRunner and then we

will review two main elements of this system, *Learner* and *Extractor*, in more details and shortly we will comment on the third element *Assessor*.

3.2.2.1 TextRunner Architecture

Original TextRunner paper proposed four elements for its architecture:

- Learner
- Extractor
- Assessor
- Query Processor

The last sub-system has no major role in relation extraction so we put our focus on the first three elements.

3.2.2.2 Model Description

TextRunner avoids using parsing for large-scale and inhomogeneous corpus like web. *Learner* sub-system is responsible for providing a substitution or an approximation for syntactic parsing which, called as extractor by the authors. The approach that they follow in [44] is that they use a small corpus to train a relation classifier and this classifer will be applied to a much bigger web corpus. After parsing the small corpus, authors have used a set of predefined heuristics just based on POS tags and syntactic role of words to generate positive ad negative relation examples. The heuristics were designed as prototypes of syntactic behavior of general relations and they are not dependent on any certain type of relation. For example: $E_1 \ Verb \ E_2$ as template of *X created Y* or $E_1 \ NP \ Prep \ E_2$ for *X is birthplace of Y* are such heuristics.

A Naive Bayes classifier is self-trained on this small dataset, it starts with small seed of relations for training then it labels more instances in the corpus. This classifier will be used as *Extractor* in the next phase. This classifier is not relying on any lexical or relation-specific features, hence it can operate in open domain like web [3]. In the extraction phase,

a maximum-entropy classifer was used to find entities and then by using the extractor learned from the previous phase, they recognize explicit relations among named entities.

These extracted relations and named entities contain many redundant or equivalent relations. The role of *Assessor* is to find equivalent relations. It starts with normalization of relations and entities and then based on string-similarity and shared relational attributes it finds explicitly equivalent relations, it should be mentioned that this approach differs from DIRT which finds semantically equivalent relations.

3.2.2.3 Evaluation

TextRunner is evaluated against a manually tagged dataset of 500 sentences. While the precision is in the accepted level, 86.6%, recall is low (23.2%). The recall has been improved by using a *Conditional Random Field* [3] as classifier instead of Naive Bayes. The model precision is slightly improved by using a CRF for about 2% but we see a tremendous increase in recall which levels to 45.2% .

TextRunner suffers from using very shallow features. Its ability to find relations is limited to explicitly mentioned relations and it can only find them if they occur in a sentence. In the next part we will review a model which does not suffer from these disadvantages due to its deeper semantic analysis.

3.2.3 USP

In this section we will review the first unsupervised semantic parsing method which proposed by Poon and Domingos [35], we will first define what is semantic parsing and after describing the method in [35] we show its application for relation extraction. In the evaluation subsection we will compare this model to its related models and analyze its advantages and disadvantages.

3.2.3.1 Semantic Parsing

Semantic parsing is mapping a sentence to its formal meaning representation [35] The aim is to represent a natural language text with first-order logic. One can derive a semantic parse of a sentence by starting from a lexicon of atomic formulas and combining

each fragment to build a composition of formulas combined with quantifiers and logical connectives. In [35] the lexicon will be induced from a raw corpus. It is in contrast to traditional means of semantic parsing with manually produced lexicons.

The main challenge in unsupervised semantic parsing is that for a single semantic representation there could be several syntactic realizations or even harder, different surface representations. For example, all of the sentences below has a same semantic representation:

- Microsoft buys Skype
- Microsoft acquires the VoIP company Skype
- Skype is acquired by Microsoft Corporation
- The Redmond software giant buys Skype
- Microsoft's purchase of Skype,...

A simple lexicon to represent all of the examples above is:

$$BUY(n_1)$$

$$\lambda x_2.BUYER(n_1, x_2) \;\; \lambda x_3.BOUGHT(n_1, x_3)$$

$$MICROSOFT(n_2) \;\; SKYPE(n_3)$$

Having a corpus of sentences in natural language, USP [35] will induce such a lexicon and will also extract a formal representation for each sentence. In the next subsection we will review the necessary steps toward this goal.

3.2.3.2 Model Description

In this section we will first identify the three key ideas that the model is built upon them and then we will describe the necessary steps toward unsupervised semantic parsing.

Three observations are made by the authors which is crucial to understand the model assumptions:

1. Different syntactic variations of predicates and constants can be clustered together to express a same meaning. This can be learned from a raw corpus, in contrast to supervised methods which use meaning annotation of text.

2. Not only fixed elements in a relation can be clustered together but also arbitrary forms with same sub-forms to co-occurred forms can be put in a cluster. In this way the meaning composition will be learned through clustering of the forms.

3. Learning syntax and semantics jointly is a complex problem. Authors have shown that translating a syntactic analysis to semantic parse has superior performance over joint learning of syntax and semantics. In this way, a model can be built on the shoulder of state-of-the-art syntactic parsers.

Based on these assumptions, they propose a method to model joint probability of the dependency tree and its meaning representation. Markov Logic Networks (MLN) [36] is used to represent the meaning of a sentence. For each sentence we will have an undirected graphical model (Markov network) which each nodes corresponds to atoms and cliques correspond to first-order clauses. The best meaning representation is the one which maximizes the probability of the observed dependency [41]. These are the major steps of the USP system:

Dependency Parsing

the authors believe that dependency parsing is a better starting point that phrase-structure parsing, since it expresses the relation-argument structure at the lexical level. They have used Minipar [25] to make the results more comparable with previous methods, e.g. DIRT.

Converting Dependency Trees into Quasi-Logical Forms

Extracting lambda forms and atoms (lexical entries) is done by a deterministic procedure. Arguments of a dependency path will be converted to atoms, their predicate consists of the lemma and their part-of-speech tags. Each edge from the head to arguments will be a predicate labeled as type of the dependency path. This predicate together with two lambda function for each argument form a QLF. In this QLF, only one constant can be presented.

Clustering Lambda-Forms

It starts from atom levels and cluster them based on their relation, then it recursively

clusters larger formula based on their subformula lambda forms. The composition of meaning of any formula can be obtained by applying *lambda reduction* and substituting constants with lambda variables. At the end we will have a clustering of lambda forms which each cluster contains a set of semantically equivalent lambda forms. this solves the problem of having different syntactic realizations with a same meaning.

Creating a Markov Logic Network

Each atom can be modeled as a node and each first-order clause can be modeled as a feature. This is still not a complete Markov Logical Network since this model needs a weighting for features.Thus, the weight of feature is defined as a weight of its correspondent clause. A log-linear distribution ??? is used to model the probability of each clique (configuration of nodes and features). Z is a normalization factor which its computation is infeasible. w_i is the weight of ith formula. n_i is number of satisfied atoms and formulas that appeared in the sentence.

Learning the weights of MLN

Given the dependency parse and QLF forms, weights can be learned from data such as the log-likelihood of having QLFs given the dependency trees will be maximized. The form of likelihood is:

$$L_\theta(QLF) = \log \Sigma_L P_\theta(QLF, L)$$

where L is a semantic parse.

Since the summation of all possible semantic parses is infeasible, a set of additional constraints on the form of clauses and also a prior over weights are added to model to make the learning feasible. Their algorithm merges any possible two clusters to form a bigger cluster or create a new cluster if either increases the likelihood.

Finding the Best Semantic Parse

A semantic parse is a partition of atoms of a QLF, assignment of its head and argument form to a cluster. This can be seen as a structure prediction problem. Given the described MLN and its parameters, inference can be done by finding a Maximum A Posterior solution. An exact solution needs a summing over all possible syntactic realizations and possible dependency parses, therefor a greedy algorithm to search for a good solution is proposed by the authors.

3.2.3.3 Evaluation

Since USP is the first system for unsupervised semantic parsing, it is compared against other models in a Question/Answering task. GENIA is an Q/A annotated dataset in medical domain and USP has been shown to find the answers with 88% precision which is about 10% higher than TextRunner and more than 30% better than DIRT. It is worthwhile to mention that GENIA is a rather small dataset.

There are two major problems observed by the authors themselves and others for USP.

- Same as the other models that we have already seen, it doesn't address the separation of antonym relations
- It is mentioned in [43] and [41] that USP suffers from using a large memory and therefore low ability to scale to bigger dataset.

Two modifications for USP runtime is proposed in [34] and [41]. In [41] it is proposed to use directed graphical models instead of undirected graphs(MLN) to increase the learning and inference speed. The result is slightly worse than the original model but it performs faster than USP. In [34] a new representation of the model is given in the framework of deep learning which yields the same performance but with faster learning algorithm.

In the next section we will see another model which shows improvement to DIRT, TextRunner and USP and is much more scalable than the former.

3.2.4 Rel-LDA & Type-LDA

In this section we will review the method proposed for relation discovery by Yao et. al. [37] The method they proposed is an unsupervised method using generative models which is based on modification of well-known model, Latent Dirichlet Allocation. They have invented three similar models which we will go through them in following. We elaborate on input of the three models and then we describe their generative story and at the end we will finish this part by discussing about model evaluation.

3.2.4.1 Input format and preprocessing

Authors have chosen a subset of New York Times articles from year 2000 to 2007 which some of its parts with abnormal style of writing like obituary content are filtered out. They have done several preprocessing steps:

1. Tokenization
2. Sentence split
3. POS tagging
4. Named entity tagging
5. Dependency parsing

Among all dependency paths they are just interested in those which link two named entities. Authors assume that realization of one relation is available in this path most likely as a verb which relates two mentioned named entities. These types of paths are collected and based on some conditions that we have already discussed in Section **??** , non-important links are filtered out. Finally in this phase, 2.5 million dependency paths were collected which will serve as an input to the models that we describe in the next subsection.

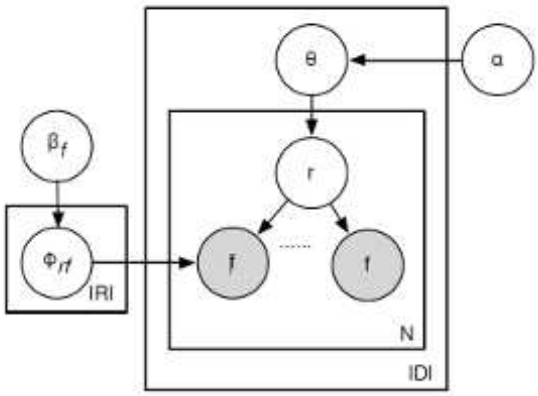
3.2.4.2 Model Description

All the three models in this work are generative models. Generative models are jointly modeling hidden variables and observable variables and are in contrast to discriminative models. For more information on generative models please see [29].

Rel-LDA is the first model proposed in [37]. In this model, features, f, are generated from a distribution , Φ_{rf}. Each type of relation is represented with a binary indicator variable, r. Each document is a mixture of few relations and relations are generated from a multinomial distribution Θ_{doc}. Θ_{doc} is a distribution of relations for a specific document and can be shown as $P(r|doc)$. By choosing a right prior distribution for Θ and Φ we can impose an assumption that any document contains a few number of relation types. Another advantage of having prior is that it helps to avoid overfitting. Dirichlet prior with

small parameter α is chosen for this purpose in the paper. For more on the role of priors and specially Dirichlet prior please read [40] and [19] . Rel-LDA is described with a graphical model in Figure 3.1 .

Figure 3.1: Rel-LDA



Shaded circles are showing the observable variables and other circles are hidden variables except the priors which are set by ourselves. The direction of arrow shows the dependency in this sense that destination node (variable) is dependent on source node.

Features or observable variables in Rel-LDA are:

- The dependency path
- Source of the path (the first named entity)
- Destination of the path (the second named entity)

An exact sampling method called *variational inference* is used by authors to compute the posterior distribution of the model, $P(r|f)$. After learning, we will have clusters of relation types which each instance of a specific cluster is a relation that is supposed to be semantically equivalent to the other members of this cluster.

In the second model, **Rel-LDA1**,The only difference is that more features are used in the learning procedure. The idea is that using more features could be useful as tie-breakers and also discriminators between instances to make new clusters. With direct reference

to the paper, the authors believe that more features lead to better refinement of clusters. We will see later that this assumption holds in practice. The new features that they have introduced in this new model are:

- Trigger: Any word in a dependency path except than stop words.
- Part of speech sequence: The sequence of POS tags of a dependency path.
- Named entity pair: The type of source and destination named entity.
- Syntactic pair: The type of dependency edges connecting source and destination to the head.

Among all the newly introduced features, NE pair is the most interesting one which leads to a significant observation that the third model will try to address that. Authors observed that Rel-LDA will put these three relations in one cluster because the second argument of all of them is location:

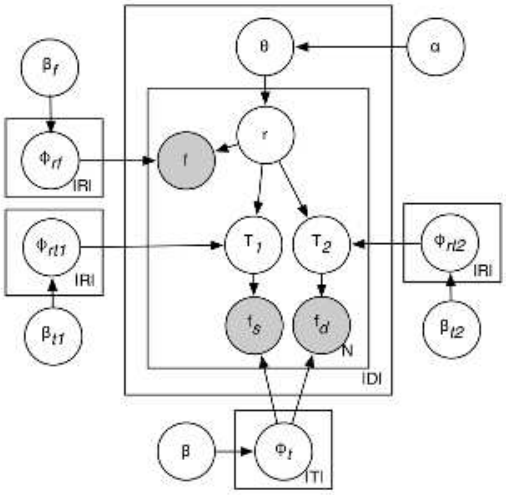
1. *X was born in Y*
2. *X lives in Y*
3. *X, a company in Y*

While the first argument in the first two relations refers to a PER, the first argument of the third relation is an instance of ORG type. By using NER pairs as feature we can split this cluster to two clusters with respect to basic types of NE in relations arguments. This observation leads us to invest more on type identification of arguments to have more pure clusters and is the main focus of the third model, **Type-LDA**.

Selectional preferences of a relation are constraint over possible arguments for the relation. Basically, any relation only accepts a few number of entity types and this could be an important constraint for inducing relations. Relations of each cluster should have similar selectional preferences and accept same entity types. Type-LDA is proposed for this reason, it will induce entity types and relation clusters jointly to benefit more from selectional preferences of relations. As we have already mentioned in Section **??**, this idea have been used in [32] .

The generative model is modified in a way that features of arguments (source and destination) will be generated from two new distributions, T_1 & T_2 which are modeling entity types. The graphical model is shown in Figure 3.2 . This model not only clusters relations

Figure 3.2: Type-LDA



but also clusters entities to entity clusters.

For their experiments on all three models they have set the number of relation clusters to 100 and for Type-LDA they used 50 entity clusters. Choosing other numbers of relation clusters in the range of 50 to 200 is shown to be not very significant.

3.2.4.3 Evaluation

They have used human judgments to measure their models precision. Humans are asked to label 50 instances in each relation cluster and in order to measure the recall, induced relations are compared against Freebase. Rel-LDA1 and Type-LDA which are using selectional preferences features are performing better than Rel-LDA. Rel-LDA1 is shown to have the best precision among other models. Rel-LDA and Type-LDA have been also compared against USP and have been shown to have superior performance both in scalability and F-measure.

Like most of the other models, antonymy is not handled in their model and therefor for example they have *X was born in Y* and *X die in Y* in one cluster.

Entity clusters also sometimes suffer from high-frequency or low-frequency words and for example, ‘New York‘ , is in the same cluster as other publications like ‘New York Times‘, ‘Vanity Fair‘ and . . .

Yao et al. have shown that they have induced relations in different granularity from Freebase and reported some relations which are not mentioned in Freebase but positively hold. For example, Freebase relation *worksFor* is subsumed with more relations each indicates a different role of employment relation. *leaderOf* and *editorOf* are such examples. This will boost the idea of inducing hierarchical relations which will be discussed more in the last chapter. A similar approach which tries to induce hierarchical structures is [1] .

3.2.5 Universal Schema

kkk

3.3 Problem Identification

Based on what we have seen in recent works, we can now give a list of vital attributes that a state-of-the-art model for extracting relations from open text should be able to carry out. The author will use these facts to suggest a list of possible improvements in the next section. Modeling all of these factors in a joint model is the necessary step to push forward the previous works.

The number of relations and entities is an unknown parameter.
: The model can not be confined to a limited set of relations or entities. Being able to extract relations in open domain text is the first and (most likely) a trivial attribute of the model. More non-trivial feature of the model should be its ability to extract as many relations as there are in text. Giving this freedom about the model complexity to have no assumption about the exact number of entities and relations is suggested by the applicant to be beneficial and is also supported in the literature. [27] [44]

Relations may not be expressed explicitly in text.
: Relation extraction task is definitely more than finding paraphrases. The model should be able to handle long-distance relations among entities as well as hidden semantic indications of a relation. [35]

Relations and entities have their inner organization and types.

It is shown by several recent works that relations of relations play a substantial role in identifying relations. Relations and entities belong to a hierarchy of types and therefore the constraints they put on each other should be learned as well.[43] [1] [30]

Using KB is necessary but not enough.

There are more relations among entities than what is collected in Knowledge Bases e.g. Freebase. At the same time, it is statistically shown that a supervision from such resources strongly contributes to convergence of any model to a better objective configurations.[43] [27]

Relations and entities are sharing information within each other.

Relations and entities should be learned jointly since they share same explanatory factors (hidden variables) . Meaning of an entity can be learned from its relation to other entities and same argument holds among relations. Basically, the model should be able to carry out multi-task learning. [43]

4 Linking Text to Knowledge Base for Relation Discovery

In this chapter I introduce an idea which relates previous work on representation learning of KBs to relation extraction from text. First I propose a method which tries to learn embeddings of entities and relations both from Freebase and a corpus prepared in a specific format. With such a rich embeddings, I will propose a model to and settings to predict links between entities in Freebase or relations among entities mentioned in the corpus. Later on, the experimental setup and a pipeline created for this purpose will be described and finally in the last section we will evaluate our model and discuss different aspects of the results.

4.1 Learning Representaion of Entities and Relations from Text and Knowledge Base

Two direction of works have been conducted previously on learning representation of words which we discussed in 2.2. 1) KB-based representation learning and 2)Corpus-based methods. Here I will demonstrate a process which will enable us to jointly learn embeddings with both contextual and lexical information. Among previously discussed methods, I borrow a model proposed in [11]. As I described in 2.2.3, this model can take set of binary predicates and their arguments and induce continuous features for both predicates and arguments in vector space. Using these embeddings and a bilinear combination of them, the model can discriminate between true facts in the dataset and negative examples. For more information of this model please see subsection 2.2.3.

We can see that this models in its plain vanilla form is limited to learn embeddings from a KB but has a great potential to be extended in various aspects. One of the possible aspects which matters for relation extraction task, which is main motivation of this work, is that it should be able to learn facts also from text. The reason is that, despite the vast effort of

gathering information and encode it as knowledge in KBs, they are limited in the sense of coverage. To compensate this problem we need to discover new facts from corpus.

In chapter 3 we have extensively discussed major related works and we also discussed important features that were being used in many important works and are shown to be very effective. Most of these feature were previously encoded in a way that can be used for classic classifiers. I will first enumerate these features and then we will see that how can we formalized them to make it possible to use them with a neural distributed model discussed in subsection 2.2.3. A thorough discussion of creating dataset will appear in section 4.4 but for now, it is sufficient to know that by a corpus we mean, a pre-processed clean pile of text in English, coming from harvested web pages or from news papers which contains these annotations: 1) part of speech tags 2)dependency parsed 3) named entities are recognized and tagged by their type. Our process of extracting features will be described fully in section 4.4, in the next part, Any feature I describe comes from a sentence which contains two named entities.

4.1.1 Informative Features for Relation Extraction

In this part, I briefly enumerate important features which have been used extensively in previous works. For an actual example, I show some instances from a dataset prepared by Riedel et al. and used in [37] and [38]. For full description of futures in their work please see subsection 3.2.5.

A relevant set of features that I harvested from text for my work is as follow:

Type of Named Entities

These types are simple set of types that usually named entity taggers tag entities with them. For example: **LOC** for locations, **PER** for persons. These types contain a minimalistic information but yet useful. The actual type of entities, features of an entity which are common with similar entities should be induced by the model.

Dependency Role of Named Entities

Dependency structures have useful information and there is a long history of using dependency patterns in literature. For example the importance of dependency roles is discussed in subsubsection 3.2.1.3. I follow previous works and will use this type of features in my proposed model. As an example a NE in a senctence can

have either of these roles: direct object *dobj*, passive object *pobj*, appositional modifier *appos*, participial modifier *partmod*, nominal subject *nsubj*, noun compound modifier *nn* and prepositional modifier *prep*.

Head of Sentence in Dependency Path

The dependency path between arguments and the head of the sentence worked as surface patterns indicating a relation if it appears in different contexts. All the head words are collected and I use them as surface patterns. They are labels for relations that we want to induce or predict among entities.

Another set of features, the most important one, which are available in KBs are actual knowledge about entities and the relation between them. This types of relations naturally are encoded as predicate-arguments relations. For example this relation:

/PEOPLE/PERSON/NATIONALITY(MIR-HOSSEIN MOUSAVI, IRAN)

indicates that *Mir-Hossein Mousavi* has *Iranian* nationality.

4.2 Task Description

We have a set of facts, $\mathcal{F}$, coming from a KB which any instance of it is in form of a triple (e_i, rel_k, e_j) which arguments of this triple are left and right named entities and the predicate is a relation from a certain type. Additionally we have a corpus of text, $\mathcal{C}$, which is tagged by part of speech, named entities and parsed with a dependency parser. A positive triple is a triple which at least an element of context in $\mathcal{C}$, here a sentence, supports this triple. Likewise, a negative triple is a triple that there is no support available for it in the corpus. Given $\mathcal{F}$ and $\mathcal{C}$ we would like to perform two tasks jointly:

1. Learning embeddings of named entities, KB relations and surface patterns from information in $\mathcal{F}$ and $\mathcal{C}$.
2. Learning a model with objective of ranking all positive triples lower than all negative triples, preferably with a large margin.

In the domain of machine learning there are several models with ability of performing learning representations and classification jointly which we discussed some of them in subsection 2.2.2. Among those models relevant to our task, we picked the neural distributed model proposed in [11]. A detail of this model is discussed in subsection 2.2.3.

This model takes a set of triples $\mathcal{T}$ as an input and perform both of our desired tasks. With having this model the only problem now is to generate $\mathcal{T}$ such as it staisfies our objective. In the next section I will describe a method for generating this dataset and later in section 4.5 I emperically show the effectivness of this method.

4.3 Linking Text to KB

To benefit from information both available in a KB and hidden in the text we need to link them so information can transfer from one to the another. This can be happened through sharing the task of learning representations of words specially NEs and surface patterns. This section is dedicated to elaborate on the approach I took to share the task of representation learning between text and KB.

The main idea is to present important features of text in such a formalization that can be mutually learned with set of facts that come from a KB, for instance Freebase. We mentioned that knowledge in Freebase is encoded as triples of a predicate and two arguments. We take the same formalism and by introducing auxillary predicates, we encode the annotations of a corpus in form of triples.

These are list of auxiliary relations which we add to $\mathcal{T}$ with their definition:

HAS_TYPE
: this predicate takes a NE as left argument and its type produced by NE-tagger as right argument. For example:

HAS_TYPE(MIR-HOSSEIN MOUSAVI, PER)

HAS_DEP_ROLE
: with this predicate we relate a NE to its dependency role:

HAS_DEP_ROLE(MIR-HOSSEIN MOUSAVI, DOBJ)

Head of Dependency path

for each dependency path, we can consider head of a path as a relation or a predicate which relates two NEs of the path together. This head word ,which in our case is always verb but in general can also be a nominal phrase, will work as a surface pattern and a candidate relation. From now on we will call this head words *triggers*, following the work in [37]. For example for this sentence and its dependency path:

Mir-Hossein Mousavi, president of Iran, said ...

PATH#APPOS|->APPOS->PRESIDENT->PREP->OF->POBJ->|POBJ

we will have the relation below in $\mathcal{T}$:

PRESIDENT (MIR-HOSSEIN MOUSAVI, IRAN)

HAS_TRIGGER

we know that we have two types of relations in $\mathcal{T}$, relations that come from corpus $\mathcal{C}$ and those from a KB $\mathcal{F}$. If there are two NEs in $\mathcal{C}$ which are related by a trigger and there exist a relation between them in $\mathcal{F}$ too then we will add a third relation, *HAS_TRIGGER* between the knowledge base relation and a trigger relation. For example given these two relations in $\mathcal{T}$:

PRESIDENT (MIR-HOSSEIN MOUSAVI, IRAN)

/GOVERNMENT/POSITION_HELD/PRESIDENT (MIR-HOSSEIN MOUSAVI, IRAN)

we will add the third relation below to $\mathcal{T}$:

HAS_TRIGGER (/GOVERNMENT/POSITION$_h$ELD/PRESIDENT, PRESIDENT)

The reason is to increase correlation between KB relations and text surface patterns which will enable the model to transfer information across KB and corpus.

Having $\mathcal{T}$ generated now we can run our experiments. In the next section I describe the pipeline and after that we will see the evaluation of our approach.

4.4 Experimental Setup

In this section, first we will describe the process of producing a dataset I use for running experiments and then we will see different experiments I ran to examine the effectiveness of proposed approach for relation discovery.

4.4.1 Creating Dataset

There are three phases to create a suitable dataset for our experiments. In the first phase, a corpus should be annotated by part of speech and named entity tags and parsed with dependency parser. This will lead to produce our desired corpus $\mathcal{C}$. In the second phase, we pick a KB and use it as our $\mathcal{F}$ dataset. If $\mathcal{F}$ is too big in sense of size of relations for our computing resource we can limit ourselves to a subset of KB. I followed the heuristic proposed in [27] and [37] and limit $\mathcal{F}$ only to relations which their argument have been co-occurred in a sentence in $\mathcal{C}$.

In the third phase, using $\mathcal{C}$ and $\mathcal{F}$ we produce triples that we discussed in section 4.3 and create our final dataset $\mathcal{T}$.

To make our corpus we follow the protocol described in [37] and [39] and use their dataset since it is fully compatible with our requirements. They used NYTimes corpus in their work. After pre-processing and tagging this corpus, it is splitted to two parts. Articles after year 2000 as training and between 1990 to 2000 for the test dataset. For KB, Freebase has been chosen and only those relations have been included to $\mathcal{F}$ that both left hand side and right hand side of relation is available in a sentence in the corpus. If for both NEs' of a sentence there was no relation in Freebase we have used *NA* relation and we call it a negative instance. We used a subset of this dataset. For training we picked 8409 positive and the same number of negative instances. For test set we used 4000 instances, balanced between number of positive and negative instances. Based on this dataset we run our experiments. As an example you can see one positive and one negative instance below:

POSITIVE

left entity:Edward M. Liddy right entity:Allstate trigger:executive path:appos|->appos->executive->prep->of->pobj->|pobj
Freebase-Rel:/business/company_shareholder/major_shareholder_of NE:(PERSON, ORGANIZATION)

NEGATIVE

left entity: Susan Arnold right entity: Sweethearts trigger: producer NE:(PERSON,MISC) path: appos|->appos->producer->dep->|dep Freebase-Rel:NA

4.4.2 Experiments

We arrange different settings and experiments to examine the role of our features and compare it to the setting previously used in [12] and [11]. We start from features used in the previous work and gradually introduce our own features in each experiment.

KB (Experiment 1)

In this experiment we only use Freebase relations for the sake of repeating the proposed setting in [12]. In this setting, there is no actual link from text to KB and therefor by comparing other experiments against experiment 1 we can see the efftect of linking through our proposed formalism.

KB+Trigger (Experiment 2)

This experiment is designed to see the effect of adding surface pattern features to Freebase relations and learn them jointly. In this settings, our train dataset consists only of Freebase relations and trigger relations.

Text+KB\Trigger (Experiment 3)

All the features are used in this experiment except the surface pattern (trigger) relations.

Text+KB (Experiment 4)

All the features including surface patterns are employed and we have full linkage between corpus and KB.

All of these experiments are conducted with same parameters of the model. Even though the structure of the forth experiment is more complex than for example the first experiment but by using same parameters the outcome of the experiments can be analyzed based on the effectiveness of the feature set of each experiments. For all experiments we used 50-dimensional embeddings and we used SME-Bil neural distributed model. Number of iterations were 500 for all experiments and with fixed number of batches, namely 100. The details of this model has been previously described in subsection 2.2.3.

Theano [7], a math compiler in python is being used to develop the model implemented previously in [12] and [11] and I further developed a pipeline based on these modules to run the experiments. Since I have not had access to GPU, I ran my experiments on

CPU which for experiment 4 took about few days CPU time and few hours for the first experiment.

In the next section we will see the results and comparison of different experiments.

4.5 Evaluation and Analysis

The goal our model is to predict a relation between two given entities. So how do we do prediction with our model? We basically rank all possible relations based on a similarity score we already introduced in (2.4). Given two entities, we take their embeddings we learned through the training phase and then iterate over all embeddings of Freebase (or surface patterns) and calculate a score for triple of these three embeddings. Then we can sort all relations based on this score, this will construct our prediction. One should notice that the output is not probabilistic. In [12] a method based on kernel density estimation has been proposed to make a probabilistic output but since it is reported ot overfit on the training dataset I don't use it here. Having all the relations sorted for a given tuple of entities for our test cases, we can find the predicted rank of the true relation and report statistics per relation and for whole test set. The statistics I report here are mean and median of true rank per relation and also the number of times that true relation appeared on top-r relations, here 100. For aggregating these statistics for the whole dataset we can follow to protocols. In one type of aggregation we give more weight for more frequent relations (Micro statistics) and in the other protocol we compute mean and median of true ranks without considering the popularity of a relation, we give equal weight for each relation (Macro statistics).

the first observation from Table 4.1 is the significant role of features. We can see that the *Experiment 4* has the best performance in all the statistics and it benefits from full set of features. The comparison between results of the first model, Bordes plain *KB* model and the *KB+Text* model reveals that introduced features are highly effective and the reconstruction (extending) of KBs can be done better. Not that with this formalization we can achieve a relation extraction model linked to text but also we can use it as a model to extend and populate current KBs. The big difference between Micro and Macro statistics for this model suggests that its good performance on frequent relations can also be achieved for less frequent relations through learning larger size of Freebase. We believe all the models will for sure benefit from this but the forth model has already shown a

Table 4.1: Relation Prediction Evaluation

Dataset	Feature Type		Micro	Macro
Experiment 1	KB	mean	71.11	78.39
		median	31.0	72.73
		r@100	67.05	62.79
Experiment 2	KB+Trigger	mean	639.67	534.72
		median	544.0	503.20
		r@100	20.92	19.85
Experiment 3	All \Trigger	mean	59.63	61.59
		median	25.50	56.49
		r@100	73.85	73.88
Experiment 4	All	mean	6.72	57.13
		median	2.0	55.71
		r@100	98.85	76.88

superior performance for relations occurred more than a certain threshold and by feeding bigger portions of our KB we can help less frequent relations to pass this threshold. To investigate this effect we compare the per relation performance for two group of high and low frequent relations.

By comparing performance of *KB* and *Text+KB* in Table 4.2 we can observe that while *Text+KB* outperforms other models overall and for high frequent relations but for very low frequent relations, **/people/person/religion** and **/book/book_edition/publisher**, it doesn't perform as good as *KB*. This suggests that since in *Text+KB* we are learning a much more complex structure,we need more iterations to learn better features. For comparison we used same number of iterations for all models to emphasis on the formalization rather than parameter tuning but we can observe that *Text+KB* is being penalized by this and by more iterations of SGD we can achieve even better results.

4.6 Conclusion

In this chapter I introduced a new formalization and a process which enables us to link a corpus to a KB and then improve the learning of features from KB. With this formalization we can grounding KB relations to text and also populate a KB with facts coming from a corpus. We showed that the features come from both text and KB are more informative for relation extraction task than features that comes only from a KB.

Table 4.2: Per Relation Evaluation

Relation	Frequency		KB	KB+Trigger	Text+KB\Trigger	Text+KB
NA	2000	mean	88.65	789.27	83.27	2.55
		median	60.0	797.0	64.00	1.0
		r@100	61.55	14.10	62.79	99.95
/location/containedby	688	mean	49.37	577.79	28.85	3.60
		median	9.0	468.00	5.00	2.00
		r@100	78.34	20.34	89.39	99.85
/people/person/place_lived	132	mean	39.03	531.25	25.09	6.68
		median	12.50	410.5	8.0	4.0
		r@100	84.84	25.75	93.18	100.0
/person/company	124	mean	47.00	359.48	16.43	2.83
		median	7.5	105.5	3.0	2.0
		r@100	79.83	50	95.16	100.0
/deceased_person/place_of_death	80	mean	47.95	433.36	15.60	3.92
		median	15.0	315.5	4.0	2.0
		r@100	78.75	36.25	95.00	100.0
/people/person/ethnicity	7	mean	77.71	444.57	34.00	33.14
		median	31.00	413.0	37.0	20.0
		r@100	57.14	28.57	100.0	85.71
/music/composer/compositions	5	mean	89.25	331.25	51.75	52.50
		median	77.0	295.5	24.5	22.5
		r@100	50.00	0.0	75.00	75.00
/book/book_edition/publisher	3	mean	46.33	354.00	135.66	71.0
		median	43.00	157.0	142.0	90.0
		r@100	100.0	33.33	33.33	100.0
/people/person/religion	3	mean	20.33	220.00	4.33	51.33
		median	8.0	179.0	2.0	18.0
		r@100	100.0	33.33	100.0	66.66

5 Entity Linking Among Lexical Resources[1]

Each KB or lexical resource contains limited amount of information and different structured resources have different perspective and partially encode information about an entity and its relations to the other entities of the resource. Therefor, if one is interested about learning different features from a lexical resource or a KB, a natural next step is to jointly learn features from multiple resources. In previous chapter we saw how to learn features from a KB and a corpus with application to relation discovery and extraction. In this chapter we focus on inducing features for entities of multiple structured KBs. For relation extraction, the most relevant available resource is Freebase and it suffices together with corpus to learn features of named entities (NE). In this chapter we focus on learning features for other type of words e.g. nouns, verbs and . . . . Learning features for other words in a sentence rather than just NEs might help improving the performance of a relation discovery system specially for verbs (relations) when some semantic aspects of them like selectional preference has been shown to be an important aspect. Another application is learning and linking entities of lexical resources in two different languages which helps to transfer information from one language domain to the another one. For these reasons, first we describe a pipeline to learn word features from multiple resources and as proof of concept I implement this idea with inducing bi-lingual word features and show its performance on semantic similarity task. The actual application of this idea and learned word features is postponed to future work which I describe potential usage of them in relation discovery.

[1] This part of the work has been done during an internship in UKP Lab, Technical University of Darmstadt. All the codes and results produced during this time belongs to Prof. Dr. Iryna Gurevych and can be obtained upon the request.

5.1 Task Description

In this part I describe the methodology we followed to encode available information in two different lexical resources, WordNet and GermaNet [9], that makes it possible to link entities of the two different resources and learn bi-lingual embeddings of word senses in German and English. The main idea is to relate two senses from two different resources using cross-lingual sense alignments. This is an additional information which can play a role of bridge between two different tasks, learning German embeddings and English embeddings, and can help to transfer knowledge from one to the another. Using this new feature we make our WordNet-GermaNet dataset which contains three types of relations (1) WordNet relations (2) GermaNet relations (3) Cross-lingual sense alignments between WordNet and GermaNet

First two types of relations are directly extracted from WordNet and GermaNet and for the cross-lingual relations we used Interlingual Index mappings between WordNet and GermaNet.
Example of relations:

WN-sense-A	WN-rel-1	WN-sense-B
GN-sense-C	GN-rel-2	GN-sense-D
WN-sense-A	ILI-rel-1-2	GN-sense-D

Left and right entities are WordNet and GermaNet senses and relations are current semantical relations in each of lexicons such as: meronymy, holonymy and ....

5.2 Experimental Setup

We have created four different dataset, each divided to train, test and validation separated subsets. Our four datasets are:

1. Only WordNet triples (WN)
2. Only GermaNet triples (GN)
3. WordNet-GermaNet triples with one-direction cross-lingual alignments (WN-GN)

4. WordNet-GermaNet with double-direction cross-lingual alignments (WN-GN DD)

Dataset 3 includes both relations extracted from WordNet and GermaNet and also the mapping between senses. Dataset 4 is same as dataset 3 but since the models we will use are assuming all the relations are assymetric we will try to encode the symmetry of cross-lingual alignments by reversing each of them and include the reverse in the dataset. Datasets 3 and 4 contains two different variants: the first variants contains only WordNet relations (test on WN) in the held-out test dataset and the second variant contains only GermaNet triples (test on GN). In this way we can observe the direction of possibly transferring information from English to German or vice versa.

For reducing the sparsity of data and boosting the learning runtime we filtered out all the entities that appeared less than 3 times in our datasets.

For implementation I used a unified lexical resource called *Uby* [17] which provides an integrated API to work with Wordnet, GermaNet and other resources.

5.3 Evaluation

To show the effectiveness of joint learning of features from multiple knowledge bases we suggest two experiment setups. In the first schema we follow Bordes et al. ranking task. The goal of this task is to show how well the information in knowledge bases can be preserved by the learned features. On the other hand, the second setup is investigating on this question that if the learned word embeddings from multiple resources are able to improve the performance of monolingual embeddings in a standard NLP task, here word-pair similarity or not. In this setup we will look to contribution of the learned features in predicting similarity of words.

5.3.1 Evaluation Using Reconstruction

Bordes et al. (Bordes2011) proposed a ranking task that for each triple (e_i, r_k, e_j) in the data set, all the entities will be ranked as a candidate for being right entity of the triple given the relation and the left entity. Depends on which one of the models is used, SE or SME-Bil, all the entities will be sorted based on their score regarding Equation (2.2) or Equation 2.4 previously introduced in section **??**. By keeping the statistics of difference

between the predicted rank of e_j and its true rank and also repeat the same process for left entities, we will be able to report the mean and median predicted rank of entities per relation and in total. Bordes et al. proposed to schema for calculating the average rank, micro averaging which emphasis on more frequent relations by weighted averaging with frequency of relations as weights and macro averaging which consider all the relations equally , either frequent or infrequent ones. The third statistic that we report following their work, r@100, is the ratio of number of times that an entity is correctly among top 100 entities ranked and predicted for a triple to the number of occurrences of this entity in the dataset. We applied SE and SME-Bil models on our created datasets and the ranking performance on each of them is presented in Table 5.1.

5.3.2 Evaluation on Semantic Similarity

We are interested to further analyze the effectiveness of learned embeddings to capture semantic features of words, therefore we compare the mono-lingual and bi-lingual embeddings against human judgments and also other embeddings learned from corpus. The other embeddings which we used for our comparison are Turian et al. [42], Mnih et al. [28] mentioned in subsection 2.2.2 and Klementiev et al. [21] embeddings To measure the similarity between any given wordpair (w_1, w_2) we find all vectors associated to different senses of the given words in our embedding dictionary and pick the pair of embeddings that maximize cosine similarity between two words. We can motivate this by saying that for each word pair any of words works as a context for disambiguating the sense of the other word.

Four datasets of word-pair similarity are used to compare correlation of predicted similairty of pair of words against human judgments. RG-65 [rubensteinGoodenough], Yang and Powers verb similarity dataset[yangPowers], MC-30[millerCharles] and WS-353 [finkelstein] are English datasets that we used for this task. RG-65 and its subset, MC-30 are providing human scored datasets for measuring synonymy among wordpairs (nouns),. WS-353 has broader notaion of semantic similarity and include word pairs for measuirng semantic relatedness too. Yang and Powers have provided a dataset for measuring semantic similarity between verbs.

Both Pearson and Spearman correlation of predicted and gold similarities are calculated and is reported in table 5.2 for English. We can see that bi-lingual embeddings learned by SME-Bil model outperformed all other embeddings both in Pearson and Spearman

correlation in all four datasets. Another observation is that SME-Bil models performes better than SE models in most cases. Among SME-Bil models, bi-lingual embeddings are always more correlated to human hudgments than mono-lingual embeddings. All models perform poorly on WS-353, bi-lingual SME-Bil model still performs better than the rest. We believe this could due to including notion of relatedness in this dataset. Poor performance of WordNet based models on semantic relatedness task is previously known and discussed in [**?**][Szulmanski2013]. Words can be related by different type of relations while in WordNet, a few number of relations are encoded e.g. synonymy which is the subject of our evaluation task.

For German, we use translated versions of RG-65, MC-30 and WS-353 which were judged by German native speakers [**?**] [gurevych et al], additionally a dataset from [**?**] [Gurevych et al.] is used. The structured embeddings are being compared to the only available German word embeddings from [21][Klem et al.]. We can observe that bi-lingual embeddings always outperform mono-lingual embeddings except for RG-DE-65 dataset which both mono-lingual and bi-lingual **SE** models have equal Spearman correlation to the gold data but mono-lingual model performs better in Pearson correlation. The good results of German evaluation is another indicator for proofing the idea of learning word embeddings from multiple resources.

5.4 Conclusion

In this chapter the idea of learning word embeddings was motivated by first formulating the task of learning bi-lingual word embeddings from WordNet and GermaNet and then evaluating the embeddings on semantic similarity task. Variation of models used to induce and compare word embeddings and the better performance of bi-lingual word embeddings in compare to mono-lingual and the other models to gold human judgments for word pair semantic similarity datasets showed the effectiveness of this idea. These induced word features, specially for verbs, have the potential of being used in a relation discovery system as an extra information. By including more resources, which is trivial in the proposed framework different aspects of a word or named enitiy can be learned which can even increase the performance of current systems potentially.

Table 5.1: Intrinsic Evaluation (Ranking Score Performance)

Dataset	#relations	#entities		Micro	Macro
GN SE	16	64025	mean	1003.59	3739.85
			median	5.0	2213.37
			global	84.23	72.49
GN SME-Bil	16	64025	mean	407.90	308.01
			median	10.0	54.18
			global	81.18	69.85
WN SE	23	148976	mean	148.72	623.10
			median	5.0	4.69
			global	92.10	89.86
WN SME-Bil	23	148976	mean	128.82	511.21
			median	10.0	26.63
			global	84.14	75.57
WN-GN SE (WN held out)	32	213002	mean	293.16	1356.30
			median	5.0	5.10
			global	91.19	88.95
WN-GN SME-Bil(WN held out)	32	213002	mean	124.85	331.82
			median	11.0	33.86
			global	82.91	73.55
WN-GN SE (GN held out)	32	213002	mean	3031.44	15470.56
			median	7.0	10080.5
			global	80.87	70.313
WN-GN SME-Bil(GN held out)	32	213002	mean	984.79	1021.37
			median	40.0	428.90
			global	64.16	55.98
WordNet-GermaNet-DD SME-Bil (WN held out)	32	213002	mean	166.18	466.91
			median	18.0	55.41
			global	77.07	65.082
WordNet-GermaNet-DD SME-Bil (GN held out)	32	213002	mean	932.49	719.47
			median	56.0	175.56
			global	59.22	50.84

Table 5.2: Word-pair Similarity Performance for English

Dataset		WN-SE	WN-GN-SE	WN-SME-Bil	WN-GN-SME-Bil	WN-GN-SME-Bil-DD	HLBL	Turian et al.	Klementiev et al
RG-65	P	0.682	0.666	0.703	0.833	0.725	-0.115	0.233	-0.380
	S	0.769	0.741	0.741	0.811	0.825	-.083	0.118	-0.398
MC-30	P	0.611	0.644	0.601	0.740	0.599	-0.363	0.150	-0.768
	S	0.720	0.648	0.756	0.846	0.954	-.450	-0.198	-0.522
WS-353	P	0.181	0.206	0.239	0.246	0.238	0.233	0.236	0.029
	S	0.093	0.146	0.185	0.224	0.201	0.197	0.210	0.040
YangPowers-130	P	0.482	0.637	0.584	0.627	0.610	-0.130	-0.076	0.154
	S	0.401	0.472	0.406	0.553	0.533	-0.186	-0.116	0.113

Table 5.3: Word-pair Similarity Performance for German

Dataset		GN-SE	WN-GN-SE	GN-SME-Bil	WN-GN-SME-Bil	WN-GN-SME-Bil-DD	Klementiev et al.
wortpaare222	P	-0.010	0.156	0.073	0.130	0.196	0.107
	S	-0.125	0.234	0.152	0.175	0.111	0.152
MC-DE-30	P	0.865	0.984	0.185	0.287	0.301	-0.887
	S	1.0	1.0	-0.500	-0.500	0.500	-1.0
WS-DE-350	P	-0.085	0.063	0.185	0.188	0.127	0.187
	S	-0.157	0.009	0.172	0.194	0.142	0.142
RG-DE-65	P	0.800	0.558	-0.572	0.485	-0.166	0.233
	S	0.800	0.800	-0.8	0.399	0.200	0.200

6 Future Work

7 Conclusion

Bibliography

[1] Enrique Alfonseca and K Filippova. Pattern learning for relation extraction with a hierarchical topic model. In *ACL*, number July, pages 54–59, 2012.

[2] Ion Androutsopoulos and Prodromos Malakasiotis. A survey of paraphrasing and textual entailment methods. *arXiv preprint arXiv:0912.3747*, 38:135–187, 2009.

[3] Michele Banko. *Open Information Extraction from the Web.* PhD thesis, University of Washingotn, 2009.

[4] Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open Information Extraction from the Web. In *IJCAI*, pages 2670–2676, 2007.

[5] Y Bengio and R Ducharme. A neural probabilistic language model. *The Journal of Machine ...*, 3:1137–1155, 2003.

[6] Yoshua Bengio. Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, 2009.

[7] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-farley, and Yoshua Bengio. Theano : A CPU and GPU Math Compiler in Python. In *Scipy 2010*, number Scipy, pages 1–7, 2010.

[8] Rahul Bhagat, Patrick Pantel, EH Hovy, and Marina Rey. LEDIR: An Unsupervised Algorithm for Learning Directionality of Inference Rules. In *EMNLP-CoNLL*, number June, pages 161–170, 2007.

[9] Helmut Feldweg Birgit Hamp. GermaNet - a Lexical-Semantic Net for German.

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

[11] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Artificial Intelligence and Statistics (AISTATS)*, volume 22, 2012.

[12] Antoine Bordes, Jason Weston, R Collobert, and Y Bengio. Learning structured embeddings of knowledge bases. *AAAI*, (Bengio):301–306, 2011.

[13] L Bottou. Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT'2010*, (x), 2010.

[14] PF Brown, PV Desouza, and RL Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 4:467–479, 1992.

[15] R Collobert and Jason Weston. A unified architecture for natural language processing: deep neural networks with multitask learning. ... *international conference on Machine learning*, 2008.

[16] Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (almost) from Scratch. *Machine Learning Research*, 12(August):2493–2537, March 2011.

[17] J Eckle-Kohler. UBY-LMF–A Uniform Model for Standardizing Heterogeneous Lexical-Semantic Resources in ISO-LMF. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, page 8, 2012.

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

[19] Samuel J. Gershman and David M. Blei. A tutorial on Bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, February 2012.

[20] ZS Harris. *Distributional structure*. Springer Netherlands, 1981.

[21] Ivan Klementiev, Alexandre; Titov. Crosslingual Distributed Representations of Words.

[22] Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(04):359–389, October 2010.

[23] D Lin and X Wu. Phrase clustering for discriminative learning. In *ACL-AFNLP*, pages 1030–1038, 2009.

[24] Dekang Lin and Patrick Pantel. DIRT - Discovery of Inference Rules from Text. In *... conference on Knowledge discovery and data mining*, pages 323–328. Association for Computing Machinery, 2001.

[25] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(04), February 2002.

[26] M Mechura. *Selectional Preferences, Corpora and Ontologies*. PhD thesis, Trinity College, University of Dublin, 2008.

[27] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 2, pages 1003 – 1011, 2009.

[28] Andriy Mnih and Geoffrey Hinton. A scalable hierarchical distributed language model. *Advances in neural information processing systems*, pages 1–8, 2009.

[29] Kevin P. Murphy. *Machine Learning A probabilistic perspective*. The MIT Press, 2012.

[30] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. PATTY: a taxonomy of relational patterns with semantic types. *EMNLP*, pages 1135–1145, 2012.

[31] P Pantel and D Lin. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international ...*, pages 613–619, 2002.

[32] Patrick Pantel and Rahul Bhagat. ISP: Learning inferential selectional preferences. *ACL*, (April):564–571, 2007.

[33] F Pereira, N Tishby, and L Lee. Distributional clustering of English words. In *ACL*, pages 183–190, 1993.

[34] H Poon and P Domingos. Deep Learning for Semantic Parsing. Technical report, 2013.

[35] Hoifung Poon and Pedro Domingos. Unsupervised semantic parsing. In *EMNLP*, volume 1, pages 1–10, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

[36] M Richardson and P Domingos. Markov logic networks. *Machine learning*, pages 107–136, 2006.

[37] Sebastian Riedel and Limin Yao. Relation Extraction with Matrix Factorization and Universal Schemas. *Proceedings of NAACL- ...*, (June):74–84, 2013.

[38] Sebastian Riedel, Limin Yao, and A McCallum. Modeling relations and their mentions without labeled text. *Machine Learning and Knowledge ...*, pages 148–163, 2010.

[39] Sebastian Riedel, Limin Yao, and Andrew Mccallum. Modeling relations and their mentions without labeled text. *Machine Learning and Knowledge ...*, pages 148–163, 2010.

[40] YW Teh. Dirichlet Processes: Tutorial and Practical Course. *Machine Learning Summer School*, (August), 2007.

[41] Ivan Titov and Alexandre Klementiev. A Bayesian model for unsupervised semantic parsing. *ACL*, pages 1445–1455, 2011.

[42] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. *ACL*, (July):384–394, July 2010.

[43] Limin Yao, A Haghighi, Sebastian Riedel, and A McCallum. Structured relation discovery using generative models. In *EMNLP*, pages 1456–1466, 2011.

[44] Alexander Yates, M Cafarella, and M Banko. TextRunner: open information extraction on the web. In *ACL*, number April, pages 25–26, 2007.