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

Relation Extraction Using Liberalism love and beloved

December 2, 2013

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 Machine Learning

1.1 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 [12]. Recently, it has been shown in [12] and [8] 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 **??** and **??** 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.

1.1.1 Artificial Neural Networks

Before diving in to different methods of representation learning, it would be useful to briefly review the architecture, applications and learning algorithms of artificial neural networks (ANN).

1.1.1.1 Architecture and Model Definition

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 (1.1) and visually in Figure 1.1.

$$\begin{aligned} x^{l-1} &: \text{input vector of size } n \text{ and layer } l-1 \\ w^{l,l-1} &: \text{weight vector of size } n \text{ from layer } l-1 \text{ to layer } l \\ y_j^l &: \text{output value of neuron} j \text{at layer } l \\ y_j^l &= f(\sum_{i=0}^{n} w_{i,j} x_i^{l-1}) \end{aligned} \tag{1.1}$$

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

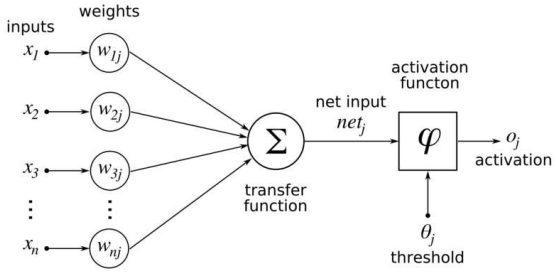
linear function $f(x) = ax + b$ where $a, b \in \mathbb{R}$

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

Figure 1.1: A model of neuron



1.1.2 Distributional Representation

In distributional semantics, the meaning of a word is expressed by the context that it appears in it [9]. 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 [10] to represent phrase and word meanings and brown clustering algorithm [6] has been shown to have impact on near to state-of-the-art NLP tasks [12].

1.1.3 Distributed Representation

Distributed representation has been introduced in the literature for the first time in [1] where Bengio et al. introduced a first language model based on deep learning methods[2]. 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 ob-

jects (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 [7], 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* [5].

A hierarchical distributed language model (HLBL) proposed by Mnih and Hinton in [11] 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 [12] 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.

1.1.4 Representation Learning of Knowledge Bases

Bordes et al. in [4] and [3] have attempted to use a neural distributed model to induce word representations from lexical resources such as WordNet and knowledge bases (KB) like Freebase. In Freebase for example, each named entity is related to another entity by an instance of a specific type of relation. In [4], 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 [4] and [3] 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 (1.2).

$$Sim(E_i, E_j, R) = ||R_k^{left} E_i - R_k^{right} E_j||_1 \tag{1.2}$$

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

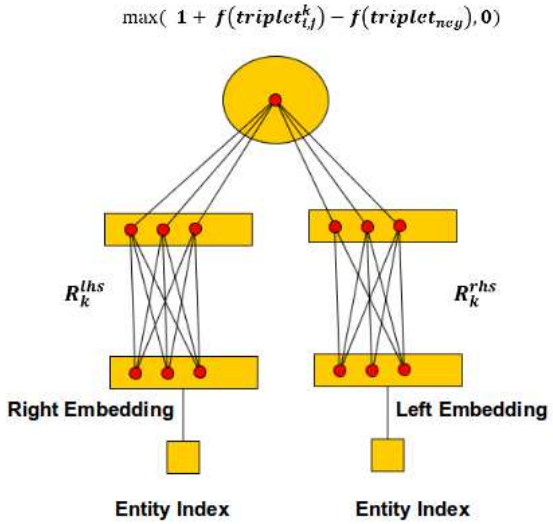
Using the idea of contrastive learning , the model will be trained to increase similarity of embeddings for a positive triple (a triple which exists in the KB) or lowering its rank among other training samples and decrease the similarity of embeddings when the relation doesn't hold (negative triple) 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 1.2 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 (1.3).

$$E'_{left} = (W_i E_i) \odot (W_k R_k) + b_{left} \tag{1.3}$$

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 asso-

Figure 1.2: Neural Distributed Model to Learn Structured Embeddings (SE)



ciated score for the triple can be calculated by dot product of E'_{left} and E'_{right} which is shown in Equation 1.4. Figure 1.3 is used by Bordes et al. to sketch SME-Bil model. 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 **??**.

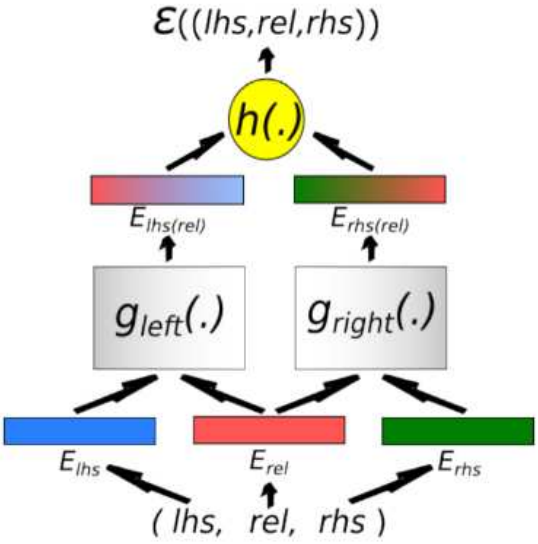
$$Sim(E_i, E_j, R_k) = -E'_{left}E'_{right} \tag{1.4}$$

1.2 Bayesian Non-Parametric Models

1.2.1 Introduction

One of the most important contributions of Machine Learning to Natural Language Processing and Computational Linguistics is to provide variety of statistical frameworks for modeling different aspects of language. Modeling makes the data more interpretable and is trying to represent it in a compact way. From language modeling to syntax and semantics, computational models are needed to first describe and then predict the certain phenomena in language. Here we address a family of models which are trying to be less

Figure 1.3: Neural Distributed Model to Learn Structured Embeddings (SME)



dependent on human choice or trial and error schema for choosing best parameters and are called Bayesian nonparametric models (BNP).

1.2.2 Model Selection with BNP Models

A traditional approach for model selection is to have a set of models and then try to fit these models on the linguistic data by tuning and inferring the parameters of them [2]. It is more like a search in possible space of models and their parameters to find the best setting. Therefore, it is necessary to have a criterion to select the models to see which one is better explaining the data and have the prediction power. There are two main features that any criterion should consider necessarily. First is the accuracy of prediction (model fitness) and it ensures that the model is well-defined and second is the complexity of model [1,2]. Let us define these terms in a mathematical language so we can introduce a Bayesian framework for modeling. The data is often a set of data points which could be represented as vectors or is in more complex structure like a graph or an ordered sequence. For simplicity we first start with data point or vector representation of data. Let us define the data as:

$$D = y_1, y_2, \ldots, y_n \tag{1.5}$$

The model could express predictions by the likelihood which is in this form:

$$P(D|\theta, m) \tag{1.6}$$

You see that the predictions are conditioned on the model and a specific parameter in this way. In a fully Bayesian model, one should specify the range of parameters or more precisely a distribution over possible parameters should be determined beforehand [2]. This could be done by adjusting the prior and then we can incorporate the role of all possible parameters to predictions weighted by their importance. Here one can notice that the set of parameters are fixed.

$$P(D|m) = \int P(D, \theta|m)\,\mathrm{d}\theta = \int P(D|\theta, m)P(\theta|m)\,\mathrm{d}\theta. \tag{1.7}$$

One can see in this formula that the prediction is average over all parameters weighted by prior. Each model is a generative process of data which gives us a joint distribution of hidden and observable variables. Each data point is drawn from a distribution of observable variables conditioned on hidden variables (parameters of model) and hidden variables are drawn from a prior distribution [1]. To infer the posterior distribution (distribution of hidden variables conditioned on observable variables) we can reverse the generative process. Instead of generating a data point from the distribution we look for the model that more likely explains the generation of the data. [1,4] The second important aspect of model selection is measuring the complexity of the model. If we have two models with about same level of fitting, we favor the one that is simpler since it performs better generalization for unseen data. Knowingly as Occam's razor rule having less complex model helps avoiding overfitting to the observed data. Bayesian framework naturally favors simpler models by putting prior over parameters and penalizing unnecessary complexity of the model [2]. One can cleverly ask how much complexity is required for a particular data and how can we adapt our modeling to change predictions in the case of having new unseen data? The answer of Bayesian nonparametric modelling (BPN) to this problem differs to all previous models in a way that instead of comparing different models with different complexities it tries to fit a single model on the data which changes its complexity by seeing new data. While previous methods have fixed set of parameters, BNP assumes an infinite number of parameters. It is like that BNP has infinite amount of money (parameters) in the bank and whenever the data is more expensive (more complex) it spends more money to model

the data. Actually *nonparametric* could be misleading since they have infinite number of parameters but only activate a finite subset of them in the learning phase. Another point of view is to see a BNP model as information channel which parameters are bandwidth of this channel. As data grows bigger, more information should be captured by the model so it increase its bandwidth (uses more parameters) [2]. There are several families of BNP models available due to different tasks and structure of data and hidden variables. In following, we enumerate these families shortly and then we go more in depth for one of them.

These groups of models are [1,4,2]:

1. Classification and Regression task (Gaussian processes)
2. Clustering (Dirichlet processes and Chinese restaurant processes)
3. Density estimation (closed link to clustering)
4. Ordered discrete sequences data (infinite HMM)
5. Tree structured data or Hierarchical data (Adaptor grammar, infinite PCFG, Kingman's coalescent, HDP)
6. Overlapping clusters and factor analysis (Indian Buffet processes)

1.2.3 Clustering (Dirichlet processes and Chinese restaurant processes)

Bibliography

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

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

[3] 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.

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

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

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

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

[8] 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.

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

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

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

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