Leveraging Structure for Learning Representations of Words, Sentences and Knowledge Bases

Alexandros Komninos

EngD

University of York

Computer Science

January 2018

To my parents, Nicos and Elena.

Abstract

This thesis presents work on learning representations of text and Knowledge Bases by taking into consideration their respective structures. The tasks for which the methods are developed and evaluated on are: Short-text classification, Word Sense Induction and Disambiguation, Knowledge Base Completion with linked text corpora, and large-scale Knowledge Base Question Answering. An introductory chapter states the aims and scope of the thesis, followed by a chapter on technical background and definitions.

In chapter 3, the impact of dependency syntax on word representation learning in the context of short-text classification is investigated. A new definition of context in dependency graphs is proposed, which generalizes and extends previous definitions used in word representation learning. The resulting word and dependency feature embeddings are used together to represent dependency graph substructures in text classifiers.

In chapter 4, a probabilistic latent variable model for Word Sense Induction and Disambiguation is presented. The model estimates sense clusters using pretrained continuous feature vectors of multiple context types: syntactic, local lexical and global lexical, while the number of sense clusters is determined by the Integrated Complete Likelihood criterion.

A model for Knowledge Base Completion with linked text corpora is presented in chapter 5. The proposed model represents potential facts by merging subgraphs of the knowledge base with text through linked entities. The model learns to embed the merged graphs into a lower dimensional space and score the plausibility of the fact with a Multilayer Perceptron.

Chapter 6 presents a system for Question Answering on Knowledge Bases. The system learns to decompose questions into entity and relation mentions and score their compatibility with queries on the knowledge base expressed as subgraphs. The model consists of several components trained jointly in order to match parts of the question with parts of a potential query by embedding their corresponding structures in lower dimensional spaces.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisor Suresh Manandhar for constantly challenging me and motivating me through the whole period of the EngD programme. It has been a pleasure working with him and this thesis could not have been realized without his contribution. This thesis would not be the same without the review and insightful comments of my thesis examiners James Cussens and David Weir, as well as the advice of Alan Frisch.

A special thanks to the people at PurpleFrog Text: Masoud Saeedi, Sirvan Yahyaei and Mohammad Asgari. During my collaboration with them I acquired valuable skills and I became aware of the challenges of implementing research ideas into commercial software.

I am grateful to the Engineering and Physical Sciences Research Council (EPSRC) for funding this project.

I thank all my colleagues in the Artificial Intelligence group for providing a friendly environment for sharing ideas and receiving constructive criticism. In particular, I thank Nils Moenning for all the hours we have worked together during our studies in York, Marcelo Sardelich for the long discussions about research ideas and their applications, and Mudita Sharma for her support during the writing of this thesis.

Last but not least, I want to thank my parents for their endless motivation and support throughout all the years of my studies. I would never have been able to reach where I am now without them.

Alexandros Komninos
January 2018, York, England

Declaration

This thesis has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree other than Doctor of Philosophy of the University of York. This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by explicit references.

I hereby give consent for my thesis, if accepted, to be made available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Some of the material contained in this thesis has appeared in the following published or awaiting publication papers:

1. Alexandros Komninos and Suresh Manandhar. Dependency Based Embeddings for Sentence Classification Tasks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2016.

2. Alexandros Komninos and Suresh Manandhar. Structured Generative Models of Continuous Features for Word Sense Induction. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, 2016.

3. Alexandros Komninos and Suresh Manandhar. Feature-Rich Networks for Knowledge Base Completion. In *Proceedings of the 55th Annual Meeting of the Association for*

Computational Linguistics (Volume 2: Short Papers), 2017.

CHAPTER 1

Introduction

The ability of machines to understand and communicate in natural language has been a long term goal of research in Artificial Intelligence. At the time of writing this thesis, many applications of Natural Language Processing (NLP) such as search engines, question answering systems and personal assistants are used daily by millions of users. These systems perform very well on specific tasks and exhibit some properties of Natural Language Understanding (NLU). However, they still lack the capabilities of humans that allow them to understand language across domains and contexts.

A significant contribution to the success of current NLP systems comes from methods that allow better encoding of the properties of language and knowledge about the world. Acquiring and representing such information is a difficult problem that has the potential to further increase the capabilities of current NLU systems. Motivated by this, the work presented in this thesis aims to investigate methods to utilize additional knowledge in order to obtain more informative representations for specific NLU systems.

Representations in natural language processing systems have evolved a lot through time. Early systems relied on programmers to express rules about the interpretation of text. These rules were in the form of lexical patterns often encoded as regular expressions. The rule-based systems saw limited success in specific domains, but it soon became obvious that it is a method that can never handle the expressiveness and productivity of natural language. Humans seem to be able to understand language in the presence of many grammatical and lexical violations and in cases where information is only implied, something that rule-based systems do not have the ability to handle at all.

Substantial progress in NLP was achieved when statistical machine learning (ML) started to be the dominant approach. In the ML paradigm, systems learn to predict the desired outputs after being given a sample of the correct behaviour. Practitioners then need to determine which sort of information is relevant so that systems will be able to learn and generalize their responses to new inputs. Representations for machine learning systems typically consist of vectors in a space where dimensions correspond to arbitrary features. For NLP, where the raw input consists of a sequence of symbols, features can encode the presence of specific elements of that sequence. Typically, these elements are words but more structure can be encoded by defining features as larger subsequences like word n-grams. However, this quickly increases the dimensionality of the space and leads to sparsity problems and difficulty in learning.

Combining basic features to express complicated properties of the input and deciding which of those features can provide valuable information and which ones introduce noise is a laborious process called feature engineering. Besides systems that focus on predictions, such as classifiers, ML can be used to learn new representations from simple features. Such ML techniques are called Representation Learning (RL) or feature learning. The learned representations are called latent features because they do not correspond to directly measurable quantities. Latent features are used in statistical modelling to explain complex interactions between observed variables, enabling efficient modelling of complex structures. RL techniques such as clustering and dimensionality reduction can be used to mitigate sparsity from feature combinations.

In the past decade, a new paradigm called deep learning (DL) has emerged from machine learning, often based on Neural Network architectures. These models jointly learn appropriate representations and functions that operate on them, in order to make more accurate predictions for a specific task. The DL paradigm offers many advantages for modelling natural language. It encourages hierarchical representations and composition, two widely accepted characteristics of language. It enables learning by taking into account the structure of the input, contrary to shallow models where structured representations have to be expressed as sparse composite features.

An important characteristic of latent feature representations in DL models is that they can be shared across systems performing different tasks. Intuitively, a model trained to perform a task such as question classification, should also learn some behaviour relevant to question answering, since both require understanding of language to some degree. This intuition is partially realized in DL systems, where parameters of a model can be initialized by the parameters of a previously trained model performing a related task, and then fine-tuned for the new task. This method of parameter sharing is often used in neural NLP models, where word representations are initialized from pretrained representations coming from simpler models that learn to predict the structure of text. Another scenario where DL models share parameters is multi-task learning, where systems learn to perform several tasks at the same time by sharing some of their parameters and another part that is task-specific.

It should be noted that the wide adoption and success of deep learning techniques cannot be

attributed to just their representational power. These models require large computational power and massive datasets to achieve the desired level of performance. Modern hardware such as GPU based computation played an important role in the advancement of DL techniques, which are ideal for parallel computation. On the other hand, data have become abundant and human generated content is shared in wikis, collaborative knowledge bases, social media and forums. It is also important to acknowledge the contribution of open source software for the DL advancement. DL systems are highly modular and components come ready to be used and combined into new models. This speeds up development and makes possible to easily construct complex models without having to consider low level implementation details. A major factor for the ease of development is decoupling of the optimization procedure from model design. Deep learning software packages such as Tensorflow (Abadi et al., 2016) and Theano (Al-Rfou et al., 2016) can perform automatic differentiation allowing for quick experimentation with different architectures.

1.1 Context of the Thesis

This research was performed in the context of an Engineering Doctorate degree, which involves collaboration with an industrial partner. The industrial partner is PurpleFrog Text, a company specializing in search technologies and automatic content analysis of text. The applications of the methods developed in the thesis concern the unambiguous semantic interpretation of text. Contrary to full semantic parsing, we are interested to extract only limited predefined semantic aspects of text. Of particular interest is the ability to represent text in the terms of an entity-relation schema. A collection of facts expressed in such a schema is called a knowledge base (KB). Learning to map the concepts expressed in text to those in a KB allow us to understand meaning of text in the terms of the KB schema and to query the KB in natural language.

The tasks for which we develop our methods and evaluate their performance on are: Short-text classification, Word Sense Induction and Disambiguation, Knowledge Base Completion (KBC) with linked text corpora, and Question Answering on Knowledge Bases (KBQA). From an industrial point of view, an important objective of this research was developing methods that can have industrial applicability. As a consequence, major emphasis was given to developing systems that improve the state-of-the-art for the tasks considered. In addition, those methods need to be practical in implementation, requiring reasonable hardware and data. Data is the most important resource in any machine learning approach and human annotated datasets are laborious to create. A direction of the thesis is to utilize efficiently data resources by systems that combine labelled with unlabelled datasets, and utilize the output of existing NLP tools that can provide additional information about text.

1.2 Research Aims of the Thesis

The research presented in the thesis has a strong focus on engineering well performing solutions for the specific tasks. The methods used to achieve this goal are based on representation learning. In particular, we develop and evaluate techniques that use structured information in order to learn representations of words, short text and knowledge bases. For the first two tasks, we look at representations of text only and for the latter two, we look at representations of knowledge bases and their alignment with text.

The general approach we take in order to push further the performance of current systems is constructing representations that encode richer information or make explicit known properties of the data. In that sense, we use established learning techniques and architectures for learning and focus our efforts on improving the inputs of the systems. The additional structure we use is acquired from domain knowledge about the problems, which may be difficult to be discovered automatically by learning systems in the presence of finite datasets. The effect of such structure is biasing learning systems towards specific hypotheses simplifying the learning process.

Deep learning models can compose structure by defining composition operators over simple feature representations. Traditional machine learning models needed structural features to encode the input, but such features are formed by combinations of simpler ones and exhibit sparsity and high dimensionality. We want to combine the two approaches and use features that encode structure but are embedded into a low dimensional dense space. We can then combine structured features with the structured composition operators of DL models, potentially encoding different structure with each approach. An important aspect of the representations explored in this thesis is utilizing multiple views of the input data. Text can be represented as a sequence of characters, a sequence of words and a graph of typed syntactic relations between words. All three of those views have valuable information in specific cases and allow us to learn robust representations that facilitate learning solutions to the problems of interest. Similarly, entities in a knowledge base can be described by their relations to other entities, but also from basic attributes such as their text description. In the work presented, we do not commit to any of those views, but try to learn representations of structured objects considering many of them together.

Another important characteristic of the representations we are interested in is that they should be shareable between models performing different tasks, in order to capture additional information that is not present in the limited annotated data available for each individual task. Our aim is to transfer information through representations either trained on large text corpora, or trained on a related simpler task than the one we are ultimately interested in. The method we use to transfer such information between systems is typically called pretraining, i.e. using an auxiliary task to learn representations and then initialize them in a model for a different. Depending on the nature of the auxiliary task this method is called semi-supervised or transfer learning. Such methods are widely used in NLP by pretraining word representations and we hope we can generalize it to

representations of other objects.

For text, we focus on representations of words and short text such as phrases or a sentence. The structure we utilize is dependency syntax, a form of syntactic parse that has been extensively developed by the NLP community with tools that specifically focus on applications. Dependency syntax has been successfully used in many NLP tasks, and has been shown very useful for understanding the semantic relations between entities. We expect that a general method that allows utilizing such information in different models and across tasks can bring significant improvements on performance of systems for NLU.

The method we use to exploit syntactic information is by decomposing dependency graphs into features and then learning a laten representation of those features. The motivation of using such a method comes from the success of such representations for words, typically referred to as word embedding. One type of feature we extracted from dependency graphs is simple words, so we can think of the method as a generalization of word representation learning. We are interested to test if these word representations are superior to models ignoring dependency structure and if the additional dependency specific feature representations can be used in composing a representation for text, which we generally refer to as text encoding. This leads us to the first research question we seek to answer in the thesis:

Q1: Can we learn latent feature representations of dependency graph features that can be used as a means to provide syntactic information to a text encoding model?

We expect that successful encoding of syntactic properties in such representations will have many advantages for NLP systems. One advantage is that we can learn these representations without labelled data, just by observing the structure of automatically parsed text, while also being able to update them when used for a specific task. Another advantage is that similar to word embeddings they should exhibit additive compositionality properties, i.e. we can simply add the latent feature vectors of a dependency graph to get a fixed length representation that approximately encodes the properties of the whole graph without the need to learn additional parameters. Finally, these representations can be used in many different machine learning models. Throughout the thesis, we evaluate them as syntax aware representation of text in the following ways: as input features to models that do not take structure into account such as Support Vector Machines and Multilayer Perceptrons, as observed features to a probabilistic generative model, and with DL models that take different structure of text into account such as recurrent and convolutional neural networks operating on sequences.

Providing structural information via structure encoding embeddings is a method to combine traditional feature engineering and DL models. The sparsity problems of encoding combinatorial structure with discrete features is mitigated by learning low dimensional representations and reusability across tasks makes them capture more information. It also allows us to utilize insights about text features and tools to extract them that have been continuously developed over the past 20 years for NLP systems.

For the tasks that jointly need to model text and KBs, we provide to learning systems information about the structural correspondence of the two mediums. KBs represent knowledge in an entity-relation schema and text can be decomposed into contiguous word sequences of entity mentions and relation mentions. For Knowledge Base Completion and Question Answering, we bias the systems to match these two substructures and learn their semantic equivalences using different features for entities and relations. The second research question we want to answer is:

Q2: Does providing alignment information of entities and relations between text and KB improve the performance of systems that need to learn semantic equivalence between the two sources?

In our experiments on KBC and KBQA, we find that the best performing way to jointly learn semantics of the two modes is an incremental process. First learn the representation of each mode individually using the structure. Then use any available prior knowledge we have about the semantic correspondence of the two modes to learn a projection into a common latent space. This information can be annotations of questions and entries in the KB that answer them, or automatically extracting potential text mentions that express the same concepts in the KB. In general, this information is either expensive to obtain or noisy. We hypothesize that knowing structural alignments of the two modes can help the mapping procedure to a common space by expressing parts of a mode with feature representations from the other mode. For example, we can represent entities in a KB by learning a representation from the KB structure and then also construct a representation from their name by using text feature representations.

We note that an alternative to utilizing domain knowledge is taking a "from scratch" approach (Collobert et al., 2011; Zhang and LeCun, 2015). The intuition behind this approach is that powerful learning models having access to large datasets can discover all of the underlying structure in the data without the need to explicitly encode such knowledge. While discovering linguistic concepts from raw data is an interesting direction for research and can potentially enrich our linguistic knowledge, it does not always result in best performing systems. Simply using the knowledge that text is composed of words can already provide considerable benefits compared to representing text as a string of characters. We argue that from an engineering point of view, it is important to utilize available prior knowledge in order to obtain good performing systems. From a research point of view, it is worth investigating methods to incorporate this knowledge in systems and evaluate its contribution.

Throughout the thesis, we perform evaluation of systems constructed with the principles mentioned above, using publicly available benchmarking datasets and testing if our hypotheses are true. Performance of the systems is measured according to the established metrics of the particular dataset, enabling comparison with prior work. The conclusions from such experimentation are drawn by observing the performance difference between the proposed systems in this work and systems described in previously published work, as well as from targeted modifications to the proposed systems' components in order to measure their contribution. Development of NLP

systems involves many engineering decisions, and minor choices are also important in order to get a better understanding of the contributing factors of good performance. When comparing with previous work, we make the assumption that the authors performed a fair amount of optimization and report results that represent the full capabilities of systems based on their chosen design principles.

1.3 Thesis Contributions

This thesis makes the following contributions:

- An embedding method for words and syntactic dependency features based on a context definition that generalizes and improves upon previous work on sequential and dependency context definition models.
- A method to construct sentence representations capturing structural information from typed dependency graphs that can be used in a variety of short text classification methods.
- A probabilistic model for Word Sense Induction and Disambiguation based on pretrained continuous feature vectors, capable of aggregating information from multiple context views of syntactic, local lexical and global context.
- A general framework for Knowledge Base Completion that can utilize linked resources and multiple feature types. The model enables prediction about unseen facts by introducing non-linear interactions between representations of different linked resources.
- A Knowledge Base Question Answering system that learns to decompose questions into entity and relation parts and jointly reasons about their correspondence with symbols in the Knowledge Base. The system utilizes multiple representations based on different views for both the question and the Knowledge base symbols, and applies pretraining to those representations on auxiliary tasks.
- Resources of latent feature representations available for usage into production systems or for conducting further research. The resources include vector representations of word and dependency features, character n-grams, and embedded entities and relations of the Freebase knowledge base.

1.4 Chapter Overview

The rest of the thesis is organized as follows:

Chapter 2 provides background material of technical concepts in representation learning used throughout the thesis. These include basic Neural Network architectures, loss and optimizer definitions, and models for learning representations for text.

Chapter 3 investigates the impact of syntax for word representation learning in the context of short-text classification. A new definition of context in dependency graphs generalizes previous embedding models and extends them to obtain representations of dependency structures. Word and dependency embedding are used together to replace simple word representations in unordered and sequential classifiers with graph substructures that capture the syntactic properties expressed by dependency graphs. The dependency augmented representations are shown to consistently outperform simple word embedding in three categorization tasks.

Chapter 4 presents a probabilistic latent variable model for Word Sense Induction and Disambiguation that estimates sense clusters with continuous feature vectors and takes into account multiple contexts. Experiments in two SemEval datasets show that it can estimate sense distributions that correlate better with human judgements than Bayesian latent variable models with discrete features.

Chapter 5 deals with Knowledge Base Completion with linked text corpora. A model that jointly embeds structures from a knowledge graph and a text corpus by learning representations on subgraphs, where the information is combined but their types are kept distinct. Evaluation shows that the model performs better than alternatives that operate on simple structures based on triples and do not explicitly indicate differences in the source of information.

Chapter 6 presents a Question Answering system on Knowledge Bases. The system utilizes the structural regularity of entity-relation mentions in questions to learn to decompose them into two parts and compares those parts with subgraphs of the Knowledge Base. Learning of the mapping between questions and subgraphs is facilitated by multiple representations of the two objects which are pretrained on the auxiliary tasks of Entity Linking and Knowledge Base Completion. The system improves upon the performance of systems taking a pipeline approach or systems that ignore the structural correspondence between information found in text and Knowledge Bases.

Chapter 7 provides a summary of the findings of the thesis and general conclusions.

CHAPTER 2

Background and Definitions

In this chapter, we present technical background and definitions for models used throughout the thesis. In the first section, we look at Neural Network components and basic architectures. In the second section, we present models from the literature that estimate latent feature representations of words from large corpora.

2.1 Neural Networks

Neural Networks (NN) are a broad class of learning models that are defined in terms of operations between tensors of weights (parameters) and an input, with element-wise non-linearities applied to intermediate steps of the computation. NNs are often organized in layers and their operations can be described by a computation graph. They naturally learn latent representations of the input data and take into account complex structures.

Activation Functions

The non-linear functions applied between layers of NNs to increase their representational power, and to their outputs are called activation functions. Typical choices of activation functions for intermediate layers are the rectified linear unit (relu):

$$relu(x) = \max(0, x) \tag{2.1}$$

and the hyperbolic tangent (tanh):

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.2}$$

NNs are often used as probabilistic classifiers, mapping an input to a probability distribution over a discrete set of outcomes. To produce a probability, a sigmoid function is applied to the output for binary classification:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.3}$$

or a *softmax* for multiclass:

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{2.4}$$

Embedding Layer

The raw observations for most Natural Language Processing tasks consist of discrete structures such as sequences of characters or words. Since words are the most common feature in an NLP system, the number of atomic symbols that form the elements of the structure are often very large, for example all the different words in a language. Representation of atomic symbols in a vector space is done by one-hot-vectors: vectors that have one value equal to one and all others are zero. The feature space then needs to have dimensionality equal to the number of different symbols. This leads to many problems for learning and generalization.

Embedding in Neural networks is the assignment of low dimensional dense representations to discrete features. The operation an embedding layer performs is selecting a vector from the embedding matrix according to the feature's index. This operation can be expressed as the dot product of the embedding matrix with the one-hot vector of the feature:

$$h = Wx \tag{2.5}$$

where $W \in R^{n \times k}$ is the embedding matrix of n discrete features to k dimensional embeddings and x is a one-hot vector of length n.

A NN with an embedding layer can learn distributed representations of the features optimized for a given task. It is common to initialize the values of the embedding layer from pretrained models on auxiliary tasks.

Multilayer Perceptrons

The Multilayer Perceptron (MLP) applies a finite number of affine transformations to an input vector, each time followed by application of an element-wise non-linear function (Figure 2.1). The block consisting of an affine transformation and a non-linearity is often called a densely

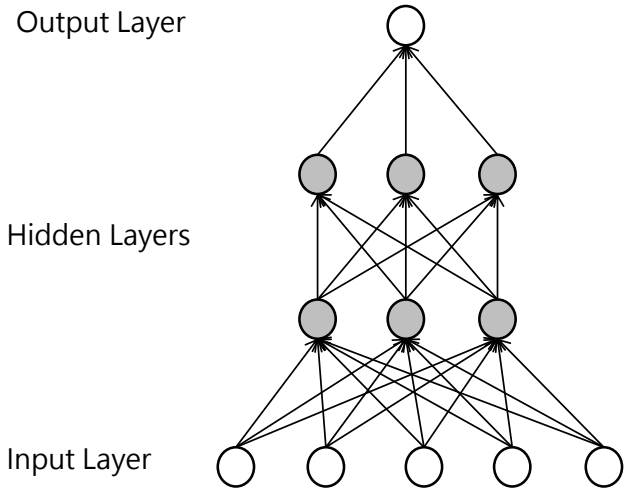


Figure 2.1: A Multilayer Perceptron with two hidden layers.

connected layer or simply dense layer. Intermediate layers are called hidden. An MLP with two hidden layers is defined by:

$$y = g(W_2 g(W_1 x + b1) + b2) \tag{2.6}$$

where $g(\bullet)$ is an activation function, W_1, W_2 weight matrices and $b1, b2$ bias vectors.

MLPs with one hidden layer are universal approximators (Hornik, 1991), i.e. they can represent any continuous function arbitrarily well for some architecture and weights. A limitation of the MLP is that it can only process input vectors of the same size, so it cannot be naturally applied to structured inputs like a sequence. Models like the neural-bag-of-words sum word embeddings to form a fixed length vector that can become the input to an MLP, similar to how bag-of-words text representations are created by summing one-hot-vector representations of words.

Convolutional Neural Networks

The Convolutional Neural Network (CNN) is a model that operates on structured input to produce a representation. Common variations are the one dimensional CNN that considers sequential structure of the input and is widely applied in NLP, and the two dimensional CNN that considers grid structure and is very common in Computer Vision. A CNN layer consists of a set of weight

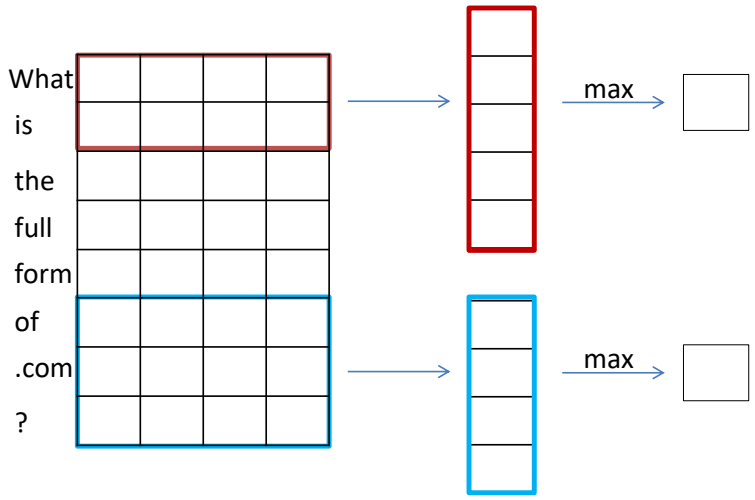


Figure 2.2: A one dimensional Convolutional Layer with two filters.

tensors called filters or kernels that have the shape of a substructure of the input. The filters act as pattern detectors that "scan" the input structure and produce values according to how closely the pattern is matched. For 1d-CNNs applied on word sequences, the filters look for patterns over word n-grams (Figure 2.2).

Considering an input sequence of length l where each element is a feature vector $\mathbf{x_t} \in R^k$ forming a matrix $\mathbf{X} \in R^{l \times k}$ and a 1d-CNN with filters $w_i \in R^{h \times k}$, the filters are applied to all possible areas of the input to produce a scalar output via a dot product:

$$c_{ij} = g(w_i \cdot X[j : j + h - 1] + b_i) \tag{2.7}$$

The output of each filter is reduced in dimensionality by a pooling operator. Here we consider a global max pooling which chooses the output with maximum value for each filter:

$$v_i = \max_{i,\bullet}(c_{ij}) \tag{2.8}$$

Applying global max pooling gives an output of fixed size equal to the number of filters. This is a single layer CNN that can only detect local patterns in the data. It is possible to apply pooling over smaller regions and still preserve structural information so that another layer of filters can be applied, an architecture that is known as deep CNN.

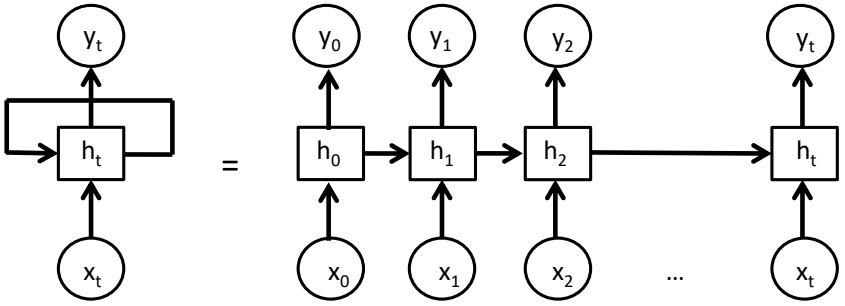


Figure 2.3: A Recurrent Neural Network.

Recurrent Neural Networks

Recurrent Neural Networks (Elman, 1990) naturally operate on sequential data by updating a hidden state vector with each input (Figure 2.3). Let's consider again a sequence of length l where each element is a feature vector $\mathbf{x_t} \in R^k$. A simple RNN updates the hidden state with:

$$h_t = g(Wx + Uh_{t-1} + b) \tag{2.9}$$

Simple RNNs exhibit problems during training because their gradient can become very small (vanishing gradient) or very large (exploding gradient). While the exploding gradient can be overcome by clipping large values, the vanishing gradient problem prohibits simple RNNs to learn long-range dependencies. The Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) uses a different parametrization of the updates that overcomes the problem. The LSTM recurrent units consist of a memory cell c and three gates i, o and f. The updates are given by the following set of equations:

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ \tilde{c_t} \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \cdot \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \tag{2.10}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c_t} \tag{2.11}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.12}$$

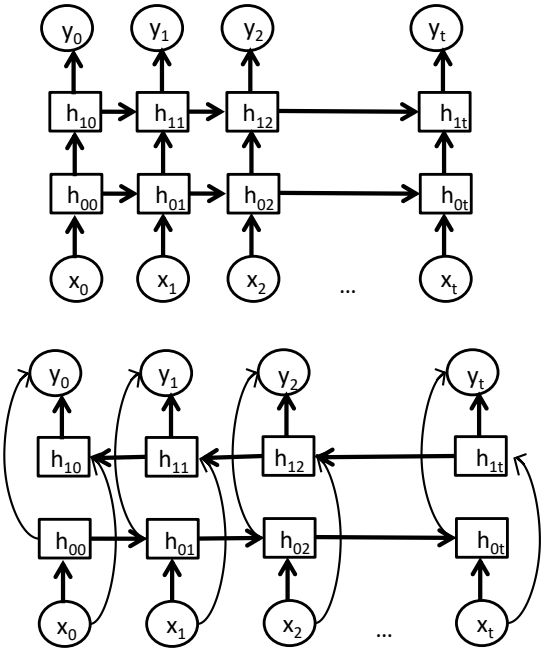


Figure 2.4: A deep and a bidirectional RNN.

where $W \in R^{4k \times 2k}$ and $\odot$ is element-wise vector multiplication.

An RNN can be used for structured prediction by applying a dense layer on top of h, e.g. a language model that predicts the next word at every step (Mikolov et al., 2010). RNNs are also very effective at producing a fixed length vector encoding of the whole sequence. This is either done by using the final updated hidden state h_T or by aggregating the hidden states at every step, e.g. taking their mean. Another option is to compute a mixture of the hidden states, where the mixture weights are estimated by another network, potentially using some context information. Such a mechanism is called attention (Bahdanau et al., 2014). Other ways to extend RNNs is stacking them together to make deep RNNs where the hidden state of one RNN becomes the input to another. It is common in text processing to use bidirectional RNNs: two different RNNs encode the sequence from different directions and at each step the hidden state is taken to be the concatenation of the states of each RNN (Figure 2.4).

Loss Functions

The choice of loss function is largely problem specific. Some commonly used loss functions are:

Binary cross-entropy Defined for a network that outputs a probability:

$$L(\theta) = -y_i \log(p_i) - (1 - y_i)(\log(1 - p_i) \tag{2.13}$$

where $y_i \in 0, 1$ is the label and p_i the prediction of the ith sample.

Categorical cross-entropy Defined for a network that outputs a probability distribution over C classes:

$$L(\theta) = -\sum_{c=1}^{C} y_{i,c} \log(p_{i,c}) \tag{2.14}$$

Labels and predictions are sample and class specific.

Mean Square Error Commonly used loss for continuous output:

$$L(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i) \tag{2.15}$$

where N is the number of samples.

Pairwise Ranking Margin Loss Used to score an item higher than another by a margin. It is commonly used in unormalized (non-probabilistic) predictions to rank observed items from a dataset higher than noise in order to learn latent representations:

$$L(\theta) = \max(0, m - y_i + y_j) \tag{2.16}$$

where m is the margin and y_i the item to be ranked higher than y_j.

Optimization

Optimization of Neural Networks is performed by some variation of stochastic gradient descent (SGD). SGD computes the gradient of parameters with respect to the loss and updates them by taking a step towards the direction of the gradient (Figure 2.5). Contrary to Gradient Descent, the updates are performed after a few samples, the mini-batch, rather than the whole training set. The parameter update rule for simple gradient descent is:

$$\theta_i = \theta_i - \alpha \nabla_{\theta_i} L(\theta_i; (x^{[i:i+m]}, y^{[i:i+m]})) \tag{2.17}$$

where m is the size of the mini-batch, α the learning rate and L the loss function being minimized. The gradient of each parameter in a network can be effectively computed by the backpropagation algorithm.

There are several modifications to the simple SGD optimizer that can converge faster to a minimum or can handle some problematic cases like saddle points or long valleys in the solution space. Momentum is one such modification that keeps track of previous updates and adds a

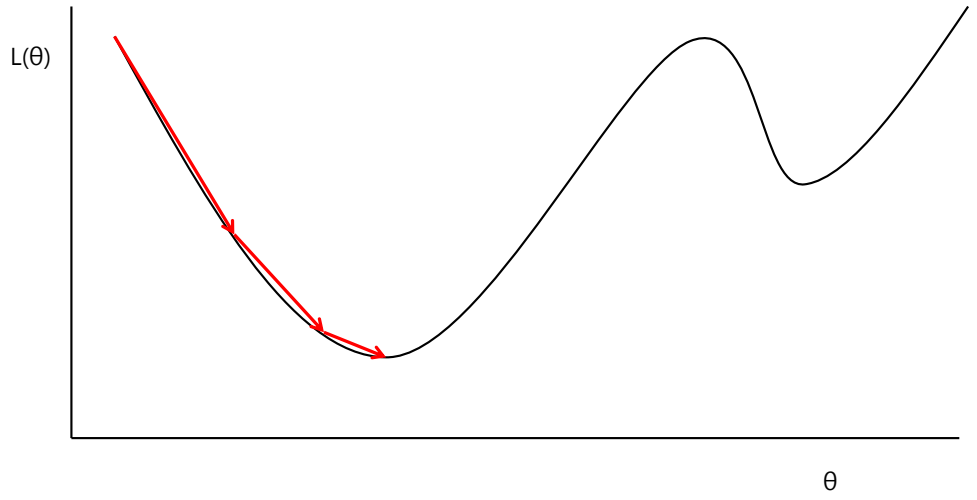


Figure 2.5: Depiction of parameter updates of Stochastic Gradient Descent approaching a local minimum.

portion of them to the gradient:

$$u_t = \gamma u_{t-1} + \alpha \nabla_\theta L(\theta)\theta = \theta - u_t \tag{2.18}$$

where γ is a hyperparameter with typical values around 0.9.

Other modifications use a different learning rate for each parameter, computed as a function of the history of that parameter's updates. Adagrad (Duchi et al., 2011) performs the following update:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t + \epsilon}} \odot \nabla_\theta L(\theta_t) \tag{2.19}$$

where G_t is a diagonal matrix containing the sum of squares of the gradient up to step t and ϵ a small number to avoid division by zero. Adagrad applies learning rate decay according to how much a parameter has been updated in the past. Adadelta (Zeiler, 2012) is a similar method that applies a less aggressive decay by reducing the contribution of past gradient sum of squares. At every gradient computation it updates:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \tag{2.20}$$

$$g_t = \nabla_\theta L(\theta_t) \tag{2.21}$$

where $E[g^2]_t$ is the mean of square gradients up to step t. It also keeps track of the square averages of previous updates:

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2$$
$$\Delta\theta_t = \theta_{t+1} - \theta_t$$

It then computes the following update:

$$\Delta\theta_t = -\frac{RMS(\Delta\theta_{t-1})}{RMS(g_t)} g_t \tag{2.22}$$
$$RMS[\bullet]_t = \sqrt{E[\bullet]_t + \epsilon} \tag{2.23}$$

where RMS is the root mean square function. Adadelta does not make use of a learning rate.

Adam (Kingma and Ba, 2014) is an SGD variant that combines an adaptive learning rate mechanism with momentum. The updates of parameters are:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$u_t = \beta_2 u_{t-1} + (1-\beta_2) g_t^2$$
$$m'_t = \frac{m_t}{1-\beta_1}$$
$$u'_t = \frac{u_t}{1-\beta_2}$$
$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{u'_t + \epsilon}} m'_t$$

where m'_t and u'_t are bias corrected momentum and speed terms. Recommended values for β_1 and β_2 are 0.9 and 0.999 respectively.

Regularization

Neural Networks are usually overparameterized, the number of weights is larger than the number of samples, and they easily overfit the training dataset. To ensure generalization to unseen samples they need to be regularized. It is typical to monitor performance in the validation set and stop training after the network starts to overfit, a method often called early stopping. A more principled method of regularization is adding an extra term to the loss that penalizes the norm of the weights:

$$L_{reg}(\Theta) = L(\Theta) + \lambda ||\Theta||_{l1/l2} \tag{2.24}$$

where λ is a hyperparameter that weighs the two objectives. This method of regularization restricts the weights to become arbitrarily large and biases the network towards simpler solutions. A similar method of regularization puts an explicit constraint over the norm of weight vectors

$||w|| < c$ where c is usually 3 to 5. The norm-constraints are enforced after every update by appropriate normalization of the weight vectors.

Another method to prevent overfitting is applying dropout to hidden layers (Srivastava et al., 2014). During training, at every step of GD values of the hidden layer where dropout is applied are set to 0 with probability p. This has the effect of removing a part of the NN for that iteration. During prediction, the input to those layers is rescaled such that on average it would be equal to the input with dropout. Dropout prevents the neurons to co-adapt and encourages them to learn robust features of the input. Another argument explaining dropout's effect on generalization is that the network exhibits properties similar to model ensembles, as it effectively trains an exponential number of smaller networks.

2.2 Distributed Word Representations

Estimating word representations from text corpora has been the focus of a lot of research in NLP. Word representations have been shown to capture semantic and syntactic properties of words and can be used to estimate similarity of text pieces beyond simple keyword matching. (Turney et al., 2010; Levy et al., 2015). Observed feature representations (also called explicit) are obtained by defining a notion of co-occurrence between words and context features. In the most typical case, context features are also words in a predefined neighbourhood of the target word. Using words as features to create word representations conforms with the distributional hypothesis in linguistics, stating that the meaning of words is determined by their context (Harris, 1954). Word representations exhibit properties of additive compositionality: adding two word vectors gives a vector that captures the meaning of a phrase, and can encode relations with simple algebraic operations, such as $v(Paris) - v(France) + v(Germany) \simeq v(Berlin)$ and $v(king) - v(man) + v(woman) \simeq v(queen)$ (Levy et al., 2015; Mikolov et al., 2013; Pennington et al., 2014).

Low dimensional distributed representations have several advantages over observed feature ones. According to Bengio et al. (2013), a good representation should disentangle the factors of variation, and observed word features are usually highly correlated. Other advantages are computational efficiency, removing noise and reducing the total number of parameters when used in another system.

Word embedding techniques learn a distributed representation of words as low dimensional vectors of real numbers. Several methods start by creating word feature vectors and then applying dimensionality reduction techniques to embed the explicit vectors into a lower dimensional latent feature space. In this setting, creating the feature vectors and defining an optimization objective to reduce their dimensionality are the two most important design decisions. Another way of estimating such vectors is from the embedding layer of a Neural Network as a by-product of using word features in any NLP task. Language modelling is a general choice since it does not

require annotated text and aims to model word structure. Some well established methods to obtain latent feature representations for words are the following:

Matrix Factorization

A method to reduce the dimensionality of word representations is by application of matrix factorization. By stacking the word feature vectors we end up with a matrix where each row correspond to a word and each column to a feature. Matrix Factorization aims to to decompose the matrix into the product of two or more matrices. By choosing the number of dimensions of the new matrices to be lower than the original, the factorization becomes approximate and the resulting matrices provide a low dimensional embedding of the word and feature vectors. Matrix factorization methods assume that word feature vectors have already been extracted from the corpus. It is common to apply a weight function to co-occurrence counts of words to transform the values of the features. It has been shown in several studies that a good choice of weight function is positive pointwise mutual information (PPMI):

$$PPMI(x, y) = \max(\log \frac{p(x, y)}{p(x)p(y)}, 0) = \max(\log \frac{Nc(x, y)}{c(x)c(y)}, 0) \tag{2.25}$$

where N is the total number of tokens in the corpus and $c(\bullet)$ are counts.

An established method of factorization is Singular Value Decomposition (SVD), where the matrix is factorized into three matrices $X_{n,m} = U_{n,n}S_{n,m}V_{m,m}$, where U and V are orthonormal and S is a diagonal matrix of the singular values sorted from larger to smaller (additional zero vectors are stacked in rows or columns to make the shape (n, m)). SVD is guaranteed to exist for any matrix of real or complex values and results in an exact factorization. Truncating the matrices to $U_{n,k}, S_{k,k}, V_{k,m}, k < m$ results in a low rank approximation of the original matrix. This approximation is optimal under the least squares loss. The truncated version can be estimated directly through optimization of the least squares loss by any optimizer (e.g. SGD):

$$\arg\min_{U,V} ||X - UV||_2 \tag{2.26}$$

In the work of Lebret and Collobert (2013), the euclidean distance in the loss was replaced by the Hellinger distance, a distance of probability distributions:

$$H(P, Q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i=1}^{k}(\sqrt{p_i} - \sqrt{q_i})^2} \tag{2.27}$$

The entries of the co-occurrence matrix were normalized to contain estimates of conditional probabilities $p(w|c) = \frac{n(w,c)}{n(c)}$.

Another matrix factorization method is Nonnegative Matrix Factorization (NMF)(Lee and

Seung, 1999). In NMF, the orginal matrix X and the resulting decomposition matrices U and V are nonnegative. There are several methods to estimate NMF, such as using alternating least squares and enforcing the non-negativity constraint after every iteration or the projected gradient method that performs gradient descent and projects the updated matrices to the closest point that satisfies the constraints. NMF has the advantage of interpretability since it behaves like a mixture model, but it does not perform as well as SVD based factorizations for semantic tasks (Van de Cruys, 2010).

GloVe

Another word embedding method based on matrix factorization is GloVe (Pennington et al., 2014). GloVe extracts a word-word co-occurrence matrix from co-occurring words in a 10 word window, weighted according to their distance from the middle word. It then computes the following low rank factorization:

$$L(\Theta) = \sum_{i,j}^{V} f(X_{i,j})(w_i \cdot w_j + b_i + b_j - \log X_{ij})^2 \tag{2.28}$$

$$f(x) = (\frac{x}{x_{max}})^a \quad \text{if} \quad x < x_{max} \quad \text{else} \quad 0 \tag{2.29}$$

where $x_{max} = 100$ and $a = 0.75$. This is an instance of weighted matrix factorization, where the loss depends on a function of a word's frequency. Glove's objective is constructed such as the word vectors will exhibit analogical reasoning properties.

Skip-gram and CBoW

A very popular method to estimate word embeddings are the skip-gram and Continuous Bag of Words (CBoW) models (Mikolov et al., 2013), which are included in the word2vec tool. Skip-gram reads a text corpus word by word and extracts pairs of words that co-occur inside a window of predefined length.

Skip-gram is a log-bilinear probabilistic classification model. Given two sets of discrete features called targets and contexts, each feature is associated with a real valued vector. The vectors of targets and the vectors of contexts form two weight matrices, which are the parameters of the classifier. For a sample set of target and context pairs (t, c), skip-gram aims to optimize the parameters in order to model the probability of observing the target given the context:

$$p(t|c) = \frac{e^{t \cdot c}}{\sum_{t'} e^{t' \cdot c}} \tag{2.30}$$

Skip-gram can be considered as an MLP with one hidden layer without the application of a non-linear activation. The input is a one-hot vector representation of the context, an embedding

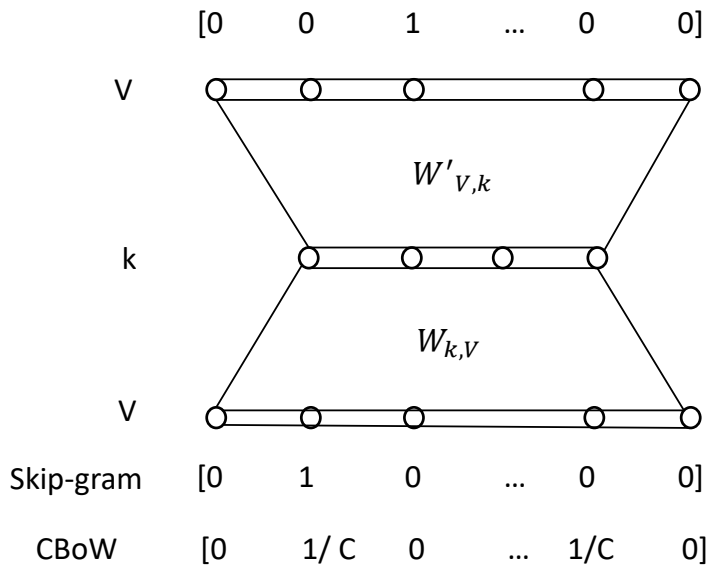


Figure 2.6: Skip-gram and CBoW as Neural Networks.

layer is applied to give the latent representation of the word followed by an output layer with a *softmax* activation to predict the probability distribution over the targets (Figure 2.6). This formulation as multiclass classification is very inefficient as it requires normalization with all the items in the context set, which in case of words can be very large. A different parametrization is using the "one-vs-rest" approach and applying binary classification. In this case, the model can be considered to model the probability that a (t, c) pair is included in the dataset:

$$p(D = 1|t, c) = \sigma(e_t \cdot e_c) \tag{2.31}$$

The weights of the model can be trained by considering a set of k negative samples (t', c) for each observed (t, c), which are pairs not observed in the data, and minimizing the following loss:

$$-\log(\sigma(e_t \cdot e_c)) - \sum_{i=1}^{k} \sigma(e_t \cdot e_{c_i}) \log(-\sigma(e_t \cdot e_{c_i})) \tag{2.32}$$

This method of word embedding is called Skip-gram with negative sampling. In practice, the negative samples are randomly selected. It was experimentally determined by Mikolov et al. (2013) that sampling from the unigram distribution of words raised to the power of 3/4 is a good

choice for choosing negative samples.

Skip-gram is a very efficient online method that does not require extracting the co-occurrences of target and contexts in advance. In addition, it performs very well in most benchmarks testing semantic and syntactic properties of word representations (Levy et al., 2015). The additive compositionality of skip-gram was explained by Gittens et al. (2017). It was also shown by Levy and Goldberg (2014a) that skip-gram with negative sampling is implicitly factorizing a word-word co-occurrence matrix with values related to PPMI:

$$\max(0, \log(\frac{p(t,c)}{p^{3/4}(t)p(c)}) - \log k) \tag{2.33}$$

where k is the number of negative samples and the power of $3/4$ comes from the negative sampling distribution. The factorization is weighted as the loss of skip-gram depends on the frequency of (t, c) pairs.

CBoW is a small modification to Skip-gram that results in fewer evaluations of the network to make a pass through a corpus. In CBoW, all of the context vectors in the co-occurrence window are first averaged and then used to make a target prediction:

$$p(t|c_1, c_2, ..., c_n) = \frac{e^{t \cdot \frac{1}{C} \sum_{c \in win} c}}{\sum_{t'} e^{t' \cdot \frac{1}{C} \sum_{c \in win} c}} \tag{2.34}$$

Similarly to skip-gram, training can become efficient by replacing *softmax* classification with a binary classifier and negative sampling.

Word Embedding via Neural Language Modelling

A different approach to obtain word embeddings is by the embedding layer of a Neural Network optimized to perform a task with words as features. A task to obtain generic word representations that do not capture a specific aspect of word semantics is language modelling. Language modelling is the prediction of a word given the previous n words. By fixing n to a specific number we can concatenate word vectors of the previous n words and predict the next with an MLP (Bengio et al., 2003) having a *softmax* output activation and a single hidden layer with a $tanh$ activation:

$$p(w_t|w_{t-1}, w_{t-2}, ..., w_{t-n}) = MLP([w_{t-1}; w_{t-2}; ...; w_{t-n}]) \tag{2.35}$$

Alternatively, we can use an RNN (Mikolov et al., 2010) and condition the prediction to all of the preceding words.

$$p(w_t|w_{t-1}, w_{t-2}, ..., w_1) = softmax(RNN(h_t)) \tag{2.36}$$

The log-bilinear language model proposed by Mnih and Teh (2012) is similar to CBoW but

uses a position specific weight matrices to estimate a context distribution that takes sequential information of n previous words into account:

$$s = \sum_{i=1}^{n} W_i c_i$$
$$p(t|s) = \frac{exp(t \cdot s)}{\sum_{t'} exp(t' \cdot s)}$$

The model is also similar to the MLP language model without the non-linearity in the hidden layer.

Contrary to Skip-gram and CBoW models, a language model aims to produce accurate probability estimates of the next word in a sequence. Because of that, the objective cannot be simplified to the extent that it does not approximate well enough the output distribution, making language models expensive for word representation learning. Collobert et al. (2011) used the MLP based language model architecture only to estimate word embeddings and ignoring the quality of the output. To make training efficient they removed the *softmax* layer and trained the network with unormalized scores and the pairwise ranking margin loss.

CHAPTER 3

Dependency Feature Embeddings for Short-text Categorization

In this chapter, we will look at methods that jointly embed words and syntactic features in a latent space, in order to be used as representations for short-text categorization. The chapter is divided into two parts. In the first part, we propose a method to utilize the structure of dependency parse graphs to obtain latent feature representations of graph features by means of the skip-gram objective. The proposed method extends definitions of syntactic context used in prior work to better capture semantic and syntactic properties in a word embedding framework. Word representations from the proposed skip-gram model are compared with versions utilizing other context definitions in predicting word similarity. In the second part, we investigate methods that can utilize both word and syntactic feature embeddings derived by the model to represent substructures of dependency graphs in a continuous space. These syntactic representations can be used with any type of classifier for short-text categorization. Experiments in three different short text categorization tasks show that the embeddings obtained by the extended definition can provide valuable information to the classifiers, which increases performance compared to baselines.

3.1 Introduction

Syntactic representations play in an important role in applications requiring some form of natural language understanding. Syntactic analysis reveals the hierarchical structure of text and provides information about the types of relations between words. They are often considered as an intermediate step to bridge the gap between raw text and semantic representations. In this chapter, we

look at the role of dependency syntax for two fundamental problems in NLP applications: word representations and text representations for short-text categorization. We evaluate our models in three categorization tasks: sentiment analysis (Socher et al., 2013a), question classification (Li and Roth, 2002), and relation classification between pairs of nominals (Hendrickx et al., 2009).

A dependency graph is a representation of a sentence with a directed labelled graph, where the nodes correspond to tokens of the sentence and edges to binary grammatical relations between the tokens. Most dependency parsers restrict the graph to be a tree. Tokens in dependency grammars of English include words and punctuation. There are several different taxonomies of grammatical relation types for dependency graphs. The one we use throughout this thesis is the Universal Dependencies (De Marneffe et al., 2014). Universal Dependencies were developed to be applicable across different languages and to facilitate NLP applications. Much of the value of dependency parsing for NLP applications can be attributed to the development of representations that target NLP applications and to the existence of fast and accurate parsers (De Marneffe and Manning, 2008; Chen and Manning, 2014).

Word embedding techniques learn a distributed representation of words as low dimensional vectors of real numbers. A common application of word embedding is using them as input features for another system or initializing parameters of neural network based models for natural language understanding tasks (Collobert et al., 2011; Kim, 2014). In this setting, they replace local high dimensional representations of words (one-hot vectors), achieving better generalization in many tasks (Turian et al., 2010). By focusing on this application of word embedding, we investigate learning techniques that can have a great impact in the end tasks of short-text categorization. In principle, word embedding techniques can be applied to obtain a latent representation of any discrete feature. In NLP tasks, where the input is typically a structured object of discrete symbols such as a word sequence or a dependency graph, it is desirable to obtain latent representation of features other than words.

Estimating latent word representations involves two main design decisions. The first is defining a notion of co-occurrence between words and context features. The second decision is choosing an optimization objective to embed the explicit feature representations into a lower dimensional space. In this chapter, we are only concerned about the first component, and adopt the widely used skip-gram loss with negative sampling (Mikolov et al., 2013) as the optimization objective.

It is common for word embedding models to define co-occurrence context as a window around words. This context definition makes estimation of word representations from huge corpora very efficient but sacrifices most of the structural information in text. To better utilize structure and additional features, modifications that better encode sequential or syntactic information about text have been proposed in prior work (Levy and Goldberg, 2014b; Ling et al., 2015). The structured versions of skip-gram have been reported to have some different properties than the window-based versions when word similarity is measured in vector space. In particular, word em-

beddings from structured context models better capture properties of functional similarity, where similar words can be used as substitutes for each other, as opposed to relational similarity in window-based ones, where similar words are related by topic. A by-product of word embedding models with structural features is that they also produce latent representations of the structural features themselves. The properties of these feature representations have not been explored and they are usually treated as auxiliary features that only facilitate learning of word representations.

In this work, we built upon the dependency based skip-gram model of Levy and Goldberg (2014b) (LG) and extend it to better utilize the structure of dependency graphs. The LG skip-gram defines a set of dependency context features and extracts target-context pairs between words and those features. In the proposed extended dependency based skip-gram (EXT), we consider target-context co-occurrences between pairs of words, words and dependency features, and between dependency features themselves. This scheme results in word embeddings that share properties between window-based models and simple dependency-based ones like LG. In addition, the EXT model treats the dependency features similarly to words and not as auxiliary context and uses them to provide syntactic information to short-text classifiers.

The use of syntactic features into sentence classifiers has a long history in NLP. The inability of bag-of-words sentence representations to capture the semantics of phrases has been a noticeable problem for many tasks of interest. In sentiment analysis, negation and modifiers can shift the valency of sentiment oriented words (e.g. "not good"), changing the sentiment label of the text. In relation classification, relations are usually directed (e.g. "LOCATION contained_in LOCATION") and taking into account the order of the arguments is crucial to capture the correct extraction. Word n-grams can provide useful information in some cases but many interesting phenomena can occur between words far from each other in sequence space (e.g. "not very good", "York is a historic walled city located in North Yorkshire, England"). In the graph structure provided by dependency syntactic parsers, word relations can occur between words far in sequence space.

Neural models such as CNNs and RNNs are able to construct a compositional representation of text respecting its sequential structure. Extensions of those models have been proposed that compose sentences given a graph representation of text acquired by syntactic parsers (Socher et al., 2013a; Tai et al., 2015; Ma et al., 2015) A limitation of proposed composition networks operating on dependency graphs is that they do not take into account the dependency type information. Using different weight matrices for each type of dependency would result in about 10 to 100 times more parameters depending on the relation taxonomy, making training very difficult for annotated datasets of the size typically used for short text classification (Bastings et al., 2017). We use a different approach where syntactic information is provided only through dependency feature embeddings. This approach is not orthogonal to using tree-structured models and the two of them could be applied together. An advantage of providing syntactic information through embeddings is that large amounts of automatically parsed textual data can be utilized in order to

learn representations of dependency types.

The methods and experimental evaluation in this chapter aim to answer the first research question: if we can encode syntactic properties in latent feature representations and use them as a method to provide explicit syntactic information in text encoders. We hypothesize we can learn representations of features from a dependency parse graph similar to the way we learn representations of word embeddings from word sequences, and that they exhibit properties of additive compositionality while also being compatible with different text encoding architectures. We also expect that syntactic structure can benefit the representations of word embeddings. By using an unsupervised method to learn such representations we can use large datasets and reuse the features in different tasks.

We evaluate the properties of word and dependency feature embeddings in word similarity and short-text classification. Evaluation of embeddings on similarity datasets is not considered representative of their usefulness in downstream applications (Faruqui et al., 2016). We show that the proposed EXT skip-gram embeddings have mixed properties between a window-based and the simple LG skip-gram variant when considering word similarity, and that they provide significant benefits when considered as features for sentence classification. In addition, we show that providing explicit syntactic information coming from parsers through latent syntactic feature representations can improve the performance of classifiers for short-text and can be used in different text classification architectures.

3.2 Related Work

Structural Word Representations

Estimating word representations with additional features besides word co-occurrence has a long history in NLP. Early work used dependency syntactic context features to construct word representations as sparse observed feature vectors based on co-occurrence statistics for computing word similarities with applications to automatic thesaurus construction (Lin, 1998; Grefenstette, 2012), but without comparing them to other methods. Padó and Lapata (2007) evaluated several designing choices for the construction of word vectors with observed features from paths in dependency trees on synonymy detection, semantic priming and word sense disambiguation, and found that representations based on syntactic features outperformed those based on word features. The distributional memory of Baroni and Lenci (2010) is formed by a weighted third order tensor of (word, link, word) triples, where dependency relations were considered as links. Collapsing the tensor into different matrices was shown to be useful for several lexical semantic tasks such as word similarity and extracting semantic analogies between pairs of words. They concluded that the semantic information captured in the tensor is enough to compete with state-of-the-art systems specifically designed for each task.

In the context of neural word embedding, several modifications have been proposed to simple window-based models that can utilize structure provided by parsers. One such model is the dependency based skip-gram of Levy and Goldberg (2014b) which is extended in this work and used as a baseline in the experiments. The dependency based skip-gram uses the same dependency context features those of Padó and Lapata (2007), but trained with the skip-gram objective. It was only evaluated in word similarity exhibiting some different properties than a word based alternative. The C-PHRASE model of Pham et al. (2015) is a modification of the CBoW model that uses an external parser to replace windows with syntactic constituents and shown to be better for constructing sentence representation via word embedding addition than the original CBoW. The effect of syntactic context compared to window-based for skip-gram word embeddings has also been explored in the work of Melamud et al. (2016). Word embeddings trained with dependency contexts performed better when used as features for parsing and worse when used for co-reference resolution, while for sentiment analysis and named entity recognition results were similar. However, using concatenated embeddings of both context types increased performance for every task showing that they capture complementary information.

Besides syntax, there has been prior work on using other structural information or extensions to capture the sequential structure of text. Hashimoto et al. (2014) proposed a log-bilinear language model based on predicate-argument structures and report improvements on phrase similarity tasks compared to standard skip-gram. In the work of Ling et al. (2015), skip-gram and CBoW models are adapted to include position specific weights for the words inside the co-occurrence window and the resulting embeddings provide slight improvements for parsing and POS tagging tasks. In another context definition by Stanovsky et al. (2015), text was parsed with an Open Information Extraction tool that extracts triples of the form (subject, relation, object) and defined context features to train a skip-gram word embedding model as all words in the same triple with indicators of their slot. The resulting embeddings were shown better for word similarity and relatedness evaluation than window based ones, and better than dependency based ones for word relatedness evaluation.

The use of syntax in short text classification

Dependency syntactic features have been used extensively in sparse linear classifiers. Some form of syntactic features have been used in all of the three tasks we consider in this chapter: Sentiment analysis, Relation Classification and Question type classification. Typically used dependency features are composite features like words with their syntactic role (nsubj_John), paths of length one in the dependency graph (John_nsubj_visited) or untyped paths similar to word skip-grams where dependencies are used to link words regardless of their proximity in the sequence.

Defining features from larger substructures quickly leads to problems with high dimensionality and sparsity. One solution to this problem has been the usage of kernel methods (Shawe-

Taylor and Cristianini, 2004), where classifiers operate on positive definite similarity matrices between sample points, implicitly making use of very large or infinite feature spaces. Various different definitions of convolution tree kernels (Haussler, 1999; Zelenko et al., 2003; Moschitti, 2006; Moschitti et al., 2007) have been used to compute similarities between syntactic trees by recursively matching all their common substructures regardless of size. The differences in kernel definitions come from the way they recursively decompose the trees into smaller substructures and the syntactic representation used. Applications to question answering and relation extraction were shown to perform better than manually constructed feature sets, showing that the additional structural information can be beneficial when feature sparsity is mitigated.

With the rise in popularity of deep learning techniques for NLP, several extensions to CNNs and RNNs that operate on trees or graphs have been proposed in order to utilize information from syntactic parsers. These models have been shown to outperform sequence models in several text classification and parsing tasks. In the 1d-CNN model of Collobert et al. (2011), syntactic positional features from a constituency parser were used in addition to word embeddings and were shown to improve the accuracy of semantic role labelling. CNNs can be extended to operate on trees and graphs by defining filters compatible with subgraphs instead of subsequences. The dependency based CNN of Ma et al. (2015) defines several filters based on different subgraph patterns (e.g. parent-word-child, word-child1-child2, etc.) and combines them with filters operating on subsequences to encode parsed sentences. This model performs better than a standard CNN for text classification and achieved the best reported result on the TREC question classification task.

Recursive Neural Networks (RecNNs)(Socher et al., 2012, 2013a, 2011) compose binary syntactic trees from a constitency based parser in a bottom-up way. The parameterization is similar to that of RNNs, but replacing composition of the hidden state and the input token with composition of left and right child. Their usefulness was demonstrated in several applications such as syntactic parsing, paraphrase identification and relation classification. RecNNs were extended to work on dependency trees as a composition of a node with its parent and children and applied to image caption generation. The variable arity of children in the dependency tree (contrary to a binary tree) was handled by simple summation of the children word vectors to a single one. As RecNNs were one of the first deep learning models to be applied in NLP, even before the rise of popularity of RNNs, and achieved state-of-the-art results in many tasks outperforming traditional models with handcrafted features. It is not clear however if that can be attributed to usage of syntax or the representation power of deep learning. Cheng and Kartsaklis (2015) used a RecNN structured according to a sentence's parse to learn word and sentence embeddings by being trained to distinguish valid sentences from randomly distorted ones. The model performs better than an RNN operating on sequences for paraphrase detection.

The treeLSTM (Tai et al., 2015) extends standard LSTMs to compose tree structures by updating the hidden state with two weight matrices corresponding to the "left"/"right" relation for

constituency trees and "parent"/"child" for dependency trees instead of the single weight matrix corresponding to the "previous" relation in sequence LSTMs. Simple LSTMs with dependency paths instead of word sequences as inputs have shown better performance for Semantic Role Labelling (Roth and Lapata, 2016) and relation classification (Xu et al., 2015b). The graph LSTM (Peng et al., 2017) extends LSTMs to operate on directed edge labelled graphs. To avoid problems with cycles during backpropagation the graph is decomposed into directed acyclic graphs and updates are done independently. Graph LSTMs were used to jointly model the structure of a sentence as a sequence and a dependency tree for cross sentence n-ary relation extraction in the biomedical domain outperforming simple LSTMs.

The usefulness of syntax in some text classification tasks like sentiment analysis has been questioned by some studies, where deeper models operating on sequences or even unordered bag of words were shown to achieve similar results (Iyyer et al., 2015; Li et al., 2015). A task were syntactic representations have been shown to consistently outperform sequential architectures is relation classification. A possible reason for this is that most approaches make use of the shortest dependency path (SDP) between the two entities that simplifies the problem by removing redundant phrases. In addition, as the SDP has a simple sequential structure, any model designed to compose sequences can be directly applied and dependency types can be simply modelled as additional tokens between the words in the path.

3.3 Extended Dependency Based Skip-gram

A dependency parse graph is a graph G(W,D) where W is the set of all words in the sentence and D is the set of directed labelled edges between the words. Words are extracted from the sentence by a tokenizer and also include punctuation. The dependency graph can be represented as a set of triples (w, d, w'), $w \in W$, $d \in D$. Each triple can be equivalently be expressed as (w', d^{-1}, w) where d is the inverse relation, meaning a relation of the same type but opposite directionality. We extract two types of discrete features (symbols) from G, word features w, and dependency features (d, w') and (d^{-1}, w). Dependency features are composite features treated as a unit and composed of a word and a dependency edge. We express dependency features as strings using the format "dependency_word". The types of dependency edges we use are the Universal Dependencies (UD). In the UD schema, dependency types can have a subtype encoded as "type:subtype". Subtypes are function words such as prepositions that are treated both as properties of syntactic relations and as words. Examples of dependency features extracted from the sentence "She asked for a cup of coffee" would be $nsubj\_she$, $nmod : of\_coffee$, $nmod : of^{-1}\_cup$.

Our aim is to estimate latent feature representations of the discrete symbols using the skip-gram objective and dependency parsed text. We consider three variations of skip-gram based on different definitions of target-context pairs of symbols:

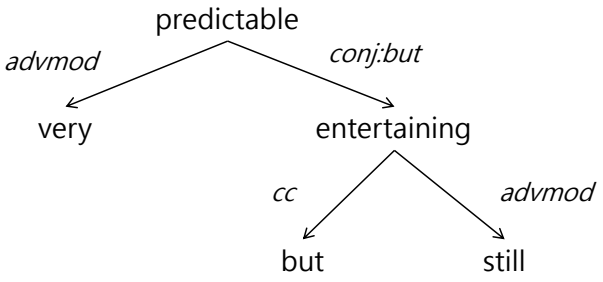


Figure 3.1: The dependency graph of a sentence.

Contexts of "entertaining"	
Win5	very, predictable, but, still
LG	conj:but^{-1}_predictable, cc_but, advmod_still
EXT	very, predictable, but, still, conj:but^{-1}_predictable, cc_but, advmod_still
Contexts of "advmod_still"	
EXT	entertaining, conj:but^{-1}_predictable, cc_but
Context feature bags for node "entertaining"	
WCF("entertaining") = {entertaining, predictable, but, still}	
DCF("entertaining") = {entertaining, conj:but^{-1}_predictable, cc_but, advmod_still}	

Table 3.1: Contexts considered by different skip-gram variants for a target word in the dependency graph of Figure 3.1, contexts of a target dependency feature by EXT skip-gram and the corresponding Word and Dependency Context Feature bags.

Window-5 based skip-gram (Win5)

This is the standard skip-gram model that considers target-context word pairs inside a window of 5 words to the right and to the left of the target word. It does not use any syntactic information and considers very limited structure expressed as proximity in sequence space defined by the window. Note that since the window size for every target instance in the corpus is uniformly sampled from the [1,5] range, the model provides a form of stochastic weighting scheme for context words according to their distance from the target word.

Skip-gram with dependency contexts (LG)

The dependency skip-gram of Levy and Goldberg (2014b) replaces context words in a window by dependency context features in the neighbourhood of a word in the dependency graph. Training of this skip-gram variant is similar to window based approaches, but each word is considered as a node in a dependency graph obtained by a parser, and embeddings are optimized to predicting their corresponding word's immediate syntactic contexts. The network's weight matrices have different shapes, where representations coming from the embedding layer weights correspond to word embeddings, while representations coming from the prediction layer correspond to dependency feature embeddings.

Extended Dependency Skip-gram (EXT)

We propose another variation of skip-gram based on dependency graphs that utilizes additional co-occurrences compared to the LG variant. The intuition is that if we consider the whole graph as a structure we can define co-occurrences that better describe the structure compared to the LG variant while still obtaining many of the word-word co-occurrences defined in the Win5 variant. In particular, considering that the graph consists of word and dependency features, we can define three types of co-occurrences: word-word, word-dependency and dependency-dependency. For the word-word co-occurrences, each target word is taken as a node in the dependency graph and then target-context pairs are extracted from all other words within distance one and two in the graph. This word-word context definition results in many common (target, context) pairs with the Win5 model, but defines a context window in the dependency graph instead of the sequence, which can filter coincidental co-occurrences while also taking into account meaningful co-occurrences of words that appear far in the sequence representation of text, but share a syntactic relation. As with the Win5 model, we apply a weighting according to distance, with words having distance one from the target counted twice. The second type of target-context pairs we extract from the graph is similar to the LG model, where each word forms a context pair with its immediate dependency features. In the LG model, only (word, dependency) pairs are formed since the two types of features do not appear in the same weight matrices. For each (word, dependency) pair we also consider a (dependency, word) one. For the third type of context definition, each dependency feature forms a target and the rest of dependency features linked to the same word node become the contexts. The three types of target-context pairs for the extended dependency skip-gram are interleaved during training. The weight matrices of this network have the same shape resulting in two embedding vectors per word and dependency feature.

We can extract all the pairs from a dependency graph by visiting each node and considering two types of context: the word context features (WCF) which are the other words connected by an edge to the target word, and the dependency context features (DCF), which are the dependency features connected to the word. The (target, context) pairs are then given by extracting all

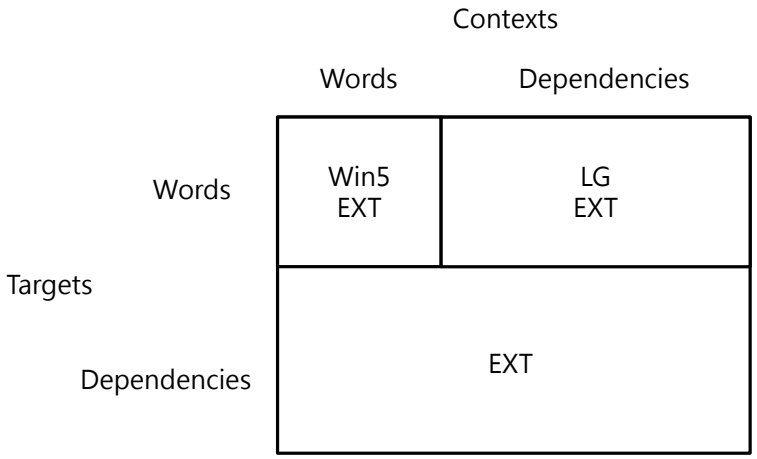


Figure 3.2: Target-Context co-occurrence types accounted for by the different skip-gram variants.

permutations of length 2 of the elements in each bag of features. A formal definition of this procedure is the following:

Given a parsed corpus as a collection of dependency graphs $G_d = (W, D)$
for each dependency graph G_d **do**
  for each word w in G_d **do**
    Extract word context feature bag:
    $WCF = \{w\} \uplus \{w'|(w, d, w') \in G_d\} \uplus \{w'|(w', d, w) \in G_d\}$
    Append all permutations of length 2 in WCF to the list of (target, context) pairs
    Extract dependency context feature bag:
    $DCF = \{w\} \uplus \{(d, w')|(w, d, w') \in G_d\} \uplus \{(d^{-1}, w')|(w', d, w) \in G_d\}$
    Append all permutations of length 2 in DCF to the list of (target, context) pairs
  end for
end for

An example of the different target-context pairs that each skip-gram variant utilizes can be seen in Table 3.1 for a dependency graph in Figure 3.1. The different types of feature co-occurrences for each model are depicted in Figure 3.2.

Implementation Details

For the three skip-gram variants, 300 dimensional versions were trained using the English Wikipedia August 2015 corpus consisting of approximately 2 billion words. Vocabularies consist of words and dependency contexts that appear more than 100 times resulting in approximately 220k

words and 1.3m dependency features. Training was done by applying negative sampling with 15 negative samples per target-context pair for 10 iterations over the entire corpus using stochastic gradient descent. The following commonly used hyperparameters (Mikolov et al., 2013; Levy et al., 2015) were applied during training: drawing negative samples according to their unigram distribution raised to the power of 0.75, linear decay of learning rate with initial $\alpha = 0.25$, and sub-sampling of target words with probability given by $p = \frac{f-10^{-5}}{f} - \sqrt{\frac{10^{-5}}{t}}$ where f is the token's frequency. Dependency parsing for LG and EXT training was done with the Stanford Neural Network dependency parser (Chen and Manning, 2014) using Universal Dependency tags (De Marneffe et al., 2014).

3.3.1 Evaluation on Word Similarity

Embeddings	WordSim-353	SimLex-999
Win5	**0.714**	0.389
LG	0.621	**0.460**
EXT	0.678	0.414

Table 3.2: Word similarity evaluation. Numbers correspond to Spearman's correlation.

As an intrinsic evaluation we look at the effect of the different contextual features for the word embeddings of the three versions in two word similarity datasets: WordSim-353 (Finkelstein et al., 2001) and SimLex-999 (Hill et al., 2015). For both datasets, we compare the cosine similarity of word embeddings for a pair of words to human judgements and report Spearman's correlation in Table 3.1. The two datasets use a different notion of word similarity for scoring. Wordsim-353 mostly captures topical similarity (or relatedness), giving high similarity to pair of words like *clothes-closet*. SimLex-999 uses a more strict version of similarity, often called substitutional similarity, where the pair *clothes-closet* has a low similarity score and pairs like *shore-coast* have high similarity. Win5 skip-gram version achieves a higher correlation for WordSim-353 compared to LG, but the results are reversed for SimLex-999. This agrees with previous research that shows that syntactic contexts correlate better with substitutional similarity judgements than using words in a window as contexts (Levy and Goldberg, 2014b). As expected, the extended model represents a middle ground solution between the two. While similarity based evaluation makes obvious that different contextual features capture different properties of words, it is not clear which kind of similarity notion is more useful when word representations are used as features for NLP tasks. We answer this question for sentence level classification tasks in the next section.

3.4 Short Text Classification with Dependency Feature Embeddings

We evaluate word embeddings derived from the different context definition on three classification datasets that have been used extensively in the NLP literature to benchmark text categorization systems: Stanford Sentiment Treebank, TREC Question Classification and SemEval 2010 Relation Classification. Besides extrinsic evaluation of word embeddings, our goal is to use the dependency features as a method to provide syntactic information about the dependency parse of a sentence to the classifiers. To this end, we test the performance of constructing dependency subgraphs with different composition operators and using them as input feature representations. The methods defined are generic and can be applied to any classifier that can make use of feature embeddings. To show the generality of the approach we apply the method with three different classification approaches: an SVM with averaged embeddings (Neural Bag-of-Words), a one dimensional shallow CNN, and an LSTM.

Our intuition behind using dependency feature embeddings to provide classification systems with syntactic information is that we expect them to exhibit additive compositionality properties similar to word embeddings. In that case, summing the dependency features extracted from a specific syntactic graph would result in a representation for the whole sentence that encodes the relations expressed in the graph. We test the performance of this approach with the SVM classifier and averaging all feature embeddings from a sentence. As the number of nodes in a dependency graph gets larger, we expect that simple summation will start losing information about the whole structure, since multiple word pairs will share the same type of dependency relation.

However, we can create representations of smaller substructures instead of the whole graph and provide them in a compositional architecture to learn the final representation of the whole sentence. This is exactly the approach we take with the 1d-CNN and LSTM that operate on sequences. Instead of having a word representation as an input, we create a subgraph representation from the node corresponding to that word and its syntactic neighbours. The nodes are still processed in the same order as the words appearing in the sentence, effectively providing information of the sequence and graph structure at the same time. This is a case where we combine two types of structure, one encoded by features and one by the model's architecture.

We expect this method to give the best results for several reasons. First, it does not enforce the structure provided by the parser, which can be wrong in some cases. Second, it mitigates the problem of long range associations of words when processing them in sequential order. For example, a shallow 1d-CNN cannot compose a representation of words that appear in distance longer than its larger filter. While a deep CNN can combine information from further regions in the subsequent layers, this comes at the cost of additional parameters and difficulties in training.

Using the dependency feature representations instead, the cost of the additional parameter estimation is mitigated by the pre-training on unlabelled text. In addition, we make use of the type of word relations, which originates from information learned by the parser from its training set.

3.4.1 Sentence Representations

We use word and/or dependency feature embeddings to create input sentence representations for the three classifiers. First, we describe how we construct the representations for the two models that operate on sequences, i.e. the CNN and the LSTM.

We begin by parsing each sentence to get its corresponding dependency graph. Each node in the graph is associated with a word w having an embedding e_w and a bag of dependency context features $d_1, d_2, ..., d_C$ with embeddings $e_{d_1}, e_{d_2}, ..., e_{d_C}$. The features associated with each node correspond to the dependency context feature bag defined in Section 3.3.3. We then create a representation x of that node using different compositions of its associated word and dependency context embeddings:

- *Words*: Using only word embeddings

$$x = e_w \tag{3.1}$$

- *Dep*: A node's representation becomes the average of its associated dependency feature embeddings:

$$x = \frac{1}{C}\sum_{c=1}^{C} e_{d_c} \tag{3.2}$$

- *Wavg*: Composition of the word and dependency feature embeddings by a weighted average that assigns equal contribution to the word and dependency context part:

$$x = \frac{1}{2}e_w + \frac{1}{2C}\sum_{c=1}^{C} e_{d_c} \tag{3.3}$$

- *Conc*: Similar to the Wavg, but dependency feature embeddings are first averaged and then concatenated to the word embedding to form a single vector:

$$x = [e_w; \frac{1}{C}\sum_{c=1}^{C} e_{d_c}] \tag{3.4}$$

  where ; is the vector concatenation operator. This method keeps the word and syntactic part separate at the expense of doubling the dimensionality.

All of the above methods are used with the LG and EXT variants to create context specific node representations. For the EXT model, both word and dependency context embeddings used

come from the embedding layer weights. The *Words* method is the only one that can be applied to the Win5 model. It is the most commonly used method to utilize word representations as features and our baseline. To make the comparison more fair for the Win5 model we include two additional variations that utilize both the embedding and prediction layer weights as an ensemble method for creating a word's representation:

- *Win5 AvgE*: Ensemble made by averaging word embeddings from the embedding and prediction layer weights of Win5 skipgram:

$$x = \frac{1}{2}(e_w + e_{w'}) \tag{3.5}$$

- *Win5 ConcE*: Another ensemble made by concatenating word embeddings from the embedding and prediction layer weights of Win5 skipgram:

$$x = [e_w; e_{w'}] \tag{3.6}$$

Embedding ensemble techniques have been reported to outperform simple word representations in some word similarity tasks (Levy et al., 2015). Since the EXT skipgram version uses weight matrices of the same shape for the embedding and prediction layer, ensemble methods like the above could also be applied, but are not considered for these experiments. Note that contrary to the dependency based models, these ensemble methods do not create context specific representations.

The dependency graph's node representations are used as a sequence with the same order of the words in the sentence to become the input for the CNN and LSTM. This input representation combines the sequential sentence structure with typed dependency graph structure. As we are evaluating the performance of pre-trained embeddings, we do not perform updates during training of CNNs and LSTMs.

For the SVM NBoW, we do not consider sequential or graph structure and aggregate all the word and dependency features together. We use all of the same methods with slightly different definitions. For a sentence s with N words and D dependency features we define:

- *Words*: $s = \frac{1}{N}\sum_{w=1}^{N} e_w$
- *Dep*: $s = \frac{1}{C}\sum_{c=1}^{C} e_{d_c}$
- *Wavg*: $s = \frac{1}{N}\sum_{w=1}^{N} e_w + \frac{1}{D}\sum_{c=1}^{C} e_{d_c}$
- *Conc*: $s = [\frac{1}{N}e_w; \frac{1}{C}\sum_{c=1}^{C} e_{d_c}]$

3.4.2 Classification Models

Descriptions and hyperparameters for the three classification methods are the following:

SVM with averaged embeddings (NBoW)

We create a sentence representation by averaging embeddings of sentence features (words and dependency contexts). This can be considered the equivalent of a Bag-of-Words sentence representation in the embedding space, hence called Neural-Bag-of-Words (NBoW). We then train a classifier by applying a Support Vector Machine with a Gaussian kernel:

$$K(x, x^{'}) = \exp(-\gamma \|x - x^{'}\|^2) \tag{3.7}$$

For hyperparameter tuning, we set parameter γ of the kernel to $1/k$, where k is the number of features (dimensionality of embeddings), and then perform cross validation for the c parameter using the standard Win5 word embeddings in the Question Classification task. This task was chosen for tuning as it stands in the middle of the three datasets considered both in number of classes and in average sentence length.

Convolutional Neural Network (CNN)

We use the simple Convolutional Neural Network of Kim (Kim, 2014) that has been shown to perform well in multiple sentence classification tasks. The network is a 1d-CNN that uses multiple filters with different sequence sizes covering different size of windows in the sentence. All hyperparameters of the network are the same as used in the original paper (Kim, 2014): stochastic dropout (Srivastava et al., 2014) with $p = 0.5$ on the penultimate layer, 100 filters for each filter region with filter regions of width 2,3 and 4. Optimization is performed with Adadelta (Zeiler, 2012) on mini-batches of size 50.

Long Short Term Memory (LSTM)

The last text classifier is a simple LSTM Recurrent Neural Network that processes the sentence left to right. The distribution of labels for the whole sentence is computed by a fully connected softmax layer on top of the final hidden state after applying stochastic dropout with $p = 0.25$. We use 150 dimensions for the size of $\mathbf{h}$, Adagrad (Duchi et al., 2011) for optimization and mini-batch size of 100. These hyperparameters were again determined by validation in Question Classification with the Win5 embeddings as features.

3.5 Evaluation

3.5.1 Question Classification

We use the TREC Question Classification dataset (Li and Roth, 2002) consisting of 5452 training questions and 500 test questions. The task is to classify each question with one of six labels

Question Type	Example
Description	What is an annotated bibliography?
Human	What actress has received the most Oscar nominations ?
Location	Where is the highest point in Japan ?
Abbreviation	What is the full form of .com ?
Numeric	How many people in the world speak French ?
Entity	What is Nebraska 's most valuable resource ?

Table 3.3: Question types and examples of the TREC Question Classification dataset.

depending on the type of answer they seek (see Table 3.3 for examples). For CNNs and LSTMs 10% of the training data were used as the validation set to pick the best model among different iterations. Classification accuracy results for each input representations and classification method can be seen in Table 3.4. We also report the state of the art result by the dependency convolutional neural network of (Mou et al., 2015). Their model consists of a convolutional neural network that operates on a dependency tree at the input layer instead of a sequence, and uses heuristics to choose the subset of nodes where pooling is applied.

Embeddings	SVM	CNN	LSTM
Win5 Words	81.4	92.8	88.4
Win5 AvgE	81.4	91.2	88.8
Win5 ConcE	82.4	92.6	90.4
LG Words	86.8	93.8	90.6
LG Dep	85.2	89.0	87.2
LG Wavg	87.2	93.4	91.2
LG Conc	84.0	94.6	92.0
EXT Words	88.4	94.2	91.8
EXT Dep	87.6	90.6	89.8
EXT Wavg	89.0	**95.0**	92.2
EXT Conc	**91.6**	93.2	**94.4**
tree CNN	**96.0**		

Table 3.4: Accuracy on 6-way TREC question classification task. Tree CNN is a CNN operating on dependency trees (Mou et al., 2015).

3.5.2 Sentiment Analysis

The Stanford Sentiment Treebank (SST) dataset (Socher et al., 2013a) has fine grained sentiment polarity scores for movie reviews on the phrasal and sentence level. The binary version of the task considers only positive and negative sentiment labels, resulting in a 6920/872/1821 split

for training/validation/testing sets. All the models were trained using only the sentence level annotations. Classification accuracies for all models are reported in Table 3.5. The state of the art for this dataset is reported in (Kim, 2014) (88.1% accuracy) using the same convolutional neural network as we do, but also utilizing the phrasal level annotations which provide about an order of magnitude larger training set. In addition, this specific configuration of the network (multichannel) uses two channels at the input layer, one updating the word embeddings during training and one that keeps them static as we do in our experiments.

Embeddings	SVM	CNN	LSTM
Win5 Words	80.1	83.5	76.1
Win5 AvgE	79.5	83.2	76.9
Win5 ConcE	80.3	82.9	77.6
LG Words	78.5	84.5	77.2
LG Dep	76.0	76.8	69.1
LG Wavg	78.9	82.0	78.6
LG Conc	79.8	82.7	79.7
EXT Words	80.5	84.1	77.6
EXT Dep	77.7	77.2	69.6
EXT Wavg	**80.6**	**84.6**	75.7
EXT Conc	**80.6**	83.5	**79.8**

Table 3.5: Accuracy on Stanford Sentiment Treebank binary classification task

3.5.3 Relation Classification

Relation	Example
Cause-Effect	wind and smoke cause flight delays
Component-Whole	timer of the device
Entity-Destination	People have been moving back into downtown
Entity-Origin	elephant descended from an aquatic animal
Product-Producer	products created by an unregulated industry
Member-Collection	essays collected in this volume
Message-Topic	citation explaining the reasons
Content-Container	lawsonite was contained in a platinum crucible
Instrument-Agency	telescope assists the eye
Other	composer has sunk into oblivion

Table 3.6: Semantic relation classes and examples of SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals dataset.

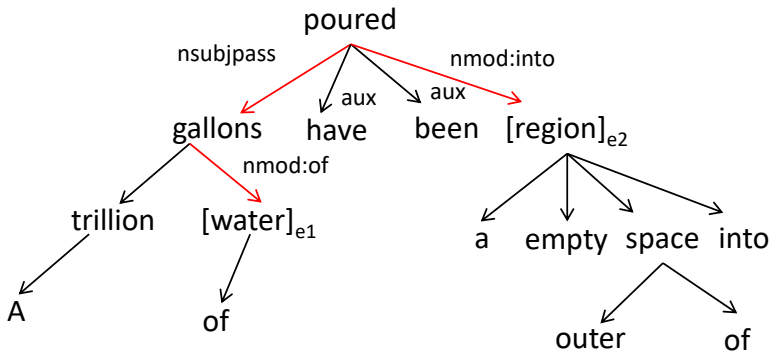


Figure 3.3: The shortest dependency path between two entities.

The dataset used for evaluation on relation classification is SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations Between Pairs of Nominals (Hendrickx et al., 2009). The task considers the classification of semantic relations between pairs of nominals into 19 classes (see Table 3.6 for examples). The classes are formed by 9 types of relations with directionality taken into account and an extra OTHER class. The dataset consists of 8000 training samples and a test set of 2712 samples. We only used the shortest dependency path (SDP) between the two nominals as the input to classifiers (Figure 3.3). In table 3.5, we report results using the official SemEval metric of macro-averaged F1-Score for (9+1)-way classification, taking directionality into account. The best reported result for this dataset is 85.6 F1-score by Xu et al. (2015a) also using a convolutional network on a sequence of word embeddings from the shortest dependency path between the pair of nominals. They also introduce negative samples during training by reversing the subject and object of the relation and WordNet features. Without using WordNet features their model achieves 84.0 F1-score.

3.6 Discussion

Our evaluation shows that dependency context embeddings can provide valuable syntactic information for sentence classification tasks using the three classification methods described. Out of the three tasks, Question Classification and Relation Identification showed great improvements when using dependency context embeddings compared to the baseline, while sentiment

Embeddings	SVM	CNN	LSTM
Win5 Words	72.23	81.60	77.30
Win5 AvgE	71.09	79.46	76.67
Win5 ConcE	72.74	81.33	78.09
LG Words	75.29	84.18	79.94
LG Dep	75.19	79.13	74.77
LG Wavg	77.61	83.17	79.69
LG Conc	**78.71**	83.41	78.57
EXT Words	74.93	83.69	80.24
EXT Dep	75.64	79.30	75.64
EXT Wavg	77.42	**84.31**	79.59
EXT Conc	78.53	83.93	**80.53**
CNN-NS-WN	**85.6**		

Table 3.7: F1 score for SemEval 2010 Relation Identification task. CNN-NS-WN is CNN with negative sampling and WordNet features (Xu et al., 2015a).

classification only showed moderate improvements. This is in agreement with previous research (Li et al., 2015), where explicit syntactic information was provided to classifiers by using tree structured networks and showed that syntax provides small improvements for binary sentiment classification in Stanford's Sentiment Treebank.

We test the statistical significance of the differences between embedding methods using a method described by Demšar (2006) for comparing classifiers between multiple datasets. We use the sign test that counts wins and losses of classification methods in different settings and does not assume any commensurability of scores. The difference in performance between LG and Win5 word embeddings si statistically significant (p=0.036) and similarly the difference between EXT and LG word embeddings is significant (p=0.036). When comparing the syntax based models using both words and dependency features (Wavg and Conc), the EXT models performs significantly better than the LG one (p=0.033).

It is notable that for QC and RI, using only word embeddings that are trained with syntactic information (LG and EXT *Words* models) still outperform the baseline window based skipgram. Using the dependency context embeddings as a means to represent the dependency parse of sentences consistently outperforms the baseline method across the three tasks and for every classification method. This indicates that this additional syntactic information cannot be recovered by the CNN and LSTM even though they have access to the sequential structure of sentences, at least when trained on datasets of this size. As expected, the SVM NBoW benefits the most by the addition of dependency context embeddings since these are its only source of structural information.

The dependency context embeddings from the EXT model outperform the LG model, both

when used alone and when in combination with the word embeddings. This can be attributed to the additional information they are exposed to during training.

The effectiveness of the *Wavg* compared to the *Conc* method for combining word and dependency context embeddings seems to depend on the classification method. In genearal, we observe that the CNN performs better with *Wavg*, while SVM and LSTM with *Conc*. On the other hand, the ensemble methods of the Win5 model (*AvgE* and *ConcE*) do not provide any consistent advantage over the baseline. In most cases, *AvgE* slightly hurts performance while *ConcE* slighty improves it.

Our evaluation also suggests that best performing models in word similarity tasks do not necessarily achieve the best performance in other NLP tasks. When considering only word embeddings as features for sentence classification (*Words* method), we observe that the EXT model on average performs better than the Win5 and LG models, while the opposite is true for word similarity evaluation. This indicates that providing additional contextual information for training embeddings results in less specialized embeddings for particular types of semantic similarity evaluations, but can be useful for a wide range of sentence level classification tasks.

While the purpose of the experiments is a comparison of embeddings and little hyperparameter tuning was done for the classifiers, results of the CNN using EXT Wavg representations for QC (95.0) and RC (84.31) are close to the best reported results with specifically engineered systems for these tasks: 96.0 for QC (Mou et al., 2015) and 85.6 for RC (Xu et al., 2015a). For the RC task without additional resources like WordNet, the proposed method provides the best reported result. As the method does not depend on a specific classification setting it would be interesting to see if those approaches can further improve using dependency based representations.

After the release of the Extended Dependency Embeddings resource, the word type embeddings of EXT and LG skip-grams have been evaluated by other researchers on additional tasks. In the experiments conducted by Eger et al. (2017), both EXT and LG variants were found to marginally outperform GloVe embeddings of Pennington et al. (2014) (window context word embeddings) for keyphrase extraction using a stacked ensemble for classification. In the work of Reimers and Gurevych (2017a,b), the authors conducted a very large search over hyperparameters for LSTM models on five sequence labelling tasks, testing seven different options of pretrained embeddings. After evaluating more than 50K configurations they concluded that the choice of pretrained embeddings is one of the most impactful hyperparameter choices, and that the EXT word embeddings outperform other options for most tasks by a large margin.

3.7 Conclusion

In this chapter, a new context definition for dependency trees was proposed and used to train a word and dependency feature model with the skip-gram objective. The model was compared against skip-gram models with a window based context and a less general dependency based

context definition in word similarity and sentence classification tasks of question classification, binary sentiment prediction and semantic relation classification. For the sentence classification, we use three classifiers (SVM, CNN, LSTM) and experiment with several methods of utilizing dependency feature embeddings to create representations that capture the syntactic role of words in dependency graphs. By evaluating on two word similarity datasets, we reaffirm that dependency based models produce word embeddings that better capture functional properties of words and that window based models better capture topical similarity. The dependency based word embeddings largely improved the performance of the three classifiers for question classification and semantic relation classification, but only marginally for sentiment prediction. Using dependency context features along with the word embeddings we observed better performance for all the three classifiers regardless of task. We conclude that dependency feature embeddings can be used as a means to provide syntactic information for short text classification, and that the method can be applied to different architectures and benefit from any additional structure that those architectures may utilize.

Using parsed data to estimate feature representation offers many opportunities for future work, for example, the presented method can be applied with a semantic parser instead of syntactic dependencies. It would also be interesting to merge graphs from different parsers and embed their substructures in the same latent feature space. Another direction of experimentation is looking at the contribution of dependency feature embeddings for tree structured Neural Networks. While such networks take dependency structure into account for composition, they do not account for dependency types and only distinguish head and dependent types of relations between words suggesting they could benefit from the dependency type information of the proposed embeddings.

CHAPTER 4

Word Sense Induction with a Probabilistic Generative Model of Continuous Feature Vectors

In this chapter we continue to experiment with latent continuous syntactic representations and examine their impact in a task where we do not use any labelled data: Word Sense Induction and Disambiguation (WSI). A structured generative latent variable model for WSI that integrates information from multiple contextual representations is presented and evaluated. The proposed model combines evidence form global lexical, local lexical and dependency syntactic context. Each context type is associated with a latent variable and the three types of variables share a linear structure. Word and dependency feature embeddings of the extended dependency skip-gram model are used to construct all three types of representations, reducing the total number of parameters to be estimated and enabling better generalisation compared to models utilizing discrete features. We describe an EM algorithm to efficiently estimate model parameters and use the Integrated Complete Likelihood criterion to automatically estimate the number of senses. The model achieves state-of-the-art results on two Word Sense Induction and Disambiguation datasets.

4.1 Introduction

Word Sense Induction (WSI) aims to automatically discover the different senses of polysemous words by unsupervised processing of text corpora. The related task of Word Sense Disambiguation (WSD) seeks to map the senses of word instances in a specific context to a predefined sense inventory. WSI overcomes the problem of having to define sense inventories, which may

not have the appropriate granularity for all applications, and the effort of updating them for new domains or novel senses (Klapaftis and Manandhar, 2013). WSI is a challenging task that remains largely unsolved, but can have applications in systems that require semantic processing of natural language like concept search (Navigli, 2009).

WSI is typically modelled as a clustering task, where the aim is to cluster samples of context representations of ambiguous words. Since context is the only available information to a WSI model, the choice of informative representations is a very important modelling aspect. Broad context related to topic or domain can restrict the possible senses that are applicable to an ambiguous word, but in order to make fine grained distinctions, context on the phrasal or syntactic level is usually needed. Ideally, a WSI system should incorporate different types of contexts to increase the confidence in its decisions. For example, in cases where local context is not sufficient to determine the word sense, a system can base the decision on broad context. Combining the information present in different context representations can be challenging in an unsupervised setting. Previous work has combined lexical with syntactic context (Brody and Lapata, 2009; Lau et al., 2012), and topical with local lexical context (Wang et al., 2015).

Another challenge for WSI systems is the need to apply clustering methods in high dimensional spaces of sparse features (Kriegel et al., 2009). Probabilistic latent variable models have been successful in WSI by inducing latent representations of features that help improve generalisation by statistical sharing of parameters. While the latent variable approach has been very successful for word features, it has not provided any considerable advantage when used with syntactic features (Brody and Lapata, 2009; Lau et al., 2012). A possible reason for this is that syntactic features such as dependency features used in Chapter 3, exhibit much more sparsity than words.

A promising method to overcome the sparsity problem inherent in high dimensional discrete feature spaces is making use of low dimensional feature embeddings learned in an unsupervised manner such as skip-gram word embedding. Skip-gram embeddings exhibit compositional properties under addition, making them useful for constructing representations of phrases and larger units of text. Dependency based skip-gram models like those presented in chapter 3 can have the potential to be used in order to incorporate syntactic information while not exhibiting severe sparsity issues. While word embeddings have been successfully used in many supervised NLP problems to overcome the problem of sparsity and improve generalisation (Turian et al., 2010; Collobert et al., 2011), their application in WSI has been very limited so far.

Given the above observations, we present a WSI model to address the issue of sparsity by utilizing both multiple context representations and low dimensional feature representations. The model is a structured generative model of continuous feature vectors that jointly models topical, phrasal and syntactic context through discrete latent variables. The probabilistic framework allows us to integrate different types of information in a principled way and also allows the application of model selection criteria to automatically determine the optimal number of senses,

a difficult problem that needs to be addressed by a WSI system. We address the issue of high dimensional feature spaces when dealing with syntactic features, by using dependency feature embeddings. In particular, we use the Extended Dependency Skip-gram embeddings to create representations for all three context types.

The methods and experiments in this chapter aim to provide additional evidence for answering research question 1, in a different scenario than the one presented in Chapter 3. We are again looking at the impact of syntactic structure and whether it can be represented by addition of dependency feature embeddings pre-trained for a different task. Contrary to Chapter 3, we are dealing with a completely unsupervised task showing the benefit of using representations that exhibit compositionality through an unparameterized operation, i.e. addition. We again make use of the two different ways to encode structure, one by structure encoding features and one by the model's architecture. In particular, we provide syntactic information with features and different definitions of context through the model's architecture. The evaluation of dependency features in WSI can also tell us if the semantic properties of the representations correlate with human judgements to a degree that they can distinguish word senses by their context.

The WSI model is evaluated in two competitive benchmarks: SemEval-2010 Task 14: Word Sense Induction and Disambiguation (Manandhar et al., 2010), and SemEval-2013 Task 11: Word Sense Induction for graded and non-graded senses (Jurgens and Klapaftis, 2013). The two tasks provide different WSI evaluation frameworks and metrics. The proposed model achieves the state-of-the-art results in both datasets.

4.2 Related Work

Word Sense Induction and Disambiguation is one of the most studied problems related to natural language understanding. Since WSI is formulated as a problem of clustering word instances based on their context, a plethora of clustering algorithms and context representations have been explored. Most evaluations are based on the SemEval series of workshops, originally called SensEval and started running in 1998 every three years. SemEval is currently a yearly workshop running controlled competitions for many semantic oriented tasks, with a WSD related task and applications still being of central focus. WSI was the focus of three tasks: SemEval-2007 task 02: Evaluating word sense induction and discrimination systems (Agirre and Soroa, 2007), SemEval-2010 Task 14: Word Sense Induction and Disambiguation (Manandhar et al., 2010) and SemEval-2013 Task 11: Word Sense Induction for graded and non-graded senses (Jurgens and Klapaftis, 2013). Besides the systems participating in the evaluation, the datasets provided have been used extensively to facilitate research on WSI.

Most approaches to WSI start by building context vectors of each occurrence of an ambiguous word. Clustering of the context vectors can reveal distinct usages of the word, which are assumed to correspond to different word senses. WSI methods mostly differ in the clustering algorithm

and features to represent context. The chosen features for constructing context vectors are typically words in some local context (same sentence, paragraph or fixed length window), syntactic features, or second-order word co-occurrences which is related to using word representations.

The work of Schütze (1998) was one of the first proposed WSI methods. Context was represented by second order word co-occurrences, which can be achieved by aggregating explicit word vector representations. Clustering was performed by fitting a Gaussian Mixture model with the hard assignment EM algorithm. Evaluation on an artificially created benchmark showed that the induced senses could identify random words substituted in text (pseudowords) and that they can assist an information retrieval system. Pantel and Lin (2002) used dependency context features to represent context and evaluated the performance of classic clustering algorithms like k-means and hierarchical agglomerative clustering. They also proposed a specifically designed algorithm for WSI, Clustering by Committee (CBC), that models the power law distribution of word sense frequency. Evaluation by comparing with WordNet senses, showed that CBC performs better in WSI than other clustering algorithms. The work of Purandare and Pedersen (2004), the features are word triplets that contain a collocation following the one sense per collocation assumption, and clustering is performed by iteratively merging instances with similarity above a predefined threshold. Evaluation was performed with the pseudowords method showing good performance but not comparing against other methods. The method by Van de Cruys and Apidianaki (2011) uses Non-negative Matrix Factorization to learn a latent variable representation of words and syntactic context. Since the latent variables take non-negative values they can be interpreted as degree of membership to clusters, and a multiplication of the word representation with the context feature representations updates the membership values given the available context. This method resulted in state-of-the-art performance on the SemEval-2010 dataset at the time it was proposed but was later outperformed by Bayesian latent variable models discussed later. Goyal and Hovy (2014) applied spectral clustering (Ng et al., 2002) to a context similarity matrix computed with word embeddings but evaluation on the SemEval-2010 did not show very promising results.

An alternative to clustering context feature vectors is using graph clustering methods. These methods create a graph where vertices represent words, and edges are weighted according to association strength between words as estimated from a text corpus. Graph clustering methods are then applied to the co-occurrence graph to identify densely connected subgraphs indicating word senses. Véronis (2004) proposed the Hyperlex method, which used the small-world properties of graphs to cluster word co-occurrence graphs for WSI, applying an algorithm to detect dense subgraphs by iteratively identifying and removing graph hubs. The algorithm was shown to be useful for entity retrieval from web pages compared against a baseline assigning the most frequent sense. The hyperparameters of the Hyperlex approach were optimized by Agirre et al. (2006) and an alternative version that determines hubs with the PageRank algorithm (Page et al., 1999) was also tested. The optimized version of Hyperlex was shown to be better than PageRank, and almost similar in performance to supervised methods when used in a Word Sense Disambig-

uation benchmark. In the work of Klapaftis and Manandhar (2008), the nodes of the graph were collocations of the ambiguous words and the weights were determined by statistical association of the collocating words, while the graph was clustered with Chinese Whispers (Biemann, 2006), a graph clustering algorithm that determines the number of clusters automatically. Using collocations to determine graph connectivity was shown advantageous compared to typical word association measures in the SemEval 2007 benchmark. In a subsequent work (Klapaftis and Manandhar, 2010), the same authors applied hierarchical graph clustering with a probabilistic algorithm to capture the hierarchical structure of word senses and obtained state-of-the-art performance on SemEval-2010, performing better than the graph of collocations approach.

Some of the best performing systems for WSI in SemEval evaluations are Bayesian latent variable models based on Latent Dirichlet Allocation (LDA) (Blei et al., 2003). LDA is a generative model of discrete grouped data such as words in a document. It assumes that words are generated from discrete latent variables by a categorical distribution and that documents are categorical mixtures of the hidden variables with a Dirichlet prior. Parameter estimation for LDA is done by Bayesian estimation using Gibbs sampling. The advantage of LDA and its extensions for clustering contexts is that they simultaneously learn word representations as the parameters of the categorical distributions that generate them, mitigating feature sparsity problems.

An extension of LDA for WSI that can handle multiple feature types was proposed by Brody and Lapata (2009). The feature types used were words in a 5 and 10 window, word n-grams, part of speech n-grams, and dependency relations. The version combining a 5 and 10 word window context achieved state-of-the-art results on the SemEval-2007 dataset at the time of publication. The Hierarchical Dirichlet Process LDA (HDP-LDA) (Teh et al., 2012) is an extension of LDA that can automatically estimate the number of components. It was shown superior to simple LDA and has been one of the best performing model in WSI evaluations of all three SemEval datasets (Lau et al., 2012). The model was evaluated with word features, word with positional information and dependency features. Words with position features were better than just words but adding dependency features did not improve further its performance. The hidden concept model of Chang et al. (2014) is similar to LDA but assumes an additional layer of latent variables, the hidden concepts, which senses are a categorical mixture of. This model is the best performing in the SemEval-2010 dataset. The topic-sense model of Wang et al. (2015) is a Bayesian latent variable model with two types of latent variables jointly modelling broad context (topic) and word senses. The authors also explored weighting of contexts by similarity computed with word embeddings and methods to artificially expand the context of ambiguous words for better estimation of their senses. The sense-topic model is the best performing model in SemEval-2013 dataset.

Another line of work seeks to perform WSI jointly with word embedding in order to estimate multiple word vectors for different senses. In the model of Neelakantan et al. (2015), word embeddings are estimated with a modified version of skip-gram where the number of senses per word is predetermined and the centroid of context vectors determines which sense specific

embedding should be updated with each sample. In the work of Li and Jurafsky (2015) and Bartunov et al. (2016), Bayesian non-parametric latent variable models are incorporated into skip-gram to automatically determine the number of senses per word. These models are mostly concerned about word embedding representations that change in context and were not evaluated as WSI systems. An evaluation of the usefulness of sense specific embeddings performed by Li and Jurafsky (2015) showed that they can increase performance of systems in some NLP tasks when used in place of typical word embeddings.

4.3 Model Description

We propose a generative model of continuous feature vectors that captures interactions between different types of contexts for a target ambiguous word to be applied for Word Sense Induction and Disambiguation. We separate context into three distinct types: global lexical, local lexical and syntactic context.

Global lexical context is indicative of the text's topic or domain, which can restrict coarse grained senses of a word. It can range between a few sentences around the target word or consider the whole document. In this work, we define global context as the bag of words G observed in the same sentence as the target word and one sentence before and after, since this is the typical context size provided by WSI evaluation datasets.

Local lexical context captures the semantics of phrases and is the most typically used context used by WSI systems. We define it as the bag of words L within a five word window before and after the target word.

Finally, we define the syntactic context of a target word as a bag S with features of the typed dependencies with its neighbour words in a dependency graph. Dependency features are the same as those used in Chapter 3. Dependency features capture the syntactic selectional preferences of words and they are more closely associated with a target word than bags of lexical contexts.

The intuition behind using multiple context definitions is that we do not want to bias the model very much towards a specific representation. We expect that syntactic relations are the most useful feature for discovering word senses but occasionally they can be uninformative or the parser may make an error. Using three context definitions we provide complementary views on the structure of text that can make the model more robust. The implementation of this idea is similar to using the graph based features in a sequence classifier as in Chapter 3: different structural biases can be provided by the features and the model architecture. We exploit the flexibility of the extended dependency representations to create the representations of all the three context types with the same feature embeddings.

4.3.1 Continuous Context Feature Vectors

While our probabilistic model allows using the output of different models for each context type representation, e.g. a topic model for global context and a neural network for local, we use the Extended Dependency Skip-gram embeddings to construct representations for all three types. We rely on the additive compositionality properties of skip-gram embeddings to create a single vector for each context type.

Given the discrete features of the three context types, we create three continuous feature vectors by aggregating their corresponding embeddings. The operation to construct continuous context vectors is a weighted addition. We use the self information of discrete features to weight the contribution of each embedding to the overall feature vector:

$$I(x) = -\log(P(x)) \tag{4.1}$$

where the probability of each feature is estimated from their frequency in the Wikipedia corpus used to train the embeddings.
The three types of context feature vectors are formally defined by:

$$x_g = \frac{\sum_{w \in G} I(w) e_w}{||\sum_{w \in G} I(w) e_w||_2} \tag{4.2}$$

$$x_l = \frac{\sum_{w \in L} I(w) e_w}{||\sum_{w \in L} I(w) e_w||_2} \tag{4.3}$$

$$x_s = \frac{\sum_{d \in S} I(d) e_d}{||\sum_{d \in S} I(d) e_d||_2} \tag{4.4}$$

G,L and S are bags of discrete features corresponding to global, local and syntactic context as defined above, w and d are discrete word and dependency features, e_w and e_d are embeddings of words and dependency contexts. We also apply L2-normalisation to all feature vectors. The weighting by self-information reflects the intuition that rare words are more important for distinguishing senses than common words, and provides an automatic way of filtering the contribution of tokens with little semantic content such as punctuation and determiners.

4.3.2 Probabilistic Generative Model of Context Feature Vectors

We infer the senses of an ambiguous word by combining information provided by the three context feature vectors within a structured generative model. The model assumes the existence of three discrete latent variables for each target word z_g, z_l and z_s, each one generating one of the context feature vectors. The three latent variables form a chain structure, where variables responsible for broader context generate a more context specific latent variable and their corresponding observed feature vector. The chain structure was chosen since it is the simplest structure

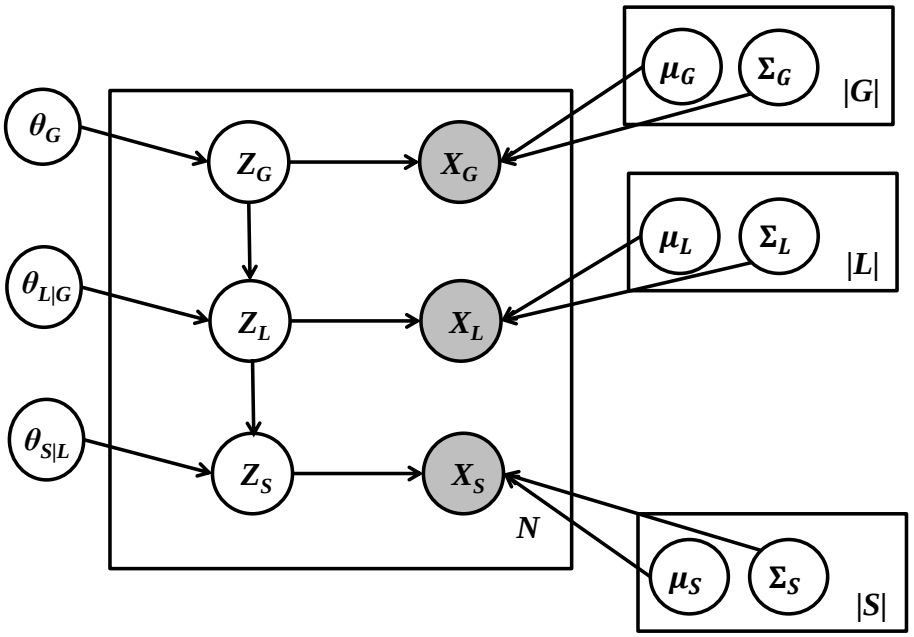


Figure 4.1: Graphical Representation of the model.

(i.e., requiring the fewest parameters) to model dependence of the three latent variables.

We model the probability density of each context vector as a Mixture of Gaussians:

$$p(x) = \sum_i p(z_i)p(x|\mu_i, \Sigma_i) \tag{4.5}$$

$$p(x|z_i) = \mathcal{N}(\mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K|\Sigma_i|)}} exp(-\frac{1}{2}(x-\mu_i)^\top|\Sigma_i|^{-1}(x-\mu_i)) \tag{4.6}$$

The full model is a structured generalisation of Gaussian Mixture Models. It should be noted that the model is generative with respect to the continuous vectors, and the mapping from discrete features to continuous is performed by a fixed deterministic function, in our case the embedding look-up and weighted summation with normalization.

We can use the model to sample context feature vectors following this generative procedure: For each target ambiguous word n:

sample $z_{ng} \backsim Categorical(\theta_g)$

sample $x_{ng} \backsim \mathcal{N}(\mu_{g=z_{ng}}, \Sigma_{g=z_{ng}})$

sample $z_{nl} \backsim Categorical(\theta_{l|g=z_{ng}})$

sample $x_{nl} \backsim \mathcal{N}(\mu_{l=z_{nl}}, \Sigma_{l=z_{nl}})$

sample $z_{ns} \backsim Categorical(\theta_{s|l=z_{nl}})$
sample $x_{ns} \backsim \mathcal{N}(\mu_{s=z_{ns}}, \Sigma_{s=z_{ns}})$

The corresponding graphical model can be seen in Figure 4.1. The parameters of the model are:

$$\Theta = \{\theta_g, \theta_{l|g}, \theta_{s|l}, \mu_g, \Sigma_g, \mu_l, \Sigma_l, \mu_s, \Sigma_s\} \tag{4.7}$$

We constrain the covariance matrices to be diagonal, hence having a smaller number of parameters compared to discrete mixture models for WSI.

4.3.3 Parameter Estimation

The parameters of the model can be estimated with several different methods. In general, we can group methods into two categories: maximum likelihood estimation (MLE) and Bayesian estimation.

In the MLE method we look for parameters that satisfy:

$$\Theta_{MLE} = \arg\max_{\Theta} \log p(x|\Theta) \tag{4.8}$$

This is an optimization problem that does not have a closed form solution for models with latent variables, but iterative methods such as gradient based optimization can be applied. The parameters need to satisfy certain constraints: that the categorical distribution parameters are non-negative and sum to 1, and that covariance matrices are positive semi-definite. An iterative method that naturally takes care of the constraints is expectation maximization (EM) (Dempster et al., 1977). EM performs two steps per iteration: the E-step and the M-step. During the E-step, we evaluate the probability of latent variables given the current parameter values and observations. Then, during the M-step, we estimate new model parameter values by treating the probabilities computed in the E step as observations, which for models as the one considered result in a closed form solution.

EM is guaranteed to increase the likelihood of the data at each iteration and shows fast convergence properties. In particular, it has proven linear convergence rate and in practice often exhibits quadratic convergence (Salakhutdinov et al., 2003). However, MLE with EM has some drawbacks. EM will only converge to a local optimum of the parameter space, and models with latent variables are known to have non-convex likelihood functions. In addition, the optimization can get affected by singular solutions of the covariance matrices. For small datasets, the mean of a Gaussian can become equal to one of the observations, which in turn will result in a zero covariance matrix and infinite likelihood.

Bayesian estimation of the model parameters takes a different approach. A prior probability distribution is placed on parameter values and then they are treated as latent variables. Estimation then amounts to computing the posterior distribution of the parameters given the data:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{4.9}$$

In the presence of latent variables, computing the posterior distribution is intractable and approximate inference methods need to be used. A commonly used method is Gibbs sampling (Geman and Geman, 1984). Gibbs sampling is an instance of a Markov Chain Monte Carlo method, where at each step we draw a sample of one latent variable conditioned on all other variables. This sampling procedure will eventually start giving samples from the true posterior distribution. We can then get a point estimate for a parameter value by selecting the mode or computing its expected value by averaging across samples.

Gibbs sampling starts with random assignments of the variables and will give samples from the true posterior after a convergence period known as burn-in. In addition, to obtain uncorrelated samples needed to approximate the quantities of interest we need to skip some samples every time we collect a new one, which is called lag time. There are no guarantees for knowing how many samples are required for the burn-in and lag time, so heuristics are used in practice. In addition, for some problematic cases Gibbs sampling can get stuck for too many iterations in a mode, and not be able to provide representative samples for the whole posterior distribution in reasonable time.

Both MLE with EM and Bayesian estimation with Gibbs sampling can effectively estimate the parameters of GMMs but may encounter problems. Dias and Wedel (2004) found that EM and Gibbs sampling give similar quality of solutions. We choose to apply MLE with EM but take measures to mitigate the problems mentioned above. We note that the E-step of EM can become computationally intractable when there are many dependent latent variables. In the case of the proposed model, there are three dependent latent variables per sample and computing the probabilities is computationally feasible. For the local optimum problem we make several runs of similar experiments with different initializations. We choose the best model with the Integrated Complete Likelihood criterion (ICL), a model selection criterion that is explained in detail in the next section. ICL is used both to avoid runs that get stuck in a bad local optimum and to choose the number of components which is a hyperparameter of the model. In addition, we initialize the cluster assignments by first running the k-means++ algorithm (Arthur and Vassilvitskii, 2007), which results in better initializations (Blömer and Bujna, 2013). To mitigate the singularity problem, we add a small constant value to the covariance matrices that prevents them from becoming zero.

The update equations for the E-step and M-step for the proposed model are the following:

E-step

For each sample $n \in \{1, .., N\}$ we compute:

$$\gamma(Z) = p(Z|X) = \frac{p(Z, X)}{\sum_Z p(Z, X)} \tag{4.10}$$

Since there are only three dependant latent variables per sample, we can easily compute this step by exact inference.

M-step

We estimate parameters that maximize:

$$\Theta^{new} = \arg\max_{\Theta} \sum_{Z} p(Z|X;\Theta^{old}) \log p(X,Z;\Theta) \tag{4.11}$$

Given the factorization implied by the graphical model, each parameter can be estimated independently. The update equations are:

$$\theta_g^{new} = \frac{1}{N} \sum_{n,l,s} \gamma(z_{ngls}) \tag{4.12}$$

$$\theta_{l|g}^{new} = \frac{\sum_{n,s} \gamma(z_{ngls})}{\sum_{n,l,s} \gamma(z_{ngls})} \tag{4.13}$$

$$\theta_{s|l}^{new} = \frac{\sum_{n,g} \gamma(z_{ngls})}{\sum_{n,g,s} \gamma(z_{ngls})} \tag{4.14}$$

Updates for means and covariances for the g context type are given by:

$$\mu_g^{new} = \frac{\sum_{n,l,s} \gamma(z_{ngls}) x_{ng}}{\sum_{n,l,s} \gamma(z_{ngls})} \tag{4.15}$$

$$\Sigma_g^{new} = \frac{\sum_{n,l,s} \gamma(z_{ngls})(x_{ng} - \mu_g)(x_{ng} - \mu_g)^T}{\sum_{n,l,s} \gamma(z_{ngls})} \tag{4.16}$$

and similarly for the l and s types.

We initialize the mean vectors with the centroids of k-means++ run independently for each context type, and covariance matrices with the sample covariance. We add a regularization value equal to 10^{-3} to covariances to avoid singularities.

4.3.4 Model Selection

One of the most challenging parts of WSI is estimating the number of senses for each ambiguous word type. Since we are working with a probabilistic model we can apply model selection criteria. We use the Integrated Complete Likelihood (ICL) criterion (Biernacki et al., 2000). ICL is a model selection criterion similar to the Bayesian Information Criterion (BIC) (Schwarz, 1978) that seeks a model that provides large evidence for the observed data with a small number of parameters. Following (McLachlan and Peel, 2004) we use the approximation:

$$ICL(m,X) = \log p(X|\Theta) - \frac{m_k}{2} \log(N) + \sum_{n,g,l,s} \gamma(z_{ngls}) \log \gamma(z_{ngls}) \tag{4.17}$$

where m_k is the total number of parameters to be estimated.

This ICL approximation is exactly equal to BIC additionally penalised by the last term, which is the mean entropy of the distribution of latent variables. The extra penalty term favours models that result in more confident assignments, since the entropy of the latent variable distribution will become lower in such cases. ICL prefers models with well-separated clusters and penalizes strongly overlapping components, an aspect not considered in model selection by BIC.

In our experiments, we set $|z_g| = |z_l| = |z_s| = K$ and train models with K in the range of [2, 50]. We observe that given enough training instances, ICL picks models with a large number of components, corresponding to more fine-grained senses. This is reasonable, since with more data, the model becomes more confident into making such fine-grained distinctions, and is also likely to encounter unusual word usages.

4.4 Evaluation

We evaluate the proposed model in two SemEval WSI datasets. For both datasets we parse the data using the Stanford Neural Network dependency parser (Chen and Manning, 2014) using Universal Dependencies (De Marneffe et al., 2014), which is the same format used by the dependency based embeddings. We train a different model for every word type.

4.4.1 SemEval-2010 Task 14: Word Sense Induction and Disambiguation

The SemEval-2010 WSI dataset consists of 50 verbs and 50 nouns. The task organisers provided a fixed training set with 879,807 instances of the target words. The distribution of instances for each word is highly imbalanced. The test set consists of 8,915 instances. Two types of evaluation are performed: supervised and unsupervised.

The supervised evaluation is performed in two steps. In the first step, a part of the data is used to map the induced sense clusters to a fixed inventory of word senses. The mapping is performed by counting co-occurrences (or summing probabilities when provided) of sense clusters with gold sense assignments, and then computing the most likely assignment by normalizing the counts. In the second step, the fixed sense inventory is used to evaluate the clustering of the rest of the data as a Word Sense Disambiguation system. The reported metric is balanced F-score:

$$F_1 = \frac{2pr}{p+r} \tag{4.18}$$

where p is precision and r is recall of the system's provided answer.

Following the SemEval task procedure we report two results, one using an 80-20 split of the data for mapping and scoring, and one using a 60-40 split. For both cases, reported results are an average over a 5-fold split.

Unsupervised evaluation for the SemEval task was performed by two clustering quality metrics, V-measure and paired F-score. A problem with clustering evaluation metrics is that they are sensitive to the number of senses (Klapaftis and Manandhar, 2013), with V-measure favouring a high number of senses and F-score the opposite. This behaviour results into ranking two uninformative baselines, 1-cluster-per-instance and most-frequent-sense (or all-in-one), as the best solutions. In (Li et al., 2014), the authors argue that in the case of the V-measure, this behaviour can be explained by biases in the estimation of entropy when there is a large number of clusters compared to the number of samples, as is the case in WSI evaluation. They propose the usage of the Best-Upper-Bound (BUB) Entropy Estimator (Paninski, 2003) instead of maximum likelihood estimation that was used by the organisers. They show that the V-measure with the BUB estimator successfully evaluates both uninformative baselines as worse solutions than actual WSI systems. Following this recommendation, we report the V-measure estimated with BUB as the unsupervised evaluation metric.

The symmetric V-measure is an instance of normalized mutual information between two distributions, in this case the sense cluster distribution c and the sense distribution k provided by the annotators. There are many possible ways to normalize mutual information, and V-measure is obtained when normalizing by the sum of the entropies of the two distributions:

$$V(k,c) = \frac{2I(k,c)}{H(k)+H(c)} = \frac{2(H(k)+H(c)-H(k,c))}{H(k)+H(c)} \tag{4.19}$$

where $H(\bullet)$ is the entropy of the distribution. The typically used MLE of the entropy for m clusters, N samples and n_i being the count of sample assignments for cluster i is given by:

$$H_{MLE}(c) = \sum_{i=1}^{m} -\frac{n_i}{N} \log \frac{n_i}{N} \tag{4.20}$$

A general formulation for entropy estimators can be expressed as a linear function of ordered histogram statistics h_j, i.e. the number of clusters that appear j times:

$$\widehat{H(a)} = \sum_{j=0}^{N} a_{j,N} h_j \tag{4.21}$$

$$h_j = \sum_{i=1}^{m} I[n_i = j] \tag{4.22}$$

where $I[\bullet]$ is the Iverson bracket. The coefficients $a_{j,N}$ depend on the different estimators used. The maximum likelihood estimator of entropy is given by coefficients:

$$a_{j,N}^{ML} = -\frac{j}{N} \log \frac{j}{N} \tag{4.23}$$

The BUB estimator can be obtained by computing the upper bound on the bias of entropy

Model	80-20	60-40	VM (BUB)
MCC-G,L,S	**70.6**	**69.0**	8.1
MCC-L,S	69.3	68.1	12.4
MCC-S	68.9	67.5	**17.1**
Hidden Concept (Chang et al., 2014)	69.7	68.9	-
HDP-LDA (Lau et al., 2012)	68.0	-	-
UoY (Korkontzelos and Manandhar, 2010)	62.4	62.0	11.4

Table 4.1: Results on the Semeval-2010 WSI dataset. Dashes indicate that this result was not reported by the authors of the corresponding model. UoY is the best performing model participating in the SemEval-2010 WSI evaluation.

estimation and then selecting coefficients that minimize the mean squared error between the estimated entropy and that upper bound. This corresponds to solving a regularized least squares optimization problem.

Our default approach for assigning a sense to a word instance is taking the value of $p(z_s|x)$ as the probability of a sense being applicable, since it is the variable most directly associated with the ambiguous word. It is possible however, to assign senses to joint configurations of two or all three of the latent variables, in order to better use the information provided by our model. Since the number of possible configurations grows exponentially with the number of latent variables, this approach can lead to very fine-grained partitions of the data. In practice, we observe that only a small number of all these configurations are given a high probability mass. Evaluation results for the proposed model MultiContextContinuous (MCC) and state-of-the-art systems are reported in Table 4.1. We report results with both the single variable approach (MCC-S) and joint variable assignments (MCC-L,S and MCC-G,L,S).

We see that our model achieves the highest score in both metrics, however, by utilizing different information. The combined evidence provided by the three latent variables helps induce an accurate mapping of the sense clusters to the fixed sense inventory and results in the best F-score for the supervised evaluation. The supervised evaluation benefits from this fine-grained sense distribution since splitting the senses into smaller clusters does not affect the mapping operation, as long as the induced senses are consistent subsets of the gold standard senses. The 60-40 split results into a more difficult mapping problem and can indicate how reliable is the mapping between the sense distribution and the gold standard (Klapaftis and Manandhar, 2013). We see that using the joint configuration of the latent variables again results in the highest F-score, providing evidence of the consistency of the mapping. Contrary to the supervised evaluation, the V-measure favours the clustering provided by the z_s variable alone, since it penalises the mismatch between the more fine-grained clustering and the gold standard.

4.4.2 SemEval-2013 Task 13: WSI for graded and non-graded senses.

The SemEval-2013 WSI evaluation dataset consists of 50 word types: 20 verbs, 20 nouns and 10 adjectives. There are several differences compared to the SemEval-2010 setting. There are fewer restriction on the training set, which can be any part or all of the UkWac corpus (Ferraresi et al., 2008). In addition, the test data include instances with multiple applicable senses. There are 4664 instances in the test set, 88.5% of which are labelled with a single sense, 11% labelled with two senses and 0.5% with three. Systems are asked to provide an estimate of the applicability of each sense.

The evaluation metrics also differ from those of SemEval-2010, in order to cope with the fuzzy sense labelling. Clustering evaluation is performed with the Fuzzy B-cubed and Fuzzy Normalised Mutual Information (NMI) criteria. Fuzzy B-cubed is computed as the harmonic mean of precision $B\_P(X,Y)$ and recall $B\_R(X,Y)$ between membership weights of cluster assignments X and annotations Y:

$$B\_P(X,Y) = \operatorname*{avg}_i[\operatorname*{avg}_{j \neq i \in \cup \mu_Y(i)} P(i,j)] \tag{4.24}$$

$$B\_R(X,Y) = \operatorname*{avg}_i[\operatorname*{avg}_{j \neq i \in \cup \mu_X(i)} R(i,j)] \tag{4.25}$$

$$P(i,j,X) = \frac{\min(C(i,j,X), C(i,j,Y))}{C(i,j,X)} \tag{4.26}$$

$$R(i,j,X) = \frac{\min(C(i,j,X), C(i,j,Y))}{C(i,j,y)} \tag{4.27}$$

$$C(i,j,X) = \sum_{k \in \ell_X(i) \cup \mu_X(j)} 1 - |w_k(i) - w_k(j)| \tag{4.28}$$

where avg is the arithmetic mean of a collection of numbers, $\mu_x(i)$ and $\mu_y(i)$ the set of clusters in clustering X or Y of which item i is a member, and $w_k(i)$ the membership weight of item i in cluster k in X.

Fuzzy NMI is a generalization of NMI for the case of fuzzy cluster membership. The NMI used here is similar to the symmetric V-measure but uses a different normalization approach:

$$NMI(k,c) = \frac{2(H(k) + H(c) - H(k,c))}{\max(H(k), H(c))} \tag{4.29}$$

Fuzzy set membership is discretized in 10 uniformly distributed bins in $[0,1]$ at 0.1 intervals. Then we compute the entropy of a categorical distribution over a set of weights that denote the degree of membership in that sense cluster:

$$H(X_i) = \sum_{i=1}^{10} p(w_i) \log p(w_i) \tag{4.30}$$

Fuzzy NMI favours solutions with many clusters giving a high score to the 1-cluster-per-instance baseline, while Fuzzy B-cubed favours few clusters and favours the all-in-one baseline. We sampled 20K instances from UkWac for each target word to train our model. Following (Wang et al., 2015), we use those two metrics but also report the geometric mean of the two, which provides a more balanced metric and also assigns a score of zero to both the uninformative baselines. We use the top 3 most probable assignments of the z_s variable with the corresponding probability as the applicability weight. Results can be seen in Table 4.2.

We distinguish results for the standard test data provided by the SemEval task organisers and the augmented test data used in the evaluation of the sense-topic model. The first data augmentation result, indicated as "add-actual-context", uses an extra two sentences before and after the provided test data sentences, extracted by finding the test instances in their original corpus. The second data augmentation result "add-UkWac-context", uses context extracted from UkWac by finding word instances in a similar context as the test instances. Similarities are calculated by averaging word embeddings in the test instance and calculating cosine similarities. These context augmentation techniques improve the performance of the sense-topic model since the test data provided usually consist of a single sentence as the context of the target word.

Since the proposed model also considers global context, it is possible that such data augmentation techniques would also increase its performance, but we did not apply this approach in our evaluation setting and only used the provided context found in the SemEval test data. Evaluating context augmentation techniques is out of scope of this thesis, but it would be interesting to see how the two models change performance as more context becomes available. However, direct comparison between the two models with context augmentation is not feasible because the authors did not release their context augmented data set and recreating their exact setting requires access to their embedding model used for similarity computation and knowledge of the training instances sampled from UkWac.

When using the actual SemEval test data, the MCC model achieves the highest score in all three metrics.

4.5 Discussion

We attribute the good performance of the proposed model to the rich structural information it has access to, while being able to cope with data sparsity at the same time. Both the multiple context representations and the usage of low dimensional feature embeddings contribute towards that goal. The importance of dealing with sparse inputs is also supported by the generally good performance of Bayesian latent variable models for WSI (Wang et al., 2015; Chang et al., 2014). While these models manage to deal with the sparsity of words, they still do not manage to effectively utilize syntactic features as effectively as we do with dependency feature embeddings. By using pretrained embeddings our model has access to a very large feature set (220K words

Model	Fuzzy-NMI	Fuzzy B-cubed	Geom. Mean
MCC-S	**7.62**	**55.6**	**20.58**
Sense-Topic (Wang et al., 2015)	6.96	53.5	19.30
Sense-Topic (sim. weighted)	7.14	55.4	19.89
AI-KU (Baskaya et al., 2013)	6.5	39.0	15.92
unimelb (Lau et al., 2013)	6.0	48.3	17.02
Sense-Topic (add-actual-context)	9.39	**59.1**	**23.56**
Sense-Topic (add-UkWac-context)	**9.74**	54.5	23.04
1cl-per-inst	7.09	0	0
all-in-one	0	62.3	0

Table 4.2: Results on the SemEval-2013: Task 13 dataset. All Sense-Topic model variants are reported from (Wang et al., 2015). Results with extra context (add-actual-context, add-UkWac-context) do not use the same evaluation setting and are not directly comparable to the rest. AI-KU and unimelb are systems participating in the SemEval-2013 evaluation.

Cluster 6
... energy-efficient appliances and how to *operate* them efficiently ...
... temperature rises enough for the heat pump to *operate* more efficiently than your old ...
... software to enable users to *operate* their computers remotely ...
Cluster 5
... 44 of these stores *operate* as monro muffler brake & service ...
... many industries with volatile profits ranging from oil exploration to computer software *operate* without substantial government regulation ...
... bfx hospitality group , inc. owns and *operates* food services ...

Table 4.3: Instances of "operate" belonging into two different clusters. The contexts do not share many common words, but they are semantically related. The second instance of cluster 5 shares words with the third instance of cluster 6 ("software", "computer"), but is assigned the correct sense because the syntactic features indicate the long range subject dependency with "industries".

and 1.3M dependency context features), while having to estimate a relatively small number of parameters. In table 4.3, we show some examples where clusters are formed by different but semantically related words, and the importance of syntactic features.

A limitation of the proposed model is that context representations corresponding to senses are assumed to follow Gaussian distributions. We cannot expect this assumption to hold in general, but our evaluation suggests that it is a reasonable approximation. It is possible to extend the model by using a separate Mixture of Gaussians to model each individual sense. While this extension would provide additional capacity to the model for modelling sense specific contextual representations with complex probability densities, it can lead to parametric explosion and severe overfitting.

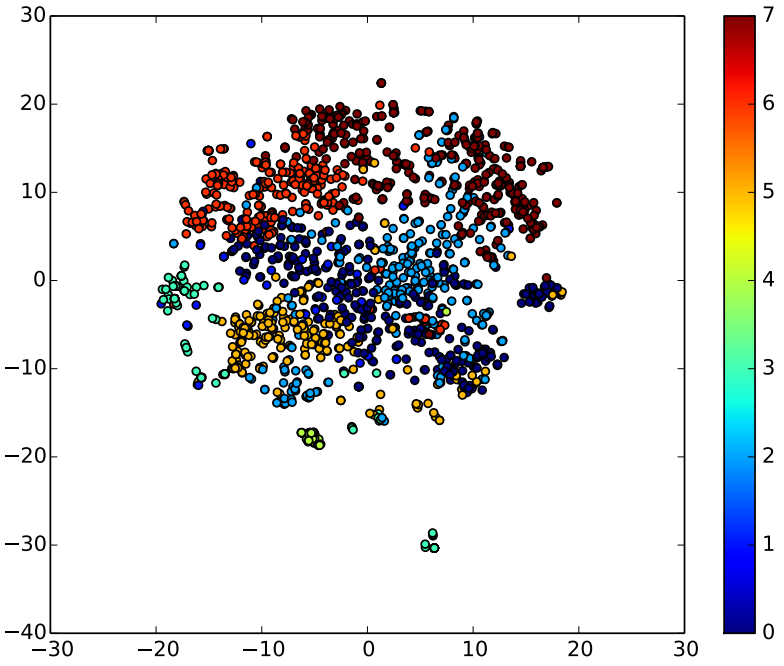


Figure 4.2: 2-d t-SNE of the syntactic context vectors of the word operate. Different colours correspond to different cluster assignments.

In Figure 4.2, we use t-SNE (Maaten and Hinton, 2008) to visualise the syntactic context vectors of the word "operate" from the SemEval-2010 training data and their sense assignments. We see that clustered points generally form compact groups, though they are not clearly separated. In order to model senses with contexts that do not follow a Gaussian distribution, the model favours additional clusters. In practice, we observe that this behaviour does not pose a significant problem as long as these finer-grained clusters are consistent with the underlying sense distribution, which was shown by the supervised evaluation of SemEval-2010. If a coarse grained distribution is desired, methods that merge Gaussian components as a postprocessing step could be considered (Hennig, 2010).

4.6 Conclusion

In this chapter, we presented a probabilistic latent variable model for Word Sense Induction and Disambiguation using continuous feature embeddings of different types. The model integrates

information from three different context types: global lexical, local lexical and dependency syntactic context. A different latent variable is inferred for each context type and dependencies are modelled in a structured top-to-bottom way, where broader context representations directly influence only more specific representations. Our context representations are constructed by weighted addition of word and dependency context embeddings that provide a way to overcome sparsity and reduce the number of parameters needed to be estimated. The number of senses is automatically determined by applying the Integrated Complete Likelihood Criterion. We evaluate our model in two competitive WSI benchmarks, achieving state-of-the-art results. We conclude that dependency feature embeddings are an effective method to provide syntactic information for unsupervised clustering of context representations, and that they can be used according to multiple different context definitions. There are several directions to be considered for future work. The first is exploring the impact of choosing a Gaussian distribution for senses compared to alternatives. A distribution to be considered is the von Mises Fisher distribution used to model data on a unit hypersphere. This is a more natural distribution to use for L2-normalized vectors than the Gaussian used in this work. Second, we can use Dirichlet process Gaussian mixture models to automatically infer the number of components and compare the results with the approach taken here, which is using the ICL criterion for model selection. Finally, we can consider hierarchical Bayesian models, similar to LDA but generating continuous features, to incorporate an additional layer of latent variables.

CHAPTER 5

Feature Embeddings for Knowledge Base Completion with Text

In this chapter, we look at the first task that requires joint modelling of text with a Knowledge Base: Knowledge Base Completion (KBC) with linked text resources. Our goal is to predict missing facts from a Knowledge Base (KB) that consists of (entity, relation, entity) triples by reasoning over its internal structure, and by using a side textual corpus where missing facts might be explicitly stated. In order to combine the evidence from the two resources, we build upon methods that jointly embed KBs and text into a latent feature space and then score facts with a function of the latent representations. Previously proposed methods for joint KB and text modelling directly extend latent feature models for KBC by assuming a joint graph with a Universal Schema consisting of the union of the KB and text relations. In this work, we maintain a distinction between the two resources and for every triple in the KB we create a new graph with additional edges expressing side information and providing alignment information between them. We then use a Multilayer Perceptron to embed and score features of this graph. We evaluate the proposed model in joint KBC with text using the FB15k-237 dataset and compare with competitive latent feature models. Our evaluation suggests that our method can better utilize the additional text resources than models that take a less structured approach.

5.1 Introduction

Knowledge Bases (KB) are an important resource for many applications such as question answering (Berant et al., 2013; Reddy et al., 2014), relation extraction (Mintz et al., 2009) and

named entity recognition (Ling and Weld, 2012). While large collaborative KBs like Freebase (Bollacker et al., 2008) and DBpedia (Auer et al., 2007) contain facts about million of entities, they are mostly incomplete and contain errors. A large amount of research has been dedicated to automatically extend knowledge bases, a task called Knowledge Base Completion (KBC). Proposed approaches to KBC either reason about the internal structure of the KB, or utilize external data sources that indicate relations between the entities in the KB.

Knowledge Bases can be represented as a directed edge labelled graph, where nodes are entities $sbj, obj \in E$ and edges are typed relations $r \in R$. A fact in the KB is encoded as a triple (sbj, r, obj), where sbj is the subject entity and obj is the object entity. Starting with an existing KB consisting of a set of observed facts, the goal of KBC is to reason about the plausibility of unobserved facts, potentially given some additional external resource.

Throughout this thesis, we use subsets of Freebase as the background KB. Commonly used subsets of Freebase for KBC are the FB15k (Bordes et al., 2013) and FB15k-237 (Toutanova et al., 2015) datasets. Freebase is a large Knowledge Base containing facts for about 40m entities, with more than 35K relation types and 600M facts. It was actively being updated by users in the 2007-2015 period, with a large portion of it being curated for consistency, and is now deprecated. The final version is available as an RDF triple store. Entities in Freebase are called topics and have a unique identifier such as `/m/02vyw`. Relations are treated as properties of the subject entity and encoded in the form of `domain/type/property`, for example `film/director/film`. Freebase also contains dummy nodes called mediator nodes, which can be used to encode compound value types that represent facts which cannot be encoded as a triple, e.g. the population of a city in a specific year. Compound types are commonly split into triples and encoded by a composite edge in datasets used as evaluation benchmarks for KBC. Freebase has a lot of redundancy, every relation has an inverse and facts are expressed twice with swapped subject and object positions.

In the most basic form, a KBC system can be built by extracting graph features, and training a binary classifier to predict if a missing edge should be present or not. Existing and missing facts from the KB can be used as positive and negative samples to estimate the classifier parameters. While this closed-world assumption contradicts the purpose of KBC, as we will see in the experiments, models trained this way can still successfully rank many unobserved true facts higher than unobserved false ones.

The approach considered in this work is an instance of latent feature models for KBC. (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013b; Nickel et al., 2016). Such models embed the symbols of the KB into a low dimensional space and assign a score to unseen triples as a function of the latent feature representations. Most approaches define a scoring function as a linear or bilinear operator. Latent feature models have shown good performance when considering the internal structure of KBs and are scalable to very large datasets.

Utilizing textual data or other external resources for KBC is a challenging task but has the

potential to constantly update KBs as new information becomes available. State-of-the-art approaches for KBC with external textual data are obtained by latent feature models that jointly embed the KB symbols and text relations into the same space (Riedel et al., 2013; Toutanova et al., 2015). The benefit of such models is that they can combine both the internal structure of the KB and textual information to reason about the plausibility of unobserved facts.

An established approach to combine a KB and a given a linked text corpus for KBC is by adopting a Universal Schema (Riedel et al., 2013), where extracted textual relations between entities are directly added to the knowledge graph and treated the same as KB relations. The schema is then the union of schemas of the KB and textual relations. This allows any latent feature model defined over triples to jointly embed the KB and text relations to the same space. An application of the Universal Schema approach resulting into state-of-the-art performance was proposed by Toutanova et al. (2015), where representations of text relations are formed compositionally by Convolutional Neural Networks (CNNs) and then composed with entity vectors by bilinear models to score a fact. However, these models show only moderate improvement when incorporating textual relations and have to balance the contribution of text and KB facts by an additional weighting parameter.

A limitation of the Universal Schema approach for joint embedding of KBs and text is that information about the correspondence between KB and text relations is only implicitly available through their co-occurrence with entities. The reason is that the knowledge graph is decomposed into the most simple substructure, an edge between two entities. Since entities are only represented by latent feature representations, the information about which KB and text relations are co-occurring with an entity pair cannot be recovered exactly once the graph structure has been decomposed into triples. Such a model is not aware that there are two types of sources that have been aligned through entity linking, and that the purpose of textual relations is using them as additional evidence to predict the existence of KB relations. This leads to loss of information that is readily available and we hypothesize that it makes learning more difficult. Treating textual relations the same as KB relations also requires a lot of modelling capacity to learn semantics that may not be relevant for the KBC task. Text relations can often be noisy and pairs of entities can co-occur in the same sentence without sharing a semantic relation. In addition, there is usually a mismatch in the relations found in the KB and those expressed in text.

We propose a different approach to combine KB and textual evidence, where the textual relations linked to an entity pair form a subgraph together with the KB triple. Instead of a Universal Schema, KB and text relations are merged in the graph but have their types distinguished. We treat additional resources like text differently since they are auxiliary to the task and the required prediction concerns the existence of a KB relation. In addition to text, we can expand the graph with any kind of information associated with the entities by adding edges to the subgraph and assigning a new type to them. We reason about the plausibility of new facts by constructing such subgraphs and then embedding them into a latent feature representation to be scored. Embedding

and scoring is performed jointly by a neural network that composes edges of each type separately and then concatenates their representations to become the input to an MLP. The MLP composes the representations of the different types and makes a prediction about the plausibility of a fact.

After composing the representations of edges of different types, we can think of a fact as an n-tuple where extra elements are added for each type of additional resource we use alongside the KB triple. We choose MLPs as they are a generic method to model interactions between latent feature vectors without having to specify the form of a composition operator for tuples of different arity. When scoring the plausibility of unseen facts, all the side evidence associated with that fact becomes explicit through the n-tuple. We define the composition of different types of edges as a simple aggregation of their feature embeddings and name the whole system Feature-Rich Network.

The proposed method of performing KBC with additional textual resources aims to provide evidence for answering the second research question. We expect that providing information about the alignment of the entity-relation based representation of the KB and the entity-relation mentions in text will be useful for embedding the two modes of input into a common space that captures their semantic properties. We note that we do not make any additional assumptions about the KB and text alignment than those already made in order to link the two resources via entity linking. The only difference is that we encode this structural information into the input of the model rather than discarding it.

We evaluate the ability of the proposed Feature-Rich Networks for KBC on the challenging FB15k-237 (Toutanova et al., 2015). We compare the performance of bilinear models to an MLP when facts are represented as simple triples, and the contribution of two additional types of side information: Freebase entity types and textual relation mentions from a side corpus. We also evaluate the contribution of initializing feature representations from external models. Evaluation suggests that while MLPs and bilinear models perform similarly when treating facts as triples of KB symbols, the proposed approach can better utilize additional textual data than a combination of CNNs with bilinear models, showing large improvements in predicting unseen facts when they have linked relation mentions in text.

5.2 Related Work

Knowledge Base Completion methods can be broadly categorized in two distinct settings depending on the available resources. The first kind is by only considering the structure of the KB itself and inferring missing facts by some form of deduction. The second is utilizing additional resources like multiple KBs, semi-structured data such as tables or text. Merging multiple KBs into a unifying schema is out of the scope of this research and we will only focus on the cases of using a KB alone and a KB along with a linked text corpus.

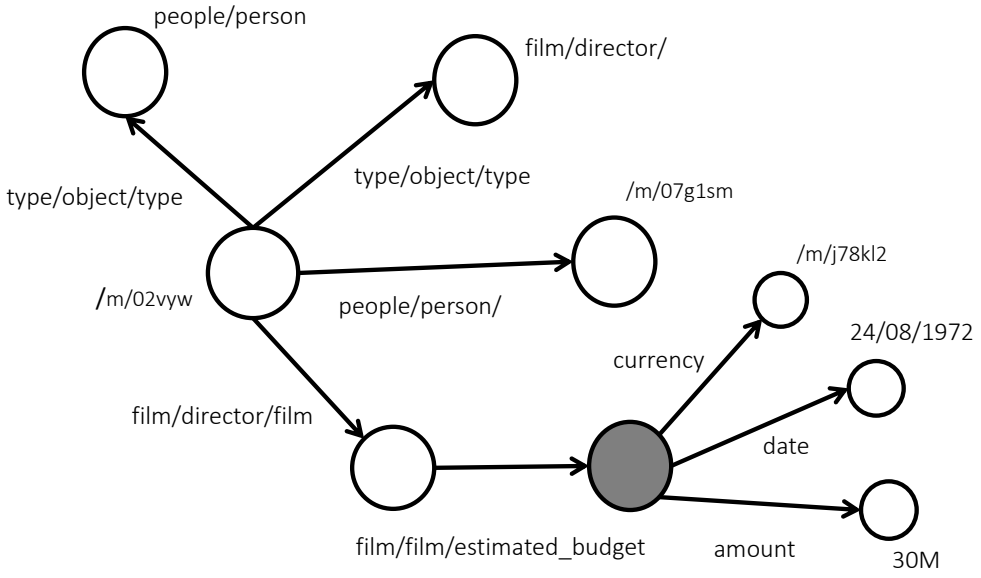


Figure 5.1: A subset of Freebase.

5.2.1 KBC without additional resources

KBC without additional resources can be formulated as edge detection on a labelled graph. This type of problem is generally called link prediction and is a well studied problem in the field of statistical relational learning (Lü and Zhou, 2011). Applying SRL methods to KBC requires them to be scalable to graphs of millions of nodes. Due to scalability issues, most studied KBC approaches are based on latent or observed feature models. Both latent and observed feature models make the simplifying assumption that the existence of a fact is conditionally independent of other facts given the features. The features should then encode information about the global structure of the graph. SRL approaches based on probabilistic logic such as Markov Logic Networks (Richardson and Domingos, 2006) offer the flexibility to encode more complex conditional independence assumptions, but due to higher computational complexity they have not been sufficiently explored for KBC.

Observed feature models have mostly used paths between entities as features in a classifier. The Path Ranking Algorithm (PRA) performs random walks in the KB to estimate the probability of reaching an entity node from another and the probabilities of different path types were used as features for KBC (Lao et al., 2011). The work of Gardner and Mitchell (2015) simplifies the extraction of paths by skipping their probability estimation through random walks, but

extracts more features from them and uses additional graph features from the entity neighbourhood, achieving better performance than PRA. The two models were compared in their ability to extract new facts from the NELL KB (Mitchell et al., 2015). Simple features of paths of length 1 have also been used in the work of (Toutanova and Chen, 2015) and shown to be able to extract new facts only when there are redundant inverse relations in the KB.

Latent feature models embed entities and relations into a low dimensional space and score the plausibility of a fact as a function of those representations. Information about the global properties of the graph is captured in the latent features. Many variations of latent feature models have been proposed differing in the definition of the scoring function, the loss function and the optimization procedure.

A class of latent variable models for KBC define the scoring function through multiplicative interactions between the representations of entities and relations. Such models are commonly referred to as bilinear models and can be described as a factorization of the third order tensor X_{ijk} formed by the KB triples. Rescal (Nickel et al., 2011) is a tensor factorization method specifically designed for KBC, where relations are represented as matrices and head and tail entities as shared vectors. Approximate reconstruction of the original tensor by multiplication of the latent factors can reveal new facts. A simplification of Rescal was proposed by Yang et al. (2014), where the relation matrices are restricted to be diagonal, making the scoring function a simple element-wise multiplication between three vectors followed by a summation of the resulting vector's values. This simple model provides a strong baseline for KBC, with a carefully tuned version shown to outperform more complicated models (Kadlec et al., 2017). The Semantic Matching Energy (Bordes et al., 2014) composes relations with subject and object entity separately through element-wise multiplication and scores the triple as the dot product of the two resulting vectors. Models F and E of Riedel et al. (2013) formulate the factorization problem only through two-way interactions reducing it to matrix factorization. Model F assigns vector representations to pairs of head and tail entities and scores facts by the dot product of entity pairs and relation vectors, while model E computes dot products between the two entities and the relation vector and sums them.

A different class of latent variable models uses distances to score fact triples. The structured embedding method of Bordes et al. (2011) projects entities into a relation specific subspace with a relation matrix and scores the fact triple by the distance of the two projected entities. Inspired by word vector analogies like (king-man+woman=queen), the TransE model (Bordes et al., 2013) estimates vector representations of entities and relations such as the sum of head entity and relation vector approximately equals the tail entity. A limitation of this model is that it cannot properly account for multiple facts with the same relation and one entity in common, as the model then assumes that all the entity vector for the differing entity should be identical. This limitation was addressed in several augmented versions of the model: TransH (Wang et al., 2014), TransR (Lin et al., 2015a) ,TransD (Ji et al., 2015),STransE (Nguyen et al., 2016a) by in-

troducing additional weight matrices or vectors to project entities into relation specific vectorial representations through different parametrizations.

More complex and non-linear scoring functions for KBC latent variable models can be formulated as Neural Networks. Knowledge Vault (Dong et al., 2014) scores triples with a single hidden layer MLP where entity and relation vectors are concatenated in the input layer, similar to the baseline version of the network presented in this work. The Neural Tensor Network of (Socher et al., 2013b) uses a non-linear scoring function parametrized by both a matrix and a third order tensor representation of relations.

Besides embedding and scoring triples, there has been research on usage of more complex graph substructures. The neighbourhood mixture model (Nguyen et al., 2016b) represents entities as the sum of an entity embedding and a mixture of vectors from paths of length one from the entity node. The mixture weights are estimated jointly with the latent feature vectors using the TransE scoring function. Paths between the two entities have been also utilized in latent feature models for KBC. The approach of (Neelakantan et al., 2014) extends the bilinear and TransE based models by applying their respective composition operator over the relations in a path. They show that including the paths in the training set results in better prediction accuracy for KBC. The PTransE model (Lin et al., 2015b) is another extension to TransE that composes paths using a composition operator. The paths are used in training phase by minimizing the distance of a relation to paths between entities and during inference of new edges by adding observed path information in the scoring of a triple. The composition operators considered were addition, multiplication and RNNs with addition performing better.

Evaluation of the above models in the FB15k dataset is inconclusive about which ones perform the best. Kadlec et al. (2017) showed that carefully tuning the simple models make them perform similar to the more complicated ones that were developed to address their shortcomings. In addition, Dettmers et al. (2017) found that because of the inverse relations in the dataset, a simple rule-based baseline can perform similarly to sophisticated models. To mitigate this phenomenon, the FB15k-237 dataset, which we use in our experiments, is recommended for evaluation.

5.2.2 KBC with text data

Text has been a very useful resource for KBC and has been utilized in multiple ways. In the most simple case, KBC models have used text descriptions from the KB to enhance entity representations. Initializing the entity vectors with the sum of word vectors of their name has been shown to be a simple and effective way to utilize such text descriptions (Socher et al., 2013b). The model of Xie et al. (2016) encodes entity representations from their text description found in the KB and then used in conjunction with KB entity embeddings in the TransE model. The advantage of such approaches are able to reason about unseen entities only given their textual description.

Model	Scoring Function
Rescal	$\mathbf{e_s R e_o}$
DistMult	$\mathbf{e_s} \odot \mathbf{r} \odot \mathbf{e_o}$
F	$\mathbf{e}_{(\mathbf{s},\mathbf{o})}^{\top}\mathbf{r}$
E	$\mathbf{e_s}^{\top}\mathbf{r} + \mathbf{e_o}^{\top}\mathbf{r}$
Structured Embedding	$\parallel \mathbf{R_s e_s} - \mathbf{R_o e_o} \parallel_{\mathbf{l1/l2}}$
Semantic Matching Energy	$(\mathbf{W_{1,1}e_s} + \mathbf{W_{1,2}r} + \mathbf{b_1})^{\top}(\mathbf{W_{2,1}e_s} + \mathbf{W_{1,2}r} + \mathbf{b_2})$
TransE	$\parallel \mathbf{e_s} + \mathbf{r} - \mathbf{e_o} \parallel_{\mathbf{l1/l2}}$
TransH	$\parallel (\mathbf{I} - \mathbf{r_p r_p}^{\top})\mathbf{e_s} + \mathbf{r} - (\mathbf{I} - \mathbf{r_p r_p}^{\top})\mathbf{e_o} \parallel_{\mathbf{l1/l2}}$
TransR	$\parallel \mathbf{R e_s} + \mathbf{r} - \mathbf{R e_o} \parallel_{\mathbf{l1/l2}}$
TransD	$\parallel (\mathbf{I} + \mathbf{r_p e_{sp}}^{\top})\mathbf{e_s} + \mathbf{r} - (\mathbf{I} + \mathbf{r_p e_{op}}^{\top})\mathbf{e_o} \parallel_{\mathbf{l1/l2}}$
STransE	$\parallel \mathbf{R_s e_s} + \mathbf{r} - \mathbf{R_o e_o} \parallel_{\mathbf{l1/l2}}$
Neural Tensor Network	$\mathbf{w}^{\top} tanh(\mathbf{e_s R^{[1:k]} e_o} + \mathbf{R_s e_s} + \mathbf{R_o e_o} + \mathbf{b_r})$
Knowledge Vault MLP	$\sigma(\mathbf{w}^{\top} tanh(\mathbf{W_s e_s} + \mathbf{W_r r} + \mathbf{W_o e_o} + \mathbf{b})$
path TransE	$\parallel \mathbf{e_s} + \mathbf{r_1} + ... + \mathbf{r_n} - \mathbf{e_o} \parallel_{\mathbf{l1/l2}}$
path bilinear	$\mathbf{e_s R_1 ... R_n e_o}$

Table 5.1: Overview of scoring functions of different latent variable models for KBC.

TEKE (Wang and Li, 2016) uses an annotated corpus where entity mentions are linked to KB entities to train a skip-gram model. Entity and relation enhanced representations are then constructed by adding to KB symbol vectors linear projections of averaged word vectors in context of entities or pairs of entities for relations. The text enhanced representations are used in a TransE model. Wang et al. (2014) train a probabilistic version of TransE on KB and text triples and also include mixed facts where some part of the triples contain KB symbols and some parts contain words. A modification was proposed by Zhong et al. (2015), where the KB-text alignment was defined by minimizing the distance between entity vectors and word vectors in their descriptions showing better performance. In general, Models that utilize text are shown to consistently outperform models that do not.

Another method to use text for KBC is by relation extraction through distant supervision. In this setting, the KB is used to create training samples in order to train a relation classification system. Given a KB and a text corpus, the first step is to link entities from text to those in the KB. This can be done by an entity linking system utilizing the surface form of entity mentions and potentially other features. Then for every pair of linked entities in the same sentence, training samples of relation instances are created by assuming that the expressed textual relation is the same as those expressed in the KB between the linked entity pair. This process creates a noisy dataset since the expressed relation may not be the assumed one. The concept was introduced by Mintz et al. (2009), where it was shown that such a dataset can be used to build a relation extraction system. Later work improved the process by taking into account the uncertainty in

label assignment. Riedel et al. (2010) used a latent factor model was used to infer if the entity pair participates in a relation from the KB, and Hoffmann et al. (2011) extended the model to account for multiple relations. Finally, Surdeanu et al. (2012) propose a probabilistic model latent variable model to formulate the problem as a multi-instance multi-label classification scenario, where joint inference is performed over all instances of an entity-pair. These series of modifications greatly increased the performance of relation extraction systems with distant supervision, with the model of Surdeanu et al. (2012) achieving state-of-the-art performance among them.

Distant supervision based relation extraction is not optimal for KBC as its goal is to build a text parser for the semantic relations expressed in the KB, and does not provide any reasoning capabilities. In addition, the internal structure of the KB is completely ignored. Instead of simply combining predictions from a relation extraction model and a KB reasoning model, a better approach is to jointly model the two tasks with a single system. By using the Universal Schema approach, the matrix factorization model of Riedel et al. (2013) includes textual and KB relations in a matrix to jointly embed them into a latent feature space and reason about unseen facts. The original model treats surface forms of relation mentions in text as a single token and assigns a latent feature representation to each one of them. In the work of Toutanova et al. (2015), latent representations of textual relations are composed by a CNN using the shortest dependency path which are trained jointly with the fact scoring model and can generalize better. Another related approach for joint KB and text embedding is the model of (Petroni et al., 2015), where facts are represented by sparse feature vectors with contextual information from the text corpus. The feature vectors are embedded into a low dimensional space by Factorization Machines (Rendle, 2010), a factorization method that models feature interactions through latent factors.

5.3 Model Definition

The proposed model in this work aims to utilize interactions between different information related to a potential fact. The core element of a fact is the $\mathcal{X} = (e_s, r, e_o)$ triple since it is the deciding factor of the truth value assignment. We assume that any source of additional information, either coming from the KB itself or from an external resource, is linked to at least one element of the triple. Thinking of the triple as the most simple structured feature we can extract form the KB, we can expand it with the linked information to create a new subgraph representing the fact with contextual information. We can now embed this expanded subgraph into a latent feature space and use a scoring function to estimate the plausibility of the fact.

This formulation gives a lot of flexibility for the type of information we can utilize to enrich the fact representation. In this work, we consider two kinds: Freebase entity type information extracted from the KB, and a corpus of text with entity mentions linked to the KB. The expanded subgraph can be seen in Figure 5.2. The Freebase types are used as a summary of the KB relations associated with the entities of a fact. While latent entity representations capture such

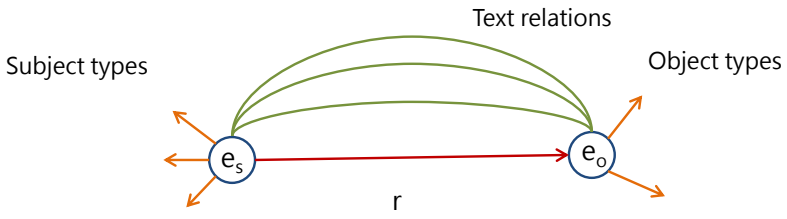
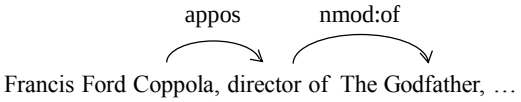


Figure 5.2: Expanded KB subgraph with linked resources.



type information, by using the Freebase type schema we provide additional explicit information related to the Freebase schema. The edges in the subgraph that represent text instances summarize all the available evidence from the text corpus. Making all this information explicit in the model, we can learn a non-linear composition function that combines the evidence, and is also aware of the absence of relevant evidence in cases where there is no text instance for a pair of entities. In addition, by aggregating all the information into the subgraph, we guarantee that every sample used for training the model is associated to a KB fact (which can either be a positive or negative sample), which is in agreement with the objective of a KBC model. This means that no effort is expended into modelling the plausibility of text relations independently of KB facts. The text relation instances can be larger in number than KB facts, leading to the need of additional hyperparameters to reduce their contribution in the loss function.

5.3.1 Feature Representation of the Joint Subgraph

There are many potential approaches to represent and score the plausibility of the KB-text aligned subgraph. We consider an efficient method that encodes the whole structure into a fixed size vector while retaining enough structural information. The subgraph is composed of 5 types of elements: the subject entity node, the object entity node, the relation edge, the subject Freebase

subject entity	`/m/02vyw`
object entity	`/m/07g1sm`
relation	`/film/director/film`
subject entity types	`/people/person` `/film/director/` `/award/award_winner`
object entity types	`/film/film` `/award/award_winning_work`
text features	`appos`$^{-1}$`_Esbj` `director` `appos_director` `of` `nmod:of`$^{-1}$`_director` `nmod:of_Eobj`

Table 5.2: Extracted features for a KB fact with a single associated textual relation mention.

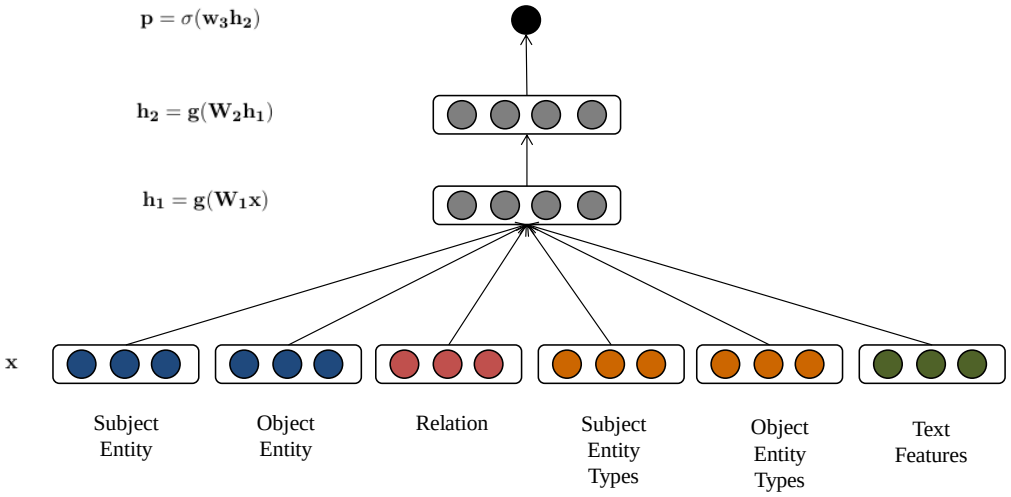


Figure 5.3: Feature-Rich Network with all the additional feature types associated with a fact.

types, the object Freebase types and the text relation mention edges. The subject, object and relation representations consist of a simple embedding vector e_{sbj}, e_{obj} and e_r respectively. We create compositional representations for the Freebase entity types and textual relation mentions with simple aggregation functions of their feature embeddings. Although not considered in this work, the overall approach is highly modular allowing for each component to be modelled by a different kind of network. An example of extracted features from a relation instance can be seen on Table 5.2.

Freebase Entity Types

Entities in Freebase can have multiple types assigned to them. While entity types are explicitly provided in Freebase, we instead learn type representations by only considering observed relations in the training set. Each relation in Freebase is encoded as a `domain/type/property`

of the subject entity. We extract the set of all triples where an entity takes the subject position in the training set, and keep the `domain/type` part as a type feature of that entity. We aggregate embeddings of all the observed discrete features using summation followed by L2-normalization to create the final representation of the subject and object entity's Freebase types: $v^f_{sbj}b$ and $v^f_{obj}b$. We use a special `UNKNOWN` symbol for entities with no observed types in the training set (i.e., entities that do not appear as subject of a triple).

Text Relations

We use a side corpus where pairs of entities are linked to the KB and take the shortest dependency path (see Chapter 3) connecting them as a textual relation mention. Since textual relations are linked to entity pairs, we collect all textual relation mentions for a given entity pair and associate them with a fact. This results in a collection of phrases that act as the complete set of textual evidence for all relations of an entity pair.

We create a representation of the associated text for each entity pair by using a Neural Bag of Words model with words and dependency features. Dependency features are defined as in Chapter 3. Similar to the Freebase Entity Type representations, embeddings of words and dependency features are aggregated by summation followed by L2-normalization, and a special `UNKNOWN` symbol is assigned to pairs of entities that do not have textual relation mentions. We use additional dependency features to indicate dependency relations with the two entities, such as E_s_nsubj and $E_o\_nmod:in$. We denote the resulting textual relation representation associated with a pair of subject and object entities as $v^{txt}_{sbj,obj}$.

The order that entity mentions appear in text does not indicate which one correspond to the subjects and object position of a KB relation. We treat textual relations as undirected and associate each one with facts containing both entity pairs (sbj,obj) and (obj,sbj). The entity dependency features are switched to indicate the correspondence of entity mentions with KB entities in a given tuple.

5.3.2 Feature-Rich Networks

We can put all the representations from the different elements of the subgraph into an n-tuple $\mathcal{X} = (e_{sbj}, e_r, e_{obj}, v^f_{sbj}b, v^{fb}_{obj}, v^{txt}_{sbj,obj})$. We want an architecture that can take into account rich non-linear interactions between the elements of the tuple. In addition, we want a model flexible enough to only use some of the elements or further expand the tuple. This would allow us to use the same model with only the basic representation of a fact expressed as (subject, relation, object) and assess the contribution of any additional information. It would also allow us to expand the subgraph with more linked resources by just adding more elements to the tuple.

By restricting the elements of the n-tuple to be fixed size vectors we can achieve this flexibility

with an MLP. The MLP learns to embed, compose and score the compatibility of the latent feature vectors of the tuple's elements. The composed vectors are concatenated to form a single vector $x \in R^k$ to become the input to the MLP. All the observed discrete features forming the elements of the n-tuple are jointly embedded into low dimensional spaces during training. The probability of a fact being true given all the evidence x is:

$$p(\mathcal{X} = 1) = \sigma(w_3 \cdot g(W_2 \cdot g(W_1 \cdot x)) \tag{5.1}$$

$$\mathbf{x} = [e_{sbj}; e_r; e_{obj}; v^f_{sbj} b; v^{fb}_{obj}; v^{txt}_{sbj,obj}] \tag{5.2}$$

where $W_1 \in R^{k1 \times k}$, $W_2 \in R^{k2 \times k3}$, $w_3 \in R^{k3}$ are the weights of the network, $g(\bullet)$ is a non-linear function applied element-wise, $\sigma(\bullet)$ is the sigmoid function. Rectified Linear Units are used as non-linearities (Nair and Hinton, 2010). Representations of entities and entity types are shared between subject and object positions. A diagram of the Network can be seen in Figure 5.2.

5.3.3 Training

The network weights are optimized by minimizing the binary cross-entropy loss over mini-batches using the Adam optimizer (Kingma and Ba, 2014). To avoid the large computational cost of training with all possible unobserved facts, we make use of negative sampling. The loss function is:

$$L(\Theta) = -\sum_{|\mathcal{X}_p|} \log p(\mathcal{X}_p) - \sum_{|\mathcal{X}_n|} \log(1 - p(\mathcal{X}_n)) \tag{5.3}$$

where Θ are all the parameters of the network including the feature embeddings, $\mathcal{X}_p$ are the observed facts in the training set and $\mathcal{X}_n$ are randomly drawn unobserved facts. We construct the negative samples by fixing the subject entity and relation, and uniformly sampling an object entity with the restriction that the resulting triple is not included in the training set. We then expand the triple with entity type and text representations. This negative sampling schedule follows the evaluation procedure, where the network has to rank triples that only differ in the object entity position. Experiments in the validation set indicated that for a fixed number of negative samples, only considering negative samples that differ in the object position performs better than also including negative samples for the subject position.

Initialization with Pre-trained Embeddings

We experiment with pre-trained embeddings to initialize the entity vectors and text feature embeddings of the model. Text feature embeddings are initialized with the extended dependency based skip-gram (Chapter 3). Features that are not included in the vocabulary of the pre-trained model are initialized with a random vector from a normal distribution with zero mean and same variance as the subset of pre-trained embeddings for features in the corpus. Entity dependency

features are initialized as the mean of all dependency embeddings having that relation. For entity vectors, we retrieve the English name of the entity from the latest version of Freebase and construct a representation by averaging the embeddings of the words appearing in the name. Entities that do not have a name property are initialized randomly.

5.4 Evaluation

5.4.1 Dataset and Evaluation Protocol

The FB15k237 dataset consists of about 15k entities and 237 relations derived from the FB15k dataset (Toutanova et al., 2015). This subset of relations does not contain redundant relations that can be easily inferred, resulting in a more challenging task compared to the original FB15k dataset. There are 310,116 triples in the dataset split into 272,115/17,535/20,466 for training/validation/testing. In addition to the KB, the dataset includes the shortest dependency paths of approximately 2.7 million relation instances of linked entity mentions extracted from the ClueWeb corpus (Gabrilovich et al., 2013). For compatibility with the pretrained dependency features, we converted the dependency types in the shortest dependency path from Stanford Dependencies (De Marneffe and Manning, 2008) to Universal Dependencies (De Marneffe et al., 2014) (Appendix B).

Evaluation follows the procedure of (Toutanova et al., 2015). Given a positive fact in the test set, the subject entity and relation are fixed and models have to rank all facts formed by the object entities appearing in the training set. The reported metrics are mean reciprocal rank (MRR) and hits@10.

$$MRR = \frac{1}{N}\sum_{i=1}^{N} rank_i \tag{5.4}$$

where $rank_i$ is the position of the positive fact in the ranked list and N is the number of facts in the test set. Hits@10 is the fraction of positive facts ranked in the top 10 positions.

The results reported are with the filtered setting. Some of the facts formed by the above procedure will be true facts from the training or validation set. Assigning them a high score and ranking high in the return list can push positive facts from the test set lower and affect the reported metrics. To prevent that we remove all positive facts in the training and validation set from the candidate fact list before ranking.

Implementation Details

Hyperparameters of the model were chosen by maximizing MRR on the validation set. We use two 300-dimensional hidden layers for the MLP, and dimensions of feature embeddings are: 300 for entity and text features, 100 for relations and 20 for entity type features. The number of negative samples was set to 20 as increasing their number only resulted in minor

	All		With Mentions		Without Mentions	
Model	MRR	H@10	MRR	H@10	MRR	H@10
KB only						
F	16.9	24.5	26.4	**49.1**	13.3	15.5
E	33.2	47.6	25.5	37.8	36.0	51.2
DistMult	35.7	52.3	26.0	39.0	39.3	57.2
E + DistMult	37.3	55.2	28.6	42.9	**40.5**	59.8
FRN trp	35.8	55.3	28.7	44.3	38.6	59.7
FRN trp + types	36.0	56.0	28.2	45.0	39.0	60.3
FRN trp + types + init	**37.6**	**57.5**	**30.5**	48.3	40.4	**61.1**
KB and text						
Conv-F	19.2	28.4	34.9	63.7	13.3	15.4
Conv-E	33.2	47.6	25.5	37.8	36.0	51.2
Conv-DistMult	36.6	53.5	28.3	43.4	39.7	57.2
Conv-E + Conv-DistMult	40.1	58.1	33.9	49.9	**42.2**	**61.1**
FRN trp + types + text	38.1	58.3	**45.4**	**68.8**	35.2	54.2
FRN trp + types + text + init	**40.3**	**62.0**	44.1	68.3	38.7	59.5

Table 5.3: Evaluation results on the FB15k-237 dataset. Results for F, E, DistMult and their CNN versions are reported from (Toutanova et al., 2015). With/Without Mentions indicates KB facts with/without aligned textual relations for their entity pair.

gains, and the batch size was set to 420. Best models were chosen among 20 epochs of training by monitoring validation MRR. Models with embedding initializations converged within the first 10 epochs. Initialization in the text model includes initializing entity and relation embeddings from the model without text mentions.

5.4.2 Results

We compare our Feature-Rich Networks with the bilinear models F and E (Riedel et al., 2013), model DistMult (Yang et al., 2014) and their CNN augmented versions (Toutanova et al., 2015). Results can be seen in Table 5.3. As explained in (Toutanova et al., 2015), results from model F are not directly comparable to others as it is only trained on the portion with mentions due to scalability issues from having to assign an embedding to every pair of entities.

We first observe that when modelling just KB triples, the MLP model outperforms individual bilinear formulations, performing similarly to the best combination of DistMult + E. This shows that an additive combination of bilinear models is a strong baseline even though it does not use additional parameters other than embeddings to compose and score triples. The addition of entity type information has a positive but small contribution to performance. This is not surprising as entity type information is extracted from observed relations, and latent feature models can

effectively learn that during training. On the other hand, initializing entity embeddings with averaged word embeddings of their names results in a substantial performance gain of about 1.5 points in both MRR and hits@10. In general, we observe that models perform worse on facts with textual relation mentions when they do not have access to those mentions.

When textual relation mentions are added, we observe that the proposed model increases its performance score about 3 points in MRR and 4.5 in hits@10 compared to the best model that does not include text. Contrary to the conv-bilinear models, the performance gain is much larger for facts with textual mentions, reaching an additional 15/20 in MRR/hits@10 respectively. We attribute this gain to the direct interactions between text and the other elements of the n-tuple which can make the correspondence between KB and text relations easier to capture, and to the non-linear composition by the MLP. The proposed model does not require an additional parameter to weigh the contribution of text and KB triples.

Initialization of embeddings results in substantial performance gain. We note that the convolutional bilinear models use pretrained word embeddings. For the model using KB and text, KB symbol embeddings are initialized from the model without text. Without initialization the model learns to utilize text effectively but does not perform as well on mentions without text. When initialized with pretrained embeddings, performance is more balanced. This suggests that it is beneficial to first model the internal structure of the KB and text separately, before trying to jointly embed the two sources.

5.5 Summary and Conclusion

In this chapter, a model for jointly embedding Knowledge Bases and text with Feature-Rich Networks was proposed and evaluated. The approach consists of constructing a subgraph representing all the associated information to a fact coming from the KB and external resources. Then the structure of the subgraph is encoded by composing representations first according to their types, and then all together to be scored by an MLP. The model can learn to utilize information from text better than bilinear latent feature models augmented with convolutional neural networks that reason about triples from the two sources individually. Besides text, we experiment with entity types and initialization with pre-trained embeddings, getting positive gains in performance. We conclude that additional structure coming from structural alignment between the KB and text can benefit learning joint representations.

The method provides a general framework for embedding linked resources in the same space and allows to efficiently model internal structure of the KB leaving room for a lot of additional experimentation. An interesting direction for future work is to combine the Feature-Rich models with additional linked resources, such as multiple KBs, in order to learn a mapping between their schemas and have more information for reasoning. Experimenting with different components for text encoding such as CNNs or LSTMs, and additional graph features such as paths between the

entity nodes are also promising extensions.

CHAPTER 6

Large-Scale Knowledge Base Question Answering with Joint Entity-Relation Identification

In this chapter we look at the second problem that requires joint modelling of KBs with text: answering questions expressed in natural language with entries from a Knowledge Base. We focus on the simple case where single facts from the KB can answer the question. The problem can be approached as a combination of entity linking and relation classification. Previously proposed systems either apply a pipeline approach where entity linking, retrieval and relation extraction are performed independently, or ignore structural equivalence between questions and queries. The proposed system takes an end-to-end approach, where the question is decomposed into entity and relation mentions, which are then mapped to the corresponding part of the query representation. Queries are encoded as a subgraph of the Knowledge Base. We also make use of multiple representations of the question and Knowledge Base parts and pretrain several parts of the network with auxiliary tasks. Evaluation on the SimpleQuestions dataset shows that the proposed system gives more accurate answers compared to state-of-the-art systems that perform the subtasks independently and systems that ignore the entity-relation structure.

6.1 Introduction

Question Answering (QA) is a popular topic of recent research as an evaluation for natural language understanding systems with many practical applications. Question Answering on Knowledge Bases (KBQA) is a specific QA setting requiring mapping questions expressed in natural language into queries to be executed against a Knowledge Base (KB). The questions are answered

by the retrieved list of entities or in the case of complex questions by a function applied to the list, such as counting or sorting. In this work, the focus is on the simple KBQA setting using the SimpleQuestions dataset (Bordes et al., 2015), where given a knowledge base consisting of facts encoded as triples of the form (subject, relation, object), questions can be answered directly by a single fact. For example, the question "What region is Oratino located in?" can be answered by retrieving a single fact from the KB, while the question "How many states border California and have population more than 1 million?" is a complex question that requires retrieval of multiple facts and application of reasoning. The simple KBQA setting is very important as statistics from users suggest that most questions fall under the simple type when dealing with large KBs (Berant et al., 2013), and it also forms the basic component of a complex KBQA system.

The simple KBQA task can be approached as the combination of entity linking and relation extraction problems. While these problems are well studied in the area of information extraction, the large scale of modern KBs poses many different challenges and current systems have still room for improvement. The first challenge a large KBQA system has to overcome is the large output space of possible queries. Typically there are millions of entities and thousands of relations in a large-scale KB resulting in a large search space of queries despite their simple structure. Standard classification methods are inapplicable and a ranking approach with some kind of pruning to restrict the number of entities is usually applied. Pruning techniques commonly consist of simple keyword search and heuristics. Another challenge is that training data will only include a small subset of entities and relations out of the total number present in the KB, requiring some zero-shot learning capabilities from systems as they must be able to generalize to unseen entities and relations during test time.

The state-of-the-art in simple KBQA consists of a pipeline approach incorporating typical steps of an information extraction system: entity recognition, entity retrieval and ranking, and finally relation classification (Yin et al., 2016; Yu et al., 2017). It is common to use the output of the previous system to simplify the problem, substituting for example the entity mention in the question with a generic entity symbol "ENT" after entity identification, so that the relation classifier will only focus on the relation mention part of the question. The subsystems of the pipeline are trained independently without sharing parameters and only their scores are combined to produce the most likely answer, i.e. they perform joint decoding by assuming independence and not joint learning.

Though mapping the entity and relation independently is efficient and can utilize known methods of entity linking and relation extraction, it has long been observed that joint learning of entity and relation classification is superior to pipeline approaches since it allows for mitigation of error propagation (Roth and Yih, 2007; Li and Ji, 2014). This becomes evident in the case of simple QA when considering the ambiguity of relations in questions such as "in what country was ENT shot", where the corresponding Freebase relations could either be `film/film/country` indicating shooting of a film, or `people/person/place_of_death` implying shooting of a

person. Similarly, detecting entities with complex names such as titles (for example, “What is a track featured on the album doesn’t play well with others”) requires the ability to distinguish between phrases resulting in valid interpretation of relations within the KB (“featured on the album”) and phrases that cannot be interpreted (“album doesn’t play”).

KBQA systems that jointly map entities and relations have been evaluated in the SimpleQA setting, but have so far been performing poorly compared to the pipeline approach (Bordes et al., 2015; Joulin et al., 2017). We hypothesize that poor performance is attributed to lack of modelling structural information about the problem. The pipeline approach makes extensive use of the knowledge that the question can be decomposed into an entity and relation mention, and that the two mentions should be mapped to their respective subject and relation symbols of the KB using appropriate features for each type. In contrast, previous work that performed joint modelling did not identify the position of entities in the question.

In this work, we propose an approach for simple QA where entity and relation mentions from the question are being extracted and compared to their respective parts of a possible query with a single network. The decomposition of the question and scoring is performed jointly by a single network that utilizes the structural constraints of the problem. Central to the proposed approach is the use a gating mechanism implemented as a feature-rich attention network to decompose the question into an entity and relation mention part. The attention network estimates the probability of a token in the question to be part of the entity mention and as a consequence the probability of being part of the relation mention. The attention probability estimates are then used in multiple ways. First, they act as a gating mechanism to aggregate character n-gram embeddings from the question and compute a similarity with the candidate entity names. They are then used along with word and character representations of the question as an input to a BiLSTM encoder. Finally, they are used to pool the hidden states of the BiLSTM into different subject entity and relation representations. The output of the full network is the probability that the question maps to a candidate query computed as a function of similarity between corresponding entity and relation parts of the question and KB.

Contrary to the pipeline approach, decomposition of the question into entity and relation mentions is implemented as a soft decision so that the network can be trained end-to-end with standard backpropagation. Pruning of candidate entities is performed by the part of the system that computes similarity between entity names in the KB and entity mention in the question in a similar way that pipeline approaches use a separate system to retrieve and rank entities. The model is trained without the need to provide annotations for the entity mentions in the question. Such annotations are not available for the SimpleQuestions dataset and have to be generated by distant supervision. The proposed network makes use of multiple representations of the question on character, word and syntactic level, and multiple representations of the KB symbols.

We also explore pretraining strategies with auxiliary tasks in order to deal with unseen entities and relations in the test set and also provide the necessary structural bias to the network for

performing the entity-relation decomposition. Word, character and syntactic embeddings are pretrained with the skip-gram objective, and KB symbol embeddings are pretrained by performing Knowledge Base Completion. In addition, initializing the attention network weights by a model trained for entity linking was found to provide considerable benefits in overall performance.

The approach taken in this chapter relies on the hypothesis that information about the alignment of text and KB with respect to entity and relation aspects is valuable for systems that need to jointly model the two sources. Therefore, it provides experimental evidence for answering the second research question in the setting of querying a KB in natural language. Contrary to the KBC setting in chapter 5, the boundaries of entities and relations are not known and the system needs to learn that itself without explicit supervision. We show that pretraining is a viable method to bias the network towards performing the correct decomposition. A benefit of comparing representations of text and KBs in their corresponding entity and relation aspects is that we can use appropriate features for each aspect. That is, we match the surface form of the KB entity name to the entity mention with latent features encoding string similarity, and the relation description in the word semantics level. The central component of the system, which is the attention network performing the entity-relation decomposition, is a very simple MLP that receives all the necessary information about the structure of the question through feature embeddings.

We perform evaluation on the SimpleQuestions dataset comparing with previous results for entity linking, the full KBQA setting and a reranking setting. By using a joint approach consisting of retrieval, entity linking and relation extraction we show that we get improved performance both for an entity linking evaluation and the full KBQA setting compared to pipeline systems. We also outperform by a large margin all the system that model the problem end-to-end without considering the entity-relation structural alignment between questions and the KB.

6.2 Related Work

The first results on the SimpleQuestions dataset were obtained by Memory Networks (Bordes et al., 2015). Memory Networks are models that can store information in memory, process it to address a specific task and return an output. They consist of four modules: input, generalization, output and response. For the simple KBQA task, those modules were used to store Freebase, retrieve candidate entities given a question and score possible queries containing those entities with the question. The representations used were bag-of-symbols and bag-of-ngrams for the Freebase subgraphs and questions respectively, embedded into a low dimensional space and scored using cosine similarity. While the Memory Networks trained for SimpleQA scored entities and relations jointly, applied multi-task training along with data augmentation and model ensembling, the accuracy they obtain is relatively low since the scoring component is very simple. FastText (Joulin et al., 2017) takes a similar approach using simple token matching to retrieve candidate entities and a linear classifier based on averaged embeddings to classify relations achieving about

10% increased accuracy compared to Memory Networks.

The work of Golub and He (2016) demonstrated the importance of using character level representations for simple KBQA by proposing a character-level encoder-decoder. Their model consists of a character-based LSTM encoder to encode the question and two character-based CNNs to create representations of entities from their names and of relation symbols from their human readable string representation (e.g., "/location/location/containedby"). In order to match questions to queries, they use an LSTM decoder with attention. Experiments with the character level encoder-decoder showed that character level representations are very important for modelling entities since they contain a large number of rare and misspelled words that makes word embedding difficult.

A large improvement in performance was obtained by the Conditional Focused neural model (Dai et al., 2016) that improves the pruning of subject entities by first identifying the entity mention in the question. The authors train a Named Entity Recognition model consisting of an LSTM with a Conditional Random Field on top in order to perform structured prediction for boundary detection. They experimentally showed that focused pruning, i.e. using only the tokens of the identified entity mention provides much better coverage. Their question-to-query mapping model consists of two BiGRU networks that map the question to relation and subject entity. They propose a conditional probabilistic model that first maps the relation given the question and then maps the subject entity given the inferred relation and question. However, the two networks do not share parameters and the conditioning only accounts for the trivial case of discarding $(subject, relation)$ pairs not in the KB.

Following the focused pruning method for candidate selection, Yin et al. (2016) use a similar NER model but improve entity linking by applying computing the least common subsequence between the question entity mention and entity names and applying additional heuristics for scoring. Their entity linker outperforms other candidate generation methods by a large margin. The rest of their approach makes use of a character-level CNN to rerank candidates and a word-level CNNs with an attention driven pooling mechanism to score relations.

The model of Yu et al. (2017) is based on an improved relation classification network: a deep BiLSTM with residual connections called hierarchical LSTM. By applying this relation detector to the entity linking results of Yin et al. (2016), they report the best accuracy for this dataset and this type of evaluation. Their work hints at the importance of joint entity relation modelling by showing that a two stage reranking, once with the full question and once with entity mention substitution improves accuracy.

Besides systems designed for the SimpleQuestions setting, work in joint entity-relation extraction for information extraction is also relevant to this work. Roth and Yih (2007) used integer linear programming to jointly estimate a structured output of extracted entities and relations, while Li and Ji (2014) used transition-based parsing to construct the output structure. Joint entity-relation extraction with neural models have been explored by Miwa and Bansal (2016),

using sequence and tree structured networks on top of each other for the two subtasks. Zheng et al. (2017), approached joint entity-relation identification as a sequence labelling problem using a special tagging scheme that captures information about the two tasks simultaneously. In all of the evaluated cases, systems that perform the two tasks jointly showed improvements compared to pipeline approaches. We note however, that joint extraction makes the problem much more complex as the desired output structure grows with combinatorial complexity.

6.3 Simple Questions Dataset

The SimpleQuestions dataset is the largest available KBQA dataset with human generated questions split into 75197/10840/21687 train/validation/test sets. It comes along with two subsets of Freebase as the background KB for answering the questions: Freebase-5M (approximately 4M entities, 7500 relations and 22M facts) and Freebase-2M (approximately 2M entities, 6700 relations and 14M facts). All questions in the dataset can be answered by a single fact. Every question contains an entity mention and a relation mention that can be mapped to a query of the form (subject, relation, ?), hence only mapping the subject and relation is sufficient to answer the question.

While we only need the subject and relation to successfully answer a question by querying the KB, we make use of the answer as well. The answer is the set of all object entities in the KB triples with the same (subject, relation) as the query being mapped to the question. The reason we also make use of the answer is that there is often evidence in the question about the expected type of the answer. For example, in the question "What region is Oratino located in?" the phrase "what region" provides valuable information about the type of entities that can be considered as answers to the question. The answer representation is constructed by averaging all the object embeddings in the subgraph.

Facts in Freebase are encoded as (subject, relation, object) triples, but since only the (subject, relation) part is relevant to form a query, a useful representation is to create a larger subgraph were all the objects are aggregated into (subject, relation, [object_1, object_2, ..., object_n]) tuples. While Freebase-5M contains about double the number of entities compared to Freebase-2M, the number of total subgraphs is very similar as most additional entities occupy the object position. and most of the $(sbj, relation)$ pairs are common between the two subsets (99% overlap). This observation suggests that SimpleQA on Freebase-5M and Freebase-2M has a similar difficulty when merging triples into subgraphs.

In order to map the entity mention to an entity in Freebase we need a text description of the entity. The final version of Freebase was used to extract the `/type/object/name` and `/common/topic/alias` properties for all the entities in English. If there is no available English name or alias, names from all other languages were extracted instead. This additional resource extends KB subsets with relations that map each entity to a set of strings E_{names}.

6.4 Model Description

The proposed network is composed of several modules that aim to compute the similarity of the question with a subgraph of the KB. The model learns to decompose the question into an entity and relation representation to be compared with the corresponding parts of the query graph. Similarities are computed using multiple views for both the question and the KB subgraph. The question is encoded using words, syntactic features and character n-grams, entities are represented both by a low-dimensional embedding encoding their relations in the graph and a text description, and similarly relations are represented both by their KB embedding and a textual description.

6.4.1 Feature Embedding Spaces

n-gram	5 most similar
“tion”	“tio”, “tion ”, “ion”, “ation”, “atio”
“155”	“ 155”, “157”, “115”, “175”, “153”
“74-19”	“74-197”, “74-1”, “974-19”, “974-1”, “74-198”
“of a”	“ of a”, “f a”, “ of an”, “of an”, “f an a”
“0.1”	“ 0.1”, “0.1 ”, “ 0.1 ”, “ 0.”, “0.15”

Table 6.1: String similarity properties exhibited by n-gram embeddings after training with the skip-gram objective.

word	dep	n-grams	positions
where	advmod^{-1}_born	‘whe’, ‘wher’, ..., ‘ere wa’	(1,5)
was	auxpass^{-1}_born	‘re was’, ‘e was ’, ... , ‘was jo’	(2,4)
john	nmod:npmod^{-1}_miltern	‘as joh’ ,‘s john’, ... , ‘ohn mi’	(3,3)
miltern	nsubjpass^{-1}_born nmod:npmod_john	‘hn mil’ ,‘n milt’, ... , ‘ern bo’	(4,2)
born	advmod_where, auxpass_was, nsubjpass_miltern	‘rn bor’, ‘n born’, ... , ‘orn’	(5,1)

Table 6.2: Features extracted for a question.

Word and Dependency Feature Embeddings

The pretrained Extended Dependency Based Skip-gram (chapter 3) vectors of words and dependency features ($e_w, e_d \in R^{300}$) were used. Words and dependency features not in the vocabulary of pretrained embeddings but appearing in the training questions set more than 3 times were randomly initialized from a normal distribution with zero mean and same variance as tokens in the

pretrained set and added to the set of embeddings. In addition, "unknown word" and "unknown dependency" tokens were added after being initialized by the mean of pretrained tokens of the corresponding type with frequency equal to 3 or less in the training set.

String N-gram Embeddings

Character based representations are important for representing entity names as they often consist of rare words or are misspelled. To show the extent of the problem, we mention that there are 800K unique tokens in the collection of entity names while only 220K word embeddings are available in the Extended Dependency Based Skip-gram vectors. The character representations used in this work come from a simple character n-gram embedding model. To extract the raw features, a string is first converted to lower-case and spaces are normalized by a tokenizer, then all n-grams with $n \in \{3, 4, 5, 6\}$ are extracted. The resulting n-grams span across word boundaries and contain spaces. The intuition behind having n-grams span across words is that entity names are often multi-word expressions, and extracting n-grams across words encodes their order. The n-gram vocabulary was built from the questions in the train set and the Freebase entity names keeping n-grams that appear more than 3 times. The embedding dimensionality was set to 100 ($e_n \in R^{100}$) and the total number of n-grams in the vocabulary is approximately 3.7 million. The n-gram embeddings were pretrained using a window based skip-gram model with negative sampling (k=15) and window size 30 for 10 epochs on training questions and entity names text. Sequences from overlapping n-grams were formed by arranging them according to the position of their first character in the text and from smaller to larger n. The relatively large window size of 30 was chosen to provide some non-overlapping context. The resulting embeddings capture properties of string similarity and character semantics. Some examples of most similar n-grams in the embedding set are shown in Table 6.1. This character based model is computationally efficient compared to CNN and LSTM based character representations as it can create representations for larger strings by simple averaging of embeddings, and experiments suggest it is very accurate in retrieval of similar strings. Similar to word embedding models, the unsupervised pretraining mitigates the drawback of introducing a large number of parameters to the model.

Bidirectional Positional Embeddings

The relative position of each token in the question is encoded using positional embeddings. We make use of a forward positional feature indicating the position of a token from the beginning of a sequence with s tokens as $1, 2, ..., s$ and a backwards positional feature indicating the position from the end of the sequence $s, s-1, ..., 1$. The positional features are treated as discrete tokens and are embedded to a lower dimensional space $e_p^f, e_p^b \in R^{10}$. To ensure generalization to sequences longer than those observed in the training set we set a maximum position equal to 20 for both directions and for longer sequences it is repeated. This bidirectional encoding can indicate word positions in the question that may have special purpose such as the first and last

word. When the two directions are used together, they encode relative position of a word from the beginning and end of the question. This can be useful as entity names are more likely to occur towards the end of the question.

Freebase Symbol Embeddings

The Freebase symbols consist of entity identifiers (Freebase mids), such as `m/0kpg4`, and relation symbols such as `film/film/written_by`. Subject entities, object entities and relations are correspondingly embedded as $e_m^{sbj}, e_m^{obj}, e_r \in R^{100}$. In addition to assigning an embedding to relations treating them as symbols, we make use of the human readable description and decompose them to words by separating the string between all "/" and "_" characters. We then obtain another embedding of relations $v_r^w \in R^{300}$ by averaging word embeddings e_w of the words found in the description. This leads to two representations of relations, one as an atomic KB symbol and one as a composed representation in the same space as question word embeddings, which can make the semantic mapping between KB and text relations easier.

6.4.2 Network Components

Gating Attention

The attention is the most important part of the network as it is responsible for decomposing the question into the entity and relation part to be compared with the corresponding parts of the KB subgraph. It takes as input a feature-rich representation of the question and returns a sequence of probabilities indicating which words are part of the entity mention. We encode all structural information about a word in the question through features and then use a simple model that does account for any structure (unlike RNNs or CNNs) to identify the entity mention subsequence of words.

The question representation consists of a sequence of vectors x_i^{att} associated with each word i of the question. Each of these vectors is formed by composing the word (e_w), dependency (e_d), n-gram (e_n) and positional embeddings (e_p^f, e_p^b) associated with the word. In Table 6.2, we show an example of all the features extracted from a question.

N-gram embeddings from the whole question string are assigned to individual words that contain the most characters of the n-gram and the mean of their embeddings forms a character-based vector representation of a word:

$$v_i^{ngram} = \frac{1}{M} \sum_m e_{n_{i,m}} \tag{6.1}$$

where M is the total number of n-grams assigned to word i. Since n-grams span across word boundaries, these vectors contain contextual information (equal to 3 characters at most) from substrings of surrounding words. While the number of characters is limited, they are enough to

identify most surrounding function words like prepositions and determiners that carry information about the syntactic role of the word in the question.

A dependency feature vector for each word is formed by the mean of dependency embeddings associated with that word:

$$v_i^{dep} = \frac{1}{K} \sum_k e_{d_{i,k}} \tag{6.2}$$

where K is the total number of dependencies of word i in the dependency graph.

The input vector for a word is formed by concatenating the feature vectors of each type to form a vector of dimension l:

$$x_i^{att} = [v_i^{ngram}; v_i^{dep}; e_{w_i}; e_{p_i}^f; e_{p_i}^b] \tag{6.3}$$

The attention is implemented as an MLP with a hidden layer of dimension k, a relu nonlinearity in the hidden layer and a sigmoid output activation:

$$p_i^{att} = \sigma(w_2 relu(W_1 x_i^{att} + b_1) + b_2) \tag{6.4}$$

where $W_1 \in R^{k \times l}, b_1 \in R^k, w_2 \in R^k, b_2 \in R^k$.

While the attention weight is computed on a single token basis, sufficient contextual information is provided by dependency, cross-word character n-grams and positional embeddings.

Question Encoder

The question is encoded into two latent representations, one for the entity mention and one for the relation mention. The input is a sequence of vectors consisting of the word embedding, the n-gram vector of each question word and the probability computed by the attention network for this word:

$$x_i^q = [v_i^{ngram}; e_{w_i}; p_i^{att}] \tag{6.5}$$

A bidirectional LSTM is used to transform the sequence of input vectors $\mathbf{x}$ to a sequence of contextualized vectors h:

$$\begin{aligned} h_i &= [h_i^f; h_i^b] \\ h_i^f &= LSTM_f(x_i, h_{i-1}^f) \\ h_i^b &= LSTM_b(x_i, h_{i+1}^b) \end{aligned}$$

The entity and relation mention representations are given by a weighted sum of elements of h followed by a dense layer with a $tanh$ activation. The weights are determined by the probabilities

from the attention network.

$$h_q^{sbj} = tanh(W_q^{sbj} \sum_i \frac{p_i^{att} h_i}{||p_i^{att} h_i||_2}) \tag{6.6}$$

$$h_q^{rel} = tanh(W_q^{rel} \sum_i \frac{(1 - p_i^{att}) h_i}{||(1 - p_i^{att}) h_i||_2}) \tag{6.7}$$

where $W_q^{sbj}, W_q^{rel} \in R^{k \times dim_h}$ The intuition behind the above equations is that the output of the attention network indicates the probability p of a token being part of the entity mention, and with $1 - p$ the probability of being in the relation mention. These probabilities can be used to as a gate to get an entity and relation representation of the question. The token representations being averaged are the states of the biLSTM network, making them carry information about the order of the tokens in the sentence. The final non-linear transformation aims to project the entity and relation representation in a new common space with the corresponding KB entities and relations.

KB Subgraph Encoder

Each possible query is represented by a subgraph of the form: (subject, relation, [object_1, object_2, ..., object_n]). The subgraph is encoded into a subject representation and into a relation representation that also includes the information of the answer as the objects entities. The subject representation is given by:

$$h_{kb}^{sbj} = tanh(W_{kb}^{sbj} \frac{e_m^{sbj}}{||e_m^{sbj}||_2}) \tag{6.8}$$

The relation part is computed by considering the atomic representation of a relation e_r, the mean of word embeddings of the words in the relation string v_r^w, and the mean of entity embeddings taking the object positions in the subgraph. The three vectors are concatenated and passed to a dense layer:

$$x_{kb}^r = [e_r; v_r^w; \frac{1}{M} \sum_i e_{m_i}^{obj}] \tag{6.9}$$

$$h_{kb}^{rel} = tanh(W_{kb}^{rel} \frac{x_{kb}^r}{||x_{kb}^r||_2}) \tag{6.10}$$

The non-linear transformations aim to project the KB entities and relations to the same space as the question representations.

Entity Name Similarity

The sequence of Bernoulli probabilities provided by the attention can also act as a gate to n-gram embeddings at the corresponding word positions and is used as a pooling operation to form a

weighted average of n-gram embeddings of the entity mention:

$$u^q_{name} = \sum_{i,m} p_{att_i} e_{n_{i,m}} \tag{6.11}$$

This character based representation of the entity is compared with the name descriptions of the entities, which are also encoded in the same way as averaged n-gram embeddings:

$$u^{kb}_{name} = \sum_{m} e_{n_m} \tag{6.12}$$

The similarity between the question entity mention and the name of a KB entity is then given by:

$$sim_{name}(u^q_{name}, u^{ent}_{name}) = \max_{u^{kb}_{name} \in KB_{names}} \{cos(u^q_{name}, u^{kb}_{name})\} \tag{6.13}$$

This is the maximum cosine similarity between the question entity mentions and any of the known aliases of the entity in the KB.

Mapping Questions to Queries

The final outcome of the model is the probability of a question being correctly mapped to a query. We obtain that by computing similarity and distance features between the encoded representations of corresponding parts between the question and possible queries. We use the following function to generate similarity features:

$$f(x, y) = [x \odot y; |x - y|] \tag{6.14}$$

The final probability is given by an output layer that receives the similarity features of corresponding entity and relation alignments between the question and the KB, as well as the entity name similarity:

$$p(question \mapsto query) = \sigma(w[f(h^{rel}_q, h^{rel}_{kb}); f(h^{sbj}_q, h^{sbj}_{kb}); sim_{name}]) \tag{6.15}$$

The full architecture of the system can be seen in Figure 6.1.

6.5 Training

We experiment with pretraining strategies and show that they can have a big effect on the final accuracy. Pretraining parts of the model with auxiliary tasks was crucial for models to converge to high accuracy and to speed up the training. In addition to skip-gram based pretraining of the text features embeddings, two auxiliary tasks were used: knowledge base completion to pretrain the KB symbol embeddings and entity linking to pretrain the gating attention network.

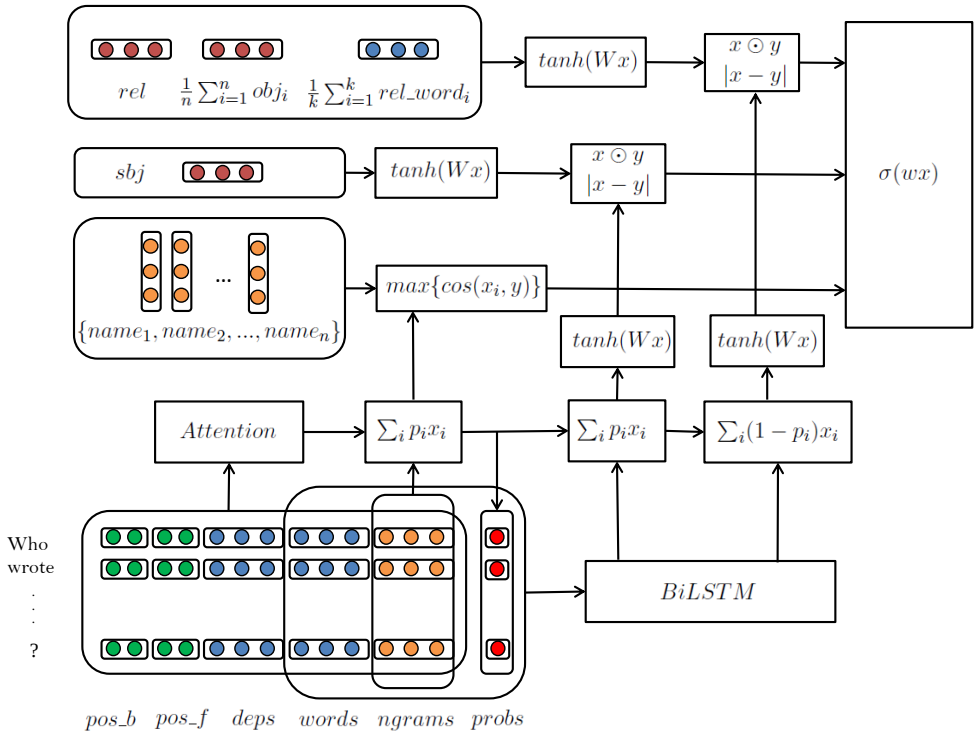


Figure 6.1: Architecture of the KBQA system.

Pretraining the Entity Name Similarity Network

We pretrain the entity name similarity network by performing entity linking given the question and gold entity names. The most important operation of the similarity network is being able to train the attention to identify the entity mention tokens. The available annotations do not provide the span of the entity mention in the question. Previous work (Yin et al., 2016) used the distant supervision heuristic to find the mention in the question based on the gold entity name(s) and trained a sequence labelling model for Named Entity Recognition. In contrast, we train the network to assign high similarity between the extracted entity mention and gold entity names. In order to maximize this similarity, the attention network is forced to correctly learn to identify the entity mention in the question. The network is trained by minimizing the following loss:

$$L_\theta = \alpha \cdot y \cdot (1 - sim)^2 + (1 - y) \cdot \max(sim - m, 0)^2 \tag{6.16}$$

where sim is the similarity as defined in equation 5, $y \in 0, 1$ is the label, α is a scaling parameter for weighting the contribution of positive and negative samples, and m is a margin hyperpara-

Hyperparameter	value
Attention h_dim	100
BiLSTM h_dim	300
dense layers dim	300
n-gram embedding dim	100
positional embedding dim	10
KB symbol dim	100
SGD learning rate	0.5
batch size	505 (5×101)
negative samples	100
max number of epochs	30

Table 6.3: Hyperparameter values of the KBQA system.

meter. The loss tries to force the similarity between incorrect pairs to be less than the margin. In our experiments, we used $\alpha = 10$ and $m = 0.5$ based on entity linking performance on the validation set.

To speed up the training we only consider entity names for negative samples that share n-grams with the question. In particular, all the name strings were initially ranked according to the cosine similarity between the whole question and entity names, using sparse binary vectors where dimensions correspond to the n-gram features. For each question, the most similar 100 name strings were chosen as negative samples.

Pretraining Freebase Symbol Embeddings

Out of the 4M entities and 7500 relations in Freebase-5M, about 60K entities and 1600 relations appear on the training set. This makes pretraining the embeddings of Freebase symbols an important step to ensure proper generalization as they cannot be accurately estimated by the training set alone. Embeddings of the KB are trained on the task of Knowledge Base Completion using the Freebase-5M subset. We define a network that learns to distinguish proper subgraphs contained in the KB from random perturbations. The network computes the probability:

$$p(SG \in KB) = \sigma(wrelu(W_s e_s + W_r e_r + W_o \sum_o e_o)) \tag{6.17}$$

For every subgraph in the KB, 10 negative samples are constructed by replacing the correct subject, and 10 negative samples by replacing the relation sampled uniformly from the set of entities and relations respectively. The network is trained by minimizing the binary cross-entropy loss and the Adam optmizer (Kingma and Ba, 2014). The best model is selected by measuring the F-score on distinguishing subgraphs from the KB and corrupted ones as in the training stage

what	is	the	release	type	of	the	album	the	soul	sessions
0.06	0.13	0.02	0.00	0.00	0.11	0.04	0.02	0.63	0.81	0.80

where	was	mr	music	hits	1992-5	released
0.12	0.07	0.69	0.60	0.68	0.73	0.00

what	type	of	music	does	david	ruffin	play
0.03	0.00	0.02	0.00	0.09	0.94	0.93	0.00

where	in	the	us	is	fontana	located
0.05	0.16	0.15	0.12	0.14	0.90	0.00

what	album	did	thad	jones	release	in	1957	?
0.13	0.11	0.14	0.83	0.80	0.03	0.00	0.09	0.12

who	is	the	author	of	a	knot	in	the	grain	and	other	stories
0.06	0.10	0.07	0.00	0.02	0.33	0.63	0.18	0.42	0.82	0.72	0.72	0.74

name	a	composer	in	the	album	,	``	the	great	composers	iv	``	.
0.19	0.00	0.17	0.18	0.02	0.03	0.07	0.00	0.57	0.65	0.63	0.79	0.00	0.00

Figure 6.2: Examples of the attention weights on several sentences. The attention network is able to identify difficult entity mentions of titles.

on a held out validation set consisting of 10% of the total subgraphs.

Full Model Training

The full network is trained using binary cross-entropy loss. Training was performed in two stages. First, all the parameters of parts that have been pretrained were frozen and the rest of the network parameters were updated using the Adam optimizer. The best performing model in the validation set within 10 epochs of training is picked. A second stage of training is then applied where all the parameters except the entity embeddings were being updated using simple stochastic gradient descent. We found that updating the entity embeddings quickly led to overfitting, which is not surprising as they consist of 400M parameters. This indicates the importance of the pretraining with KBC as entity embeddings are not only difficult to learn from random initializations, but due to the number of entities they are even difficult to fine tune without overfitting. The best performing model in validation set within 20 epochs is chosen as the final model.

Another training decision important for fast convergence is choosing appropriate negative

samples. For each (pos_subject, pos_relation) pair we select 100 negative samples with some restrictions. Using the pretrained Entity Name Similarity network we select the top 100 subjects with the most similar name to the entity mention in the question. The negative samples are then formed by considering two cases: queries formed by positive subject and an incorrect relation, and queries formed by one of the top-100 ranked subjects and a random relation. At most 20 samples are used for the first case and samples for the second case are picked to complete a set of 100. In a few cases where not enough negative subjects with similar name were able to be identified, queries were sampled uniformly from the pool of available queries to complete the negative sample set. In all of the above cases, subjects and relations cannot vary independently and every negative sample has to correspond to a valid subgraph of the KB.

The network has many hyperparameters to be tuned. Performing grid search even with small sets of values for each hyperparameter is very expensive. We started by setting reasonable values to every hyperparameter and then varied each one independently. For hyperparameters that had large impact on validation accuracy, more values were tried, while for others that seemed to have little effect the default value was kept. The final values of hyperparameters can be seen in Table 6.3.

6.6 Evaluation

We conduct experiments to evaluate the effectiveness of the proposed model by comparing with models in the literature using the same evaluation settings. We first show results on Entity Linking by using the Name Similarity Network to rank entities from Freebase-2M. Then we show final results of the full model. We also consider a reranking evaluation setting in order to compare with the results of Yu et al. (2017). We follow the evaluation procedure described by Yin et al. (2016). We take each query as correct if both subject entity and relation match the gold annotations. Since applying the full network to every possible query is very expensive, we follow the typical procedure of first creating a candidate set of subject entities and then performing the full ranking using all valid relations of the top 20 candidate subjects. We use the entity name similarity network to rank the candidates. We compare the performance of the similarity network for candidate generation to the state-of-the-art linker of Yin et al. (2016) that uses an LSTM-CRF to locate the entity mention and then string matching operations with heuristics to compute a similarity score with entity names. Finally, we compare our joint entity-relation ranking by the full network to previously proposed systems.

6.6.1 Entity Linking Evaluation

Following previous work we report results on the test portion of the data with Freebase-2M as the background KB. We look at subsets of the top-N results were $N \in \{1, 5, 10, 20, 50, 100\}$

top-N	Freebase API	CFO Linker	Active Linker	Attention Linker
1	40.9	52.9	73.6	**74.4**
5	-	-	85.0	**87.0**
10	64.3	74.0	87.4	**89.6**
20	69.7	77.8	88.8	**91.5**
50	75.7	82.0	90.4	**93.2**
100	79.6	85.4	91.6	**94.5**

Table 6.4: Entity linking results on the test set of Simple Questions with Freebase-2M as the background KB comparing the proposed Attention Linker to the CFO linker of Dai et al. (2016) and the Active Linker of Yin et al. (2016).

and consider a success if the positive subject is in the subset. Many of the correct subjects share the same name with a large number of other entities. Since all of the linkers compared only consider string similarity for ranking, different resolution of ties can affect the overall results, an observation also mentioned by Joulin et al. (2017). To avoid arbitrary ranking in the case of ties, we resolve ties by favouring entities appearing with more relation types in the KB. We also used an additional pruning strategy: only entities that have a common word with the question except for a list of stopwords are considered. The stopwords are the 50 most common words in the training set of questions which are not content words (Appendix A). This simple pruning mechanism discards on average about 90% of possible entities without hurting performance. Ranking without pruning showed about 0.1% higher accuracy. Responsible for this gain were the rare occasions where every word of the entity name was misspelled.

Results in the entity linking task show that the proposed attention linker with n-gram embedding similarity clearly outperforms all previously proposed linkers (Table 6.4). The differences in performance with the linker of Yin et al. (2016) are significant for every N reported ($p < 0.05$, test of equal proportions). We note that entity linking performance is very important for the overall performance of the system as only a small subset of candidate subjects is jointly ranked by the full model.

6.6.2 Full KBQA Evaluation

The full model was evaluated using both the Freebase-2M and Freebase-5M subsets as the background KB. The same model trained with Freebase-5M is used for evaluation with both subsets. We evaluate in two settings. The first setting follows the evaluation protocol for full large-scale KBQA. The name similarity network ranks all the subject names in the KB according to string similarity with the entity mention in the question. The top-20 ranked subjects are used to extract the set of all queries (subgraphs) that have this entity as subject in order to be ranked by the full model. Systems are ranked by accuracy, the portion of questions for which the correct query was

system	FB-2M	FB-5M
Memory Network (multi-task, ensemble) (Bordes et al., 2015)	62.7	63.9
Character Encoder-Decoder (Golub and He, 2016)	70.9	70.3
fastText (Joulin et al., 2017)	72.7	-
CFO (Dai et al., 2016)	-	75.7
Attentive CNN (Yin et al., 2016)	76.4	75.9
This work	**77.8**	**77.8**
HierLSTM (reranked) (Yu et al., 2017)	77.0	-
This work (reranked)	**78.0**	-
HierLSTM (reranked, ensemble)	**78.7**	-

Table 6.5: Full model results on Simple Questions compared with previously proposed models in different evaluation settings.

Ablation	Freebase-5M accuracy
no entity linking pretraining	75.9
no words, dependencies for attention	76.9
no dependencies for attention	77.4

Table 6.6: Ablation tests of specific components of the system.

ranked the highest. Results can be seen in Table 6.5. The proposed model achieves state-of-the-art results with 77.8% accuracy for both KB subsets. The accuracy difference from the second best system using AttentiveCNNs in a pipeline approach is statistically significant ($p < 0.001$, test of equal proportions).

The second setting follows a reranking approach using the entity linking results of Yin et al. (2016) with Freebase-2M. The full model was applied to queries involving the linked entities and scored without taking into consideration the original entity linking scores. Accuracy is compared to the hierarchical LSTM model (Yu et al., 2017) that focuses only on relation identification. The difference in accuracy is statistically significant ($p < 0.05$, test of equal proportions) further supporting the argument in favour of joint entity-relation mapping. Comparing with a fixed pre-extracted set of candidate entities shows the benefits of joint entity-relation mapping by controlling for the performance gain of more accurate pruning provided by our entity linking component. Overall, the proposed system uses a simpler text encoder without residual connections but can leverage important information associated with the entity in the question.

In Table 6.6, we show ablation tests of skipping the entity linking pretraining step and removing features from the attention network. Removing the pretraining by entity linking shows a severe drop in performance of almost 2%. A similar finding was reported in the joint entity-relation extraction experiments of Miwa and Bansal (2016), where pretraining the entity network

was shown to significantly increase performance. We observe a similar trend as in Chapter 5, where first training the two sources independently and then jointly performs better. We also observe the impact of providing information about the correct alignment of entities and relations between the two sources. The pretrained attention network is biased towards this decomposition and performs better. Even though the information we use in pretraining the attention network is the same set of annotations in the training set, the model trained without that bias never converges to the same solution. In the feature ablation experiments, we see that n-gram embeddings in the attention network are sufficient for a high accuracy of 76.9%, but adding word and dependency embeddings improves that score to 77.4% and finally 77.8%.

6.7 Summary and Conclusion

We presented a system for end-to-end large-scale KBQA that jointly learns to identify and map entity and relation mentions from a question to KB subgraphs. The approach relies on a gating attention mechanism that has multiple functionality. It first identifies the entity mention enabling comparison with entity names in the KB with latent character n-gram features. This mechanism is used to prune the search space of possible queries by creating a candidate list of entities. The attention output is then used as an input along with question tokens represented on word and character level to be encoded by a bidirectional LSTM to a sequence of latent representations. The resulting LSTM representations are then merged according to the same attention into fixed length entity and relation representations, which are compared with their respective parts in the KB subgraphs. Parameters of the network are pretrained on auxiliary tasks of Knowledge Base Completion and entity linking. Evaluation on the SimpleQuestions dataset shows improved performance compared to models that perform the subtasks of entity linking and relation classification independently and models that do not consider structure. In addition, the name similarity component used for entity candidate generation performs better in entity linking than entity linking systems proposed in previous work. We conclude that providing information about the decomposition of the question into entity and relation and then matching the resulting structures provides a significant advantage in the performance of the system. For future work, it would be interesting to extend the joint entity-relation identification approach to complex question settings, where multiple entities and relations need to be identified in the same question.

CHAPTER 7

Conclusion

In this thesis, we investigated methods to utilize structured information for learning latent representations that capture useful properties of language and general knowledge expressed in text and Knowledge Bases. The representations were used in systems performing different tasks that require aspects of natural language understanding. The way we used structure and the impact it had on each of the different settings is the following:

In chapter 3, we investigated methods of embedding word and dependency features from parsed sentences into a low dimensional space. Then we investigated methods to make better use of such information in order to provide structural information to systems for short-text categorization. Given evidence from past research that syntactic information is useful to capture semantic properties of text required in text categorization, we developed and evaluated a general method to provide such information into classification systems only through shareable vector representations. We experimentally established that this is possible by applying the method with three different classification methods to three short-text categorization tasks and showed that it can increase their performance compared to a non-syntactic and a syntactic but less structured approach.

In chapter 4, we looked at the application of latent representations of words and syntactic features to Word Sense Induction and Disambiguation. Contrary to previously proposed models in the literature, we made use of pretrained feature embeddings and showed that they can be used in a probabilistic clustering framework reducing the total number of variables compared to sparse discrete features. The pretrained feature vectors can provide to the model diverse information from multiple views of context: syntactic, phrasal and topical. The rich information from the

pretrained feature representations and the different context types resulted in models that correlate better with human judgements for discovering and distinguishing word senses.

In chapter 5, we looked at Knowledge Base Completion with linked text corpora through usage of latent feature models. We presented a framework where for each fact expressed as a typed edge between two entity nodes, we add edges from text relations and the KB to construct a subgraph. By applying a type system on the edges, the information of the subgraph can be aggregated and encoded into a fixed length n-tuple and embedded into a lower dimension space to be scored by a Multilayer Perceptron. The richer information in the subgraph helps the model utilize the textual information to infer new facts compared to approaches where text is treated as an additional collection of triples. In addition, we showed that initializing parameters of the model from simpler versions without text and using pretrained feature embeddings substantially increased its performance.

In chapter 6, we focused on the problem of Question Answering on Knowledge Bases. We developed a system that jointly learns to decompose the questions into entity and relation mentions and score them by matching substructures of a Knowledge Base subgraph corresponding to the query. The system makes extensive usage of pretrained representations and combines multiple views of the data. The Knowledge Base symbols are pretrained by performing Knowledge Base Completion, text features of words, syntactic dependencies and character n-grams are pretrained with variations of the skip-gram model, and the network responsible for detecting entities is pre-trained in entity linking which helps bias the network towards making the right entity-relation decomposition. The system compares favourably against systems that ignore structural correspondences between text and the Knowledge Base or address subtasks with a pipeline and reranking approach.

One of the aims of this thesis was to push the state-of-the-art further for the tasks considered. This aim was largely accomplished. The word embeddings from chapter 3 were shown to be superior for sentence classification than typically used alternatives, and incorporating the dependency features as a means to provide syntactic information for short-text classification reached the performance of the best systems for question and relation classification using generic off-the-shelf systems. The probabilistic WSI model achieved state-of-the-art performance in two benchmarks and and the models that jointly embed text and KBs applied in KBC and KBQA were shown to outperform other approaches that may use more complicated component networks but ignore a lot of the structural information about how the two sources can be aligned.

Another aim of the thesis was to show the impact of utilizing structure and types based on domain knowledge for representation learning. We can conclude that the linguistic structure expressed in typed dependency graphs and encoded in the word and dependency feature embeddings had a positive impact on performance across all the tasks considered. We were able to exploit the benefit of unsupervised pretraining and also combine them with other structural information that is typically used for text encoding models. In addition,unsupervised pretraining

was successfully used for KB and n-gram representations.

The final aim of this research was to show that we can jointly learn the semantics of text and KBs by considering simple structural alignments in the entity-relation level. This decomposition allowed us to simplify the problem as there were fewer semantic correspondences to be matched and also utilize different feature representations for each aspect. We also observed that learning first a representation of each source individually and then their semantic equivalence was beneficial in both tasks of KB-text interaction. This provides evidence that the latent spaces can be transformed in whole to be embedded together in a new space given only a few sample points where we know the semantic correspondence between them. After the transformation, much of the properties of the original spaces are preserved providing better generalization for mapping areas where we do not have direct supervision of how the mapping should be done.

Given the above experimental evidence we can answer the two questions we made in the beginning: 1) We can learn latent feature representations of dependency graph features that can be used as a means to provide syntactic information to a text encoding model and 2) providing alignment information of entities and relations between text and KB improve the performance of systems that need to learn semantic equivalence between the two sources.

There are many research opportunities for future work in utilizing structure in representation learning. Incorporating structural information, either known or automatically learned, can potentially benefit learning representations of other types of unstructured data besides text, such as images and raw signals from devices. Taking advantage of structured data contained in databases can assist this process and result in machines that can process information in more intelligent ways.

Appendix A: list of stopwords used in candidate entity generation

the
of
in
a
is
,
’s
what
:
(
)
?
and
for
to
from
on
.
who
by
was
an
that
with
where
at
&
which
/
are
-
’
as
“

does
did
”
have
how
about
this
can
no
under
out
has
off
n’t
\
after

Appendix B: Map of Stanford Dependencies to Universal Dependencies

acomp = xcomp
advcl = advcl
advmod = advmod
agent = nmod:agent
amod = amod
appos = appos
aux = aux
auxpass = auxpass
cc = cc
ccomp = ccomp
conj = conj:and
cop = cop
csubj = csubj
csubjpass = csubjpass
dep = dep
discourse = discourse
dobj = dobj
expl = expl
goeswith = goeswith
iobj = iobj
mark = mark
mwe = mwe
neg = neg
nn = compound
npadvmod = nmod:npmod
nsubj = nsubj
nsubjpass = nsubjpass
num = nummod
number = compound
parataxis = parataxis
poss = nmod:poss
possessive = case
pobj = nmod
preconj = conj:preconj

predet = det:predet
prt = compound:prt
punct = punct
quantmod = advmod
rcmod = acl:relcl
ref = ref
root = root
tmod = nmod:tmod
xcomp = xcomp
det = det
vmod = acl if head is Noun else advcl
pcomp = acl if prep head is Noun else advcl

List of Abbreviations

AI - Arfificial Intelligence
BoW - Bag of Words
BUB - Best Upper Bound
CBoW - Continuous Bag of Words
CNN - Convolutional Neural Network
DCF - Dependency Context Feature
DL - Deep Learning
EM - Expectation Maximization
EXT - Extended Dependency Skip-gram
FB - Freebase
FRN - Feature Rich Network
GD - Gradient Descent
GMM - Gaussian Mixture Model
ICL - Integrated Complete Likelihood
KB - Knowledge Base
KBC - Knowledge Base Completion
KBQA - Knowledge Base Question Answering
LDA - Latent Dirichlet Allocation
LSTM - Long Short Term Memory
MCC - Multi-Context-Continuous
ML - Machine Learning
MLE - Maximum Likelihood Estimation
MLP - Multilayer Perceptron
MRR - Mean Reciprocal Rank
NBoW - Neural Bag of Words
NLP - Natural Language Processing
NLU - Natural Language Understanding
NMF - Nonnegative Matrix Factorization
NMI - Normalized Mutual Information
NN - Neural Network
QA - Question Answering
QC - Question Classification
RC - Relation Classification
RL - Representation Learning
RNN - Recurrent Neural Network
SDP - Shortest Dependency Path

SGD - Stochastic Gradient Descent
SVD - Singular Value Decomposition
SVM - Support Vector Machine
UD - Universal Dependencies
US - Universal Schema
WCF - Word Context Feature
WSD - Word Sense Disambiguation
WSI - Word Sense Induction

Bibliography

L. Mou, H. Peng, G. Li, Y. Xu, L. Zhang, and Z. Jin, "Discriminative neural sentence modeling by tree-based convolution," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, September 2015, pp. 2315–2325. [Online]. Available: http://aclweb.org/anthology/D15-1279

K. Xu, Y. Feng, S. Huang, and D. Zhao, "Semantic relation classification via convolutional neural networks with simple negative sampling," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, September 2015, pp. 536–540.

J. Wang, M. Bansal, K. Gimpel, B. D. Ziebart, and T. Y. Clement, "A sense-topic model for word sense induction with unsupervised data enrichment," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 59–71, 2015.

K. Toutanova, D. Chen, P. Pantel, H. Poon, P. Choudhury, and M. Gamon, "Representing text for joint embedding of text and knowledge bases." in *Proceedings of the 2015 conference on Empirical Methods for Natural Language Processing (EMNLP)*, vol. 15, 2015, pp. 1499–1509.

Z. Dai, L. Li, and W. Xu, "Cfo: Conditional focused neural question answering with large-scale knowledge bases," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, "Simple question answering by attentive convolutional neural network," in *Proceedings of COLING*, 2016.

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. Bleecher Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. de Brébisson,

O. Breuleux, P.-L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.-A. Côté, M. Côté, A. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. Ebrahimi Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.-P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. Léonard, Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.-A. Manzagol, O. Mastropietro, R. T. McGibbon, R. Memisevic, B. van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I. V. Serban, D. Serdyuk, S. Shabanian, E. Simon, S. Spieckermann, S. R. Subramanyam, J. Sygnowski, J. Tanguay, G. van Tulder, J. Turian, S. Urban, P. Vincent, F. Visin, H. de Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, 2016.

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

X. Zhang and Y. LeCun, "Text understanding from scratch," *arXiv preprint arXiv:1502.01710*, 2015.

K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *The Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.

M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

P. D. Turney, P. Pantel *et al.*, "From frequency to meaning: Vector space models of semantics," *Journal of artificial intelligence research*, vol. 37, no. 1, pp. 141–188, 2010.

O. Levy, Y. Goldberg, and I. Dagan, "Improving distributional similarity with lessons learned from word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 3, pp. 211–225, 2015.

Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

R. Lebret and R. Collobert, "Word emdeddings through hellinger pca," *arXiv preprint arXiv:1312.5542*, 2013.

D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

T. Van de Cruys, "Mining for meaning. the extraction of lexico-semantic knowledge from text," *Groningen Dissertations in Linguistics*, vol. 82, 2010.

A. Gittens, D. Achlioptas, and M. W. Mahoney, "Skip-gram-zipf+ uniform= vector additivity," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 69–76.

O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization," in *Advances in neural information processing systems*, 2014, pp. 2177–2185.

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *The Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *In Proceedings of the International Conference on Machine Learning*, 2012.

R. Socher, A. Perelygin, J. Wu, J. Chuang, D. C. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2013, pp. 1631–1642.

X. Li and D. Roth, "Learning question classifiers," in *Proceedings of COLING*, 2002, pp. 1–7.

I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proceedings of the 2008 Workshop on Semantic Eval-*

uations, 2009, pp. 94–99.

M.-C. De Marneffe, T. Dozat, N. Silveira, K. Haverinen, F. Ginter, J. Nivre, and C. D. Manning, "Universal stanford dependencies: A cross-linguistic typology," in *Proceedings of LREC*, 2014, pp. 4585–4592.

M.-C. De Marneffe and C. D. Manning, "The stanford typed dependencies representation," in *Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation*, 2008, pp. 1–8.

D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, vol. 1, 2014, pp. 740–750.

Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1746–1751.

J. Turian, L. Ratinov, and Y. Bengio, "Word representations: a simple and general method for semi-supervised learning," in *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, 2010, pp. 384–394.

O. Levy and Y. Goldberg, "Dependencybased word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, vol. 2, 2014, pp. 302–308.

W. Ling, C. Dyer, A. Black, and I. Trancoso, "Two/too simple adaptations of word2vec for syntax problems," 2015.

S. K. Tai, R. Socher, and D. C. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 1556–1566.

M. Ma, L. Huang, B. Xiang, and B. Zhou, "Dependency-based convolutional neural networks for sentence embedding," p. 174, 2015.

J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an, "Graph convolutional encoders for syntax-aware neural machine translation," *arXiv preprint arXiv:1704.04675*, 2017.

M. Faruqui, Y. Tsvetkov, P. Rastogi, and C. Dyer, "Problems with evaluation of word embeddings using word similarity tasks," *arXiv preprint arXiv:1605.02276*, 2016.

D. Lin, "Automatic retrieval and clustering of similar words," in *Proceedings of COLING*, 1998, pp. 768–774.

G. Grefenstette, *Explorations in automatic thesaurus discovery*. Springer Science & Business Media, 2012, vol. 278.

S. Padó and M. Lapata, "Dependency-based construction of semantic space models," *Computational Linguistics*, vol. 33, no. 2, pp. 161–199, 2007.

M. Baroni and A. Lenci, "Distributional memory: A general framework for corpus-based semantics," *Computational Linguistics*, vol. 36, no. 4, pp. 673–721, 2010.

T. N. Pham, G. Kruszewski, A. Lazaridou, and M. Baroni, "Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 971–981.

O. Melamud, D. McClosky, S. Patwardhan, and M. Bansal, "The role of context types and dimensionality in learning word embeddings," *arXiv preprint arXiv:1601.00893*, 2016.

K. Hashimoto, P. Stenetorp, M. Miwa, and Y. Tsuruoka, "Jointly learning word representations and composition functions using predicate-argument structures," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1544–1555.

G. Stanovsky, I. Dagan *et al.*, "Open ie as an intermediate structure for semantic tasks," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, vol. 2, 2015, pp. 303–308.

J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

D. Haussler, "Convolution kernels on discrete structures," Technical report, Department of Computer Science, University of California at Santa Cruz, Tech. Rep., 1999.

D. Zelenko, C. Aone, and A. Richardella, "Kernel methods for relation extraction," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1083–1106, 2003.

A. Moschitti, "Efficient convolution kernels for dependency and constituent syntactic trees," in *Proceedings of ECML*, vol. 4212, 2006, pp. 318–329.

A. Moschitti, S. Quarteroni, R. Basili, and S. Manandhar, "Exploiting syntactic and shallow semantic kernels for question answer classification," in *Proceeings of the 2007 Annual Meeting of the Association for Computational Linguistics*, vol. 45, no. 1, 2007, p. 776.

R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing (EMNLP) and Computational Natural Language Learning (CoNLL)*, 2012, pp. 1201–1211.

R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Advances in Neural Information Processing Systems*, 2011, pp. 801–809.

J. Cheng and D. Kartsaklis, "Syntax-aware multi-sense word embeddings for deep compositional models of meaning," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, September 2015, pp. 1531–1542.

M. Roth and M. Lapata, "Neural semantic role labeling with dependency path embeddings," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), 2016.

Y. Xu, L. Mou, G. Li, Y. Chen, H. Peng, and Z. Jin, "Classifying relations via long short term memory networks along shortest dependency paths." in *Proceedings of the 2015 conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2015, pp. 1785–1794.

N. Peng, H. Poon, C. Quirk, K. Toutanova, and W.-t. Yih, "Cross-sentence n-ary relation extraction with graph lstms," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 101–115, 2017.

M. Iyyer, V. Manjunatha, J. Boyd-Graber, and H. Daumé III, "Deep unordered composition rivals syntactic methods for text classification," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, vol. 1, 2015, pp. 1681–1691.

J. Li, T. Luong, D. Jurafsky, and E. Hovy, "When are tree structures necessary for deep learning of representations?" in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, September 2015, pp. 2304–2314.

L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 406–414.

F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, 2015.

J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

S. Eger, E.-L. D. Dinh, I. Kuznetsov, M. Kiaeeha, and I. Gurevych, "Eelection at semeval-2017 task 10: Ensemble of neural learners for keyphrase classification," *arXiv preprint arXiv:1704.02215*, 2017.

N. Reimers and I. Gurevych, "Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging," *arXiv preprint arXiv:1707.09861*, 2017.

——, "Optimal hyperparameters for deep lstm-networks for sequence labeling tasks," *arXiv preprint arXiv:1707.06799*, 2017.

I. P. Klapaftis and S. Manandhar, "Evaluating word sense induction and disambiguation methods," *Language resources and evaluation*, vol. 47, no. 3, pp. 579–605, 2013.

R. Navigli, "Word sense disambiguation: A survey," *ACM Computing Surveys (CSUR)*, vol. 41, no. 2, p. 10, 2009.

S. Brody and M. Lapata, "Bayesian word sense induction," in *Proceedings of the 2009 Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2009, pp. 103–111.

J. H. Lau, P. Cook, D. McCarthy, D. Newman, and T. Baldwin, "Word sense induction for novel sense detection," in *Proceedings of the 2012 Conference of the European Chapter of the Asso-*

ciation for Computational Linguistics (EACL), 2012, pp. 591–601.

H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 3, no. 1, p. 1, 2009.

S. Manandhar, I. P. Klapaftis, D. Dligach, and S. S. Pradhan, "Semeval-2010 task 14: Word sense induction & disambiguation," in *Proceedings of the 2010 international workshop on semantic evaluation*, 2010, pp. 63–68.

D. Jurgens and I. P. Klapaftis, "Semeval-2013 task 13: Word sense induction for graded and non-graded senses." in *Proceedings of the 2013 international workshop on semantic evaluation*, 2013, pp. 290–299.

E. Agirre and A. Soroa, "Semeval-2007 task 02: Evaluating word sense induction and discrimination systems," in *Proceedings of the 2007 International Workshop on Semantic Evaluations*, 2007, pp. 7–12.

H. Schütze, "Automatic word sense discrimination," *Computational linguistics*, vol. 24, no. 1, pp. 97–123, 1998.

P. Pantel and D. Lin, "Discovering word senses from text," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 613–619.

A. Purandare and T. Pedersen, "Word sense discrimination by clustering contexts in vector and similarity spaces," in *Proceedings of the 2004 Conference on Computational Natural Language Learning (CoNLL)*, 2004.

T. Van de Cruys and M. Apidianaki, "Latent semantic word sense induction and disambiguation," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011, pp. 1476–1485.

K. Goyal and E. H. Hovy, "Unsupervised word sense induction using distributional statistics." in *Proceedings of COLING*, 2014, pp. 1302–1310.

A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

J. Véronis, "Hyperlex: lexical cartography for information retrieval," *Computer Speech & Language*, vol. 18, no. 3, pp. 223–252, 2004.

E. Agirre, D. Martínez, O. L. de Lacalle, and A. Soroa, "Two graph-based algorithms for state-of-the-art wsd," in *Proceedings of the 2006 Conference on Empirical Methods for Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2006, pp. 585–593.

L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Tech. Rep., 1999.

I. P. Klapaftis and S. Manandhar, "Word sense induction using graphs of collocations." in *Proceedings of ECAI*, 2008, pp. 298–302.

C. Biemann, "Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems," in *Proceedings of the first workshop on graph based methods for natural language processing*, 2006, pp. 73–80.

I. P. Klapaftis and S. Manandhar, "Word sense induction & disambiguation using hierarchical random graphs," in *Proceedings of the 2010 conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2010, pp. 745–755.

D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the american statistical association*, 2012.

B. Chang, W. Pei, and M. Chen, "Inducing word sense with automatically learned hidden concepts." in *Proceedings of COLING*, 2014, pp. 355–364.

A. Neelakantan, J. Shankar, A. Passos, and A. McCallum, "Efficient non-parametric estimation of multiple embeddings per word in vector space," *arXiv preprint arXiv:1504.06654*, 2015.

J. Li and D. Jurafsky, "Do multi-sense embeddings improve natural language understanding?" 2015.

S. Bartunov, D. Kondrashkin, A. Osokin, and D. Vetrov, "Breaking sticks and ambiguities with adaptive skip-gram," in *Artificial Intelligence and Statistics*, 2016, pp. 130–138.

A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

R. Salakhutdinov, S. T. Roweis, and Z. Ghahramani, "Optimization with em and expectation-conjugate-gradient," in *Proceedings of the 2003 International Conference on Machine Learning (ICML)*, 2003, pp. 672–679.

S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 721–741, 1984.

J. G. Dias and M. Wedel, "An empirical comparison of em, sem and mcmc performance for problematic gaussian mixture likelihoods," *Statistics and Computing*, vol. 14, no. 4, pp. 323–332, 2004.

D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, 2007, pp. 1027–1035.

J. Blömer and K. Bujna, "Simple methods for initializing the em algorithm for gaussian mixture models," *CoRR*, 2013.

C. Biernacki, G. Celeux, and G. Govaert, "Assessing a mixture model for clustering with the integrated completed likelihood," *IEEE transactions on pattern analysis and machine intelligence*, vol. 22, no. 7, pp. 719–725, 2000.

G. Schwarz, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp.

461–464, 1978.

G. McLachlan and D. Peel, *Finite mixture models*. John Wiley & Sons, 2004.

L. Li, I. Titov, and C. Sporleder, "Improved estimation of entropy for evaluation of word sense induction," *Computational Linguistics*, vol. 40, no. 3, pp. 671–685, 2014.

L. Paninski, "Estimation of entropy and mutual information," *Neural computation*, vol. 15, no. 6, pp. 1191–1253, 2003.

I. Korkontzelos and S. Manandhar, "Uoy: Graphs of unambiguous vertices for word sense induction and disambiguation," in *Proceedings of the 2010 international workshop on semantic evaluation*, 2010, pp. 355–358.

A. Ferraresi, E. Zanchetta, M. Baroni, and S. Bernardini, "Introducing and evaluating ukwac, a very large web-derived corpus of english," in *Proceedings of the 4th Web as Corpus Workshop*, 2008, pp. 47–54.

O. Baskaya, E. Sert, V. Cirik, and D. Yuret, "Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation," 2013, pp. 300–306.

J. H. Lau, P. Cook, and T. Baldwin, "unimelb: Topic modelling-based word sense induction for web snippet clustering," in *Proceedings of the 2013 International Workshop on Semantic Evaluation*, 2013, pp. 217–221.

L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

C. Hennig, "Methods for merging gaussian mixture components," *Advances in data analysis and classification*, vol. 4, no. 1, pp. 3–34, 2010.

J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs." in *Proceedings of the 2013 conference on Empirical Methods for Natural Language Processing (EMNLP)*, vol. 2, no. 5, 2013, p. 6.

S. Reddy, M. Lapata, and M. Steedman, "Large-scale semantic parsing without question-answer pairs," *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 377–392, 2014.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics*, 2009, pp. 1003–1011.

X. Ling and D. S. Weld, "Fine-grained entity recognition." in *Proceedings of AAAI*, 2012.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *The semantic web*. Springer, 2007, pp. 722–735.

A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*,

2013, pp. 2787–2795.

M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 2011 international conference on machine learning (ICML)*, 2011, pp. 809–816.

R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in neural information processing systems*, 2013, pp. 926–934.

M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2016.

S. Riedel, L. Yao, A. McCallum, and B. M. Marlin, "Relation extraction with matrix factorization and universal schemas," 2013.

L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A: statistical mechanics and its applications*, vol. 390, no. 6, pp. 1150–1170, 2011.

M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1, pp. 107–136, 2006.

N. Lao, T. Mitchell, and W. W. Cohen, "Random walk inference and learning in a large scale knowledge base," in *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2011, pp. 529–539.

M. Gardner and T. M. Mitchell, "Efficient and expressive knowledge base completion using subgraph feature extraction." in *Proceedings of the 2015 conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2015, pp. 1488–1498.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling, "Never-ending learning," in *Proceedings of AAAI*, 2015.

K. Toutanova and D. Chen, "Observed versus latent features for knowledge base and text inference," in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, 2015, pp. 57–66.

B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.

R. Kadlec, O. Bajgar, and J. Kleindienst, "Knowledge base completion: Baselines strike back," *arXiv preprint arXiv:1705.10744*, 2017.

A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "A semantic matching energy function for learning with multi-relational data," *Machine Learning*, vol. 94, no. 2, pp. 233–259, 2014.

A. Bordes, J. Weston, R. Collobert, Y. Bengio *et al.*, "Learning structured embeddings of knowledge bases." in *Proceedings of AAAI*, vol. 6, no. 1, 2011, p. 6.

Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes." in *Proceedings of AAAI*, 2014, pp. 1112–1119.

Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion." in *Proceedings of AAAI*, 2015, pp. 2181–2187.

G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix." in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015, pp. 687–696.

D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson, "Stranse: a novel embedding model of entities and relationships in knowledge bases," *arXiv preprint arXiv:1606.08140*, 2016.

X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 601–610.

D. Q. Nguyen, K. Sirts, L. Qu, and M. Johnson, "Neighborhood mixture model for knowledge base completion," *arXiv preprint arXiv:1606.06461*, 2016.

A. Neelakantan, B. Roth, and A. McCallum, "Knowledge base completion using compositional vector space models," in *Workshop on Automated Knowledge Base Construction (AKBC) at NIPS*, 2014.

Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, "Modeling relation paths for representation learning of knowledge bases," *arXiv preprint arXiv:1506.00379*, 2015.

T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," *arXiv preprint arXiv:1707.01476*, 2017.

R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions." in *Proceedings of AAAI*, 2016, pp. 2659–2665.

Z. Wang and J.-Z. Li, "Text-enhanced representation learning for knowledge graph." in *Proceedings of IJCAI*, 2016, pp. 1293–1299.

H. Zhong, J. Zhang, Z. Wang, H. Wan, and Z. Chen, "Aligning knowledge and text embeddings by entity descriptions." in *Proceedings of the 2015 conferece on Empirical Methods in Natural Language Processing (EMNLP)*, 2015, pp. 267–272.

S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," *Machine learning and knowledge discovery in databases*, pp. 148–163, 2010.

R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 2011, pp. 541–550.

M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning, "Multi-instance multi-label learning for relation extraction," in *Proceedings of the 2012 joint conference on empirical methods in natural language processing (EMNLP) and computational natural language learning (CoNLL)*, 2012, pp. 455–465.

F. Petroni, L. d. Corro, and R. Gemulla, "Core: Context-aware open relation extraction with factorization machines," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2015.

S. Rendle, "Factorization machines," in *Proceedings of the 2010 IEEE International Conference on Data Mining (ICDM)*, 2010, pp. 995–1000.

V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 2010 international conference on machine learning (ICML)*, 2010, pp. 807–814.

E. Gabrilovich, M. Ringgaard, and A. Subramanya, "Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0)," *http://lemurproject. org/clueweb09/FACC1*, vol. 5, 2013.

A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," *arXiv preprint arXiv:1506.02075*, 2015.

M. Yu, W. Yin, K. S. Hasan, C. d. Santos, B. Xiang, and B. Zhou, "Improved neural relation detection for knowledge base question answering," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.

D. Roth and W.-t. Yih, "Global inference for entity and relation identification via a linear programming formulation," *Introduction to statistical relational learning*, pp. 553–580, 2007.

Q. Li and H. Ji, "Incremental joint extraction of entity mentions and relations." in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 402–412.

A. Joulin, E. Grave, P. Bojanowski, M. Nickel, and T. Mikolov, "Fast linear model for knowledge graph embeddings," *arXiv preprint arXiv:1710.10881*, 2017.

D. Golub and X. He, "Character-level question answering with attention," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

M. Miwa and M. Bansal, "End-to-end relation extraction using lstms on sequences and tree structures," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016.

S. Zheng, F. Wang, H. Bao, Y. Hao, P. Zhou, and B. Xu, "Joint extraction of entities and relations based on a novel tagging scheme," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017.