Universitat Politècnica de Catalunya

PhD Dissertation

Low-Rank Regularization for High-Dimensional Sparse Conjunctive Feature Spaces in Information Extraction

Audi Primadhanty

Supervisors

Xavier Carreras
Ariadna Quattoni

Tutor

Horacio Rodríguez Hontoria

Computer Science Department
Ph.D. Program in Artificial Intelligence
Universitat Politècnica de Catalunya

September 2017

Abstract

One of the challenges in Natural Language Processing (NLP) is the unstructured nature of texts, in which useful information is not easily identifiable. Information Extraction (IE) aims to alleviate it by enabling automatic extraction of structured information from such text sources. The resulting structured information will facilitate easier querying, organizing, and analyzing of data from texts.

In this thesis, we are interested in two IE related tasks: (i) named entity classification and (ii) template filling. Specifically, this thesis examines the problem of learning classifiers of text spans and explore its application for extracting named entities and template slot-fillers.

In general, our goal is to construct a method to learn classifiers that: (i) require less supervision, (ii) work well with high-dimensional sparse feature spaces and (iii) are able to classify unseen items (i.e. named entities/slot-fillers not observed in training data).

The key idea of our contribution is the utilization of unseen conjunctive features. A conjunctive feature is a combination of features from different feature sets. For example, to classify a phrase, one might have one feature set for the context and another set for the phrase itself. When learning a classifier, only a factor of these conjunctive features will be observed in the training set, leaving the rest (i.e. unseen features) unusable for predicting items in test time. We hypothesize that utilizing such unseen conjunctions is useful to address all of the aspects of the goal.

We develop a general regularization framework specifically designed for sparse conjunctive feature spaces. Our strategy is based on employing tensors to represent the conjunctive feature space, and forcing the model to induce low-dimensional embeddings of the feature vectors via low-rank regularization on the tensor parameters. Such compressed representation will help prediction by generalizing to novel examples where most of the conjunctions will be unseen in the training set.

We conduct experiments on learning named entity classifiers and template filling, focusing on extracting unseen items. We show that when learning classifiers under minimal supervision, our approach is more effective in controlling model capacity than standard techniques for linear classification.

Keywords
low-rank regularization, minimal supervision, named entity classification, slot-filling

Acknowledgement

I would like to thank my advisors, Xavier Carreras and Ariadna Quattoni, for the continuous guidance, for the support in every step of the way, for the motivation, patience and inspiration. Thank you for believing in me.

I would also like to thank my tutor, Horacio Rodriguez for the invaluable advices and support.

Thanks to my office mates, in Barcelona and Grenoble, for the fun times and fruitful discussions. Special thanks to Pranava Swaroop for the collaboration and for sharing your insights, knowledge and wisdom.

I thank my family–ibu, bapak, ade, ebin–for their endless support, understanding and unconditional love.

Finally, to everyone not mentioned here, *gracias*!

Contents

List of Figures

List of Tables

1 Introduction

This thesis investigates the use of linear models with a low-rank constraint, as well as their application in the tasks of Information Extraction (IE). Specifically, we focus on addressing the challenge of feature sparsity in Named Entity Classification (NEC) and template filling.

We will describe the tasks in Section 1.1, in which we will also motivate our research. We will then describe the specific contributions of this thesis in Section 1.2. Finally, Section 1.4 will provide a guide to navigating this thesis.

1.1 Motivation and Goal

Throughout the history of human civilization, the use of written text has been key in passing knowledge from one generation to another. With the advancement of technology, it is getting more and more easy to produce text and share knowledge to each other. The world wide web alone has allowed the existence of an innumerable amount of pages of texts to be available to almost everyone in the world. Not to mention efforts in automating the digitization of printed materials, the transcription of audio recordings, and more recently, the captioning of images and videos.

In a modern era like today, one can find themselves drowned in a sea of literature in search of particular information of their interest. Since it is nearly impossible for anyone to read all existing texts, the ambition is to have machines do it for us, synthesize all the information and present them to us by request.

It is this very idea that fuels research efforts in Natural Language Processing (NLP), which aims to allow machines to analyze human languages for different purposes. Such ability is useful for many applications, from assisting people in their day-to-day activities (e.g. voice-driven assistants, natural-language search) to assisting corporations in their business ventures (e.g. sentiment analysis for automated trading, social media analytics, customer care).

One of the primary challenges in analyzing texts is its nature of being mainly unstructured or semi-structured, in which useful information are not easily identifiable. A subfield of NLP, called Information Extraction (IE), aims to enable automatic extraction of structured information from such text sources.

INVESTIGATIONS UNDER WAY THROUGHOUT THE COUNTRY HAVE BEEN FRUITLESS REGARDING THE ALLEGED PRESENCE OF SPANISH, BRITISH, ISRAELI, OR ITALIAN TERRORISTS IN COLOMBIA, FOLLOWING THE FOILED ATTACK ON SEVEN GENERALS, EIGHT COLONELS, AND OTHER TOP COMMANDERS OF THE ARMY AND NATIONAL POLICE IN MEDELLIN.

MEANWHILE, GENERAL **MIGUEL ALFREDO MAZA MARQUEZ**, DIRECTOR OF THE ADMINISTRATIVE DEPARTMENT OF SECURITY, IDENTIFIED **EUGENIO EXENDESTE**, AN IDEOLOGIST OF THE BASQUE FATHERLAND AND LIBERTY [ETA] SEPARATIST ORGANIZATION, AS THE LINK BETWEEN THE MEDELLIN CARTEL AND THE SPANISH TERRORISTS.

POLICE OPERATIONS DIRECTOR GENERAL **OCTAVIO VARGAS SILVA** SAID THAT ALL SECURITY ORGANIZATIONS ARE ON MAXIMUM ALERT. HE NOTED THAT THE SECRET ORGANIZATIONS HAVE COMPOSITE SKETCHES OF THE SUSPECTED ETA MEMBERS AND THE TWO COLOMBIANS WHO COORDINATED ALL THE STEPS TO CARRY OUT THE ATTACK THAT WAS INTENDED TO KILL MANY CIVILIANS AS A RESULT OF A ROCKET ATTACK ON A GASOLINE TRUCK THAT CARRIED 3,000 GALLONS OF FLIGHT FUEL.

Fig. 1.1.: An example article (DEV-MUC3-1229) from MUC-4 (Sundheim, 1992) corpus annotated (in bold, purple colored, underlined) with PERSON named entities.

1.1.1 Information Extraction

IE is concerned with identifying predefined types of information from text (Riloff, 1999). It represents documents as sets of entities and frames that are another way of formally describing the relationships between the entities (Feldman and Sanger, 2007).

For example, consider a text article such as in Figure 1.1, an important step in understanding the article is to identify information relevant to the news story; what type of event is described, where does it take place, who is involved, what are their relations, etc. It is the goal of IE to identify such information from text and organize it in a manner that benefits people or other systems. In this thesis, we are interested in identifying two types of information; (i) Named Entities (NE) and (ii) template slot-fillers.

Named Entities Named Entities (NE) are objects that can be referred to with a proper name, such as a person, a location, an organization or a disease. For example, in Figure 1.1, we identify some proper names of individuals and annotate the text with such information. The annotated text contains the original news story, the information of the people involved in it and the location of their mentions in the text. Such is the expected result of a Named Entity Recognition and Classification (NERC) system. The types of entity extracted by a system are application-specific, depending on the domain of interest. Such information can be useful for further processing, such as linking the entities to a knowledge base entry, extracting relation between the entities or as an additional information for other NLP tasks.

Weapon	ROCKET
Perpetrator (individual)	TERRORISTS
Perpetrator (organization)	BASQUE FATHERLAND AND LIBERTY [ETA] SEPARATIST ORGANIZATION; MEDELLIN CARTEL
Target	GASOLINE TRUCK

Fig. 1.2.: An example of ATTACK template for article *DEV-MUC3-1229* from MUC-4 corpus.

Template slot-fillers A template is typically related to a description of an event, and the slots are the predefined types of information related to the event. For example, one might have a template describing an event of airline ticket booking with slots for the city of departure, the city of arrival, and the date of the intended trip. Another example could be a template describing an attack, with slots for the weapons, the perpetrators and the targets. For instance, from the text in Figure 1.1, a template filling system could be expected to extract information for an attack event such as shown in Figure 1.2. With this type of structured information, it could be easier to perform tasks such as text summarization or automated customer service.

1.1.2 Challenges in Information Extraction Systems

In general an IE system could use hand-crafted rules, machine-learning models or a combination of both. Although rules and heuristics might perform well, it could be costly, especially when dealing with large corpora with many entity/slot types. Moreover, it is also task specific; one should create a new set of rules for a new domain. In this thesis, we opt for making use of machine learning to model the problem. We identify the main areas of concern for our research below.

1.1.2.1. Extracting entities/role-fillers

In extracting named entities/slot-fillers, one must address two sub-tasks; (i) recognizing if a mention is an entity/slot-filler and (ii) classifying it as one of the predefined named entity/slot-filler types. The task of classifying could become more challenging when mentions are ambiguous, i.e. the same phrase could belong to different types in different contexts. In our case, we observe that entity/slot-filler candidates are mostly unambiguous[1]. Unambiguous mentions suggest one can design a system that memorizes the types of entities it sees during training and use them to predict mentions of the same entities during test time. Therefore, in this case, the important challenge is on extracting named entity/slot-filler candidates that are not observed during training (i.e. unseen candidates).

[1] Less than 3% of entity/slot-filler candidates in CoNLL 2003 named entity task corpus (Tjong Kim Sang and De Meulder, 2003) and MUC-4 template filling task corpus (Sundheim, 1992) are ambiguous.

1.1.2.2. Utilizing high-dimensional sparse parameter spaces

To train a statistical model, we need to represent our entity/slot-filler candidates in a parameter space that encodes linguistic information. In NLP, such parameter space is often high-dimensional and sparse. For example, a typical representation called "bag-of-words" represents a text span with the list of all words contained in the span. It would induce a parameter space that includes all words in the dictionary. With such feature spaces, the challenge is to control the capacity of the model to prevent overfitting of training data.

Conjunctive feature spaces To enrich the model, one might want to leverage conjunctions of different feature sets, such as combinations of features of the context and features of the candidate mentions. These sets of features can be grouped into vectors which we call **elementary feature vectors**, and their combinations are referred as **conjunctive features**. Ideally, we would like to train a classifier that can utilize all conjunctions of elementary features, since among them there might be some that are discriminative for the classification task at hand. Using conjunctive features will produce an even higher dimensional sparse feature space. Thus it is even more crucial to control the capacity of the model.

Unseen conjunctions Although conjunctive features can help models to learn better during training, such fine features might not influence prediction during test time, especially for unseen candidates. In models like linear classifiers, the score of a prediction is determined by weights associated to its features. A model learns such weights based on examples it observes during training. Conjunctive features that do not occur in the train set (i.e. unseen conjunctions), therefore, will not have non-zero weights, and thus will not affect predictions.

A standard approach to control the capacity of a model is to use ℓ_1 or ℓ_2 regularization on the parameter vector. However, this type of regularization does not seem to be effective when dealing with sparse conjunctive feature spaces. The main limitation is that ℓ_1 and ℓ_2 regularization can not let the model give weight to unseen conjunctions. Without such ability it is unlikely that the model will generalize to novel examples, where most of the conjunctions will be unseen in the training set. Therefore, figuring out how to utilize such unseen conjunctions is another important challenge to resolve.

Of course, one could impose a strong prior on the weight vector so that it assigns weight to unseen conjunctions, but how can we build such a prior? What kind of reasonable constraints can we put on unseen conjunctions?

Another common approach to handle high dimensional conjunctive feature spaces is to manually design the feature function so that it includes only a subset of "relevant" conjunctions. But designing such a feature function can be time consuming and one

might need to design a new feature function for each classification task. Ideally, we would have a learning algorithm that does not require such feature engineering and that it can automatically leverage rich conjunctive feature spaces.

1.1.2.3. Minimal Supervision

Machine learning, unfortunately, does not always remove the need for human effort in annotating corpora. Therefore, we seek to reduce the annotation requirement to a minimal level; requiring only a few examples of each entity/slot type. This is a typical scenario in an industrial setting for many NLP classification tasks. For instance, developers might be interested in classifying entities according to their own classification schema and can only provide a handful of examples of each class.

The challenge is then to design a system that is effective in such setting. Using only a few examples for each type means most mentions in the test set are unseen during training. Moreover, having less training data will require to train the model with an even more sparse feature space where less feature conjunctions are observed. Under such circumstances, it is imperative for the system to be able to extract unseen mentions and utilize unseen feature conjunctions effectively. Improvements on these two fronts will, in the end, improve the performance of models trained with minimal supervision and allow the construction of competitive models with less supervision.

1.1.3 Goal

In summary, the goal of this thesis is to investigate a method for: (i) extracting unseen items of predefined types, (ii) utilizing a high dimensional conjunctive feature space and (iii) minimizing required supervision. The practical goal is to apply such method on two important sub-tasks of IE; named entity classification and template filling.

1.2 Contributions

In this thesis, we attempt to address the goals mentioned above with two main contributions. First, we develop a regularization framework specifically designed for sparse conjunctive feature spaces. Second, we conduct experiments on learning entity classifiers and template filling, both with minimal supervision.

1.2.1 Development of Low-Rank Regularization Method for Sparse Conjunctive Feature Spaces

An important key feature of our method is enabling an effective utilization of conjunctive features, including those that are unseen during training. For the unseen

conjunctions to be useful, we need to be able to give weights to them. Such ability will enable more unseen mentions to be classified and thus allows for building competitive models with less supervision.

To do that, we construct a regularization scheme for training linear feature-based models in high-dimensional sparse conjunctive feature spaces. Our approach results in a more effective way of controlling model capacity, and it does not require feature engineering. More importantly, it allows unseen conjunctive features to be mapped closer and thus makes it possible to give weights to the unseen ones.

Our strategy is based on:

- Employing tensors to represent the parameter space in the scoring function of the model.

- Forcing the model to induce low-dimensional embeddings of elementary vectors via low-rank regularization on the tensor parameters.

Parameter tensor The standard approach to handling conjunctive feature spaces in NLP is to regard the parameters of the linear model as long vectors of weights for all possible conjunctions. In our models, the parameters are tensors that represent weights of the elementary features and their conjunctions. To perform computations on the tensor, we employ the technique of tensor matricization, which turns the tensor into a matrix and allows the computations to be performed on the resulting matrix instead of the original tensor form.

Low-rank constraint We show that the rank of the parameter matrix has a very natural interpretation. It can be seen as the intrinsic dimensionality of a latent embedding of the elementary feature vectors. Therefore, by imposing a low-rank penalty on the matrix parameters, we are encouraging the model to induce a low-dimensional projection of the elementary feature vectors. In a lower dimensional space, unseen feature conjunctions can be projected closer to seen conjunctions that are similar and thus weighted similarly.

Convex optimization Using the low-rank regularization constraint in the learning algorithm would result in a non-convex optimization. Instead, we follow a standard approach to use the nuclear norm as a convex relaxation of the rank. We also present a simple convex learning algorithm that alternates between gradient update and proximal projection of the parameters. In addition, we also describe implementation details of the algorithm that addresses with memory issues.

1.2.2 Experiments on Named Entity Classification and Template Filling

To evaluate the efficacy of our method, we conduct experiments on learning entity classifiers and extracting slot-fillers, both with minimal supervision.

Minimal supervision and unseen evaluation To minimize supervision, we define the required supervision as a list of phrase examples for each entity/slot-filler types. Considering that our candidates are mostly unambiguous, we then assume that all mentions of these phrases in the corpus always belong to the same type as specified in the list. This method allows us to use a handful of seed phrases to obtain many examples for training. Meanwhile, to measure the ability of the method in extracting unseen items, we adapt the evaluation procedure to focus on unseen entities/role-fillers.

Results Some focused experiments were done in the framework of named entity and slot-filler classification. The experiments show that nuclear norm regularization (ℓ_*) improves over ℓ_2 regularization. A full development of named entity and slot-filler extraction for comparison with state-of-the-art is left for future work.

1.3 Publications

Publications related to this thesis are as follows:

- Primadhanty et al. (2015)
  Low-rank regularization for sparse conjunctive feature spaces: An application to named entity classification.
  Audi Primadhanty, Xavier Carreras, Ariadna Quattoni. ACL 2015
  This publication contains the main contributions for Chapter 3 and Chapter 4.
- Ferrero et al. (2017)
  Intoevents: An interactive toolkit for discovering and building event schemas
  Germán Ferrero, Audi Primadhanty, Ariadna Quattoni. EACL 2017
  This publication is related to the task in Chapter 5.

We are also pursuing another publication related to Chapter 5.

1.4 Navigating The Thesis

The thesis is organized as follows: Chapter 2 provides some background and presents some related work. Our main contribution is described in Chapter 3, with its applications to named entity classification and template filling described in Chapter 4 and 5, respectively. Finally, in Chapter 6, we summarize our conclusions and discuss possible future works to improve different aspects of our contributions.

2 Related Work

In this chapter, we will present studies or methods related to aspects of this thesis, including low-rank learning, Named Entity (NE) classification, template filling and minimal supervision. We start in Section 2.1 by presenting the use of low-rank learning in different Natural Language Processing (NLP) applications. We then describe methods used in solving NE classification tasks in Section 2.2, and event role filler extraction in Section 2.3. In Section 2.4 we briefly describe the use of different supervision settings in NLP systems in general, focusing on works on IE with minimal supervision for training a supervised model. The literature presented here are by no means exhaustive; we will briefly describe some papers and highlight those relevant to aspects of this thesis.

2.1 Low-Rank Learning in NLP

Our objective in using low-rank constraint in this thesis is to (i) control the capacity of our model and (ii) to allow the model to give weight to conjunctions of features not observed at training by (iii) implicitly induces a low dimensional embedding of feature vectors. The use of low-rank constraints to achieve each of these objectives is not a novel concept in machine learning. We will show below how low-rank constraints have been used to achieve them in NLP.

In Section 2.1.1 we present some works whose objective is to generate a latent low dimensional embedding of words. In these works, low-rank tensor or matrix decomposition was used as an essential (if not the main) part of their methods. In Section 2.1.2, we present works that use low-rank to perform matrix or tensor completion on different NLP tasks. The utilization of low-rank in such is analogous to giving weights to unseen parameters in our method. Finally, in Section 2.1.3 we present methods that make use of low-rank to control the capacity of their model.

2.1.1 Low Rank Constraint for Generating Low-Dimensional Embedding

Numerous efforts in NLP have focused on generating low-dimensional word/term representations that can capture their semantic meanings. Examples of early efforts include Latent Semantic Analysis (LSA) (Landauer and Dumais, 1997), Probabilistic LSA (PLSA) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003). Results of some recent work on word embeddings, such as Word2vec (Mikolov et al., 2013a,b,c), GloVe (Pennington et al., 2014), SENNA (Collobert et al., 2011)

and the work of Lebret and Collobert (2014), have also been used in many NLP applications for different tasks.

Many of such works adopted the use of a low-rank matrix or tensor decomposition, including Levy and Goldberg (2014); Levy et al. (2015); Madhyastha et al. (2014, 2015); Pennington et al. (2014); Salle et al. (2016). One can view the idea of creating word embeddings using low-rank decomposition as a way of projecting a high-dimensional representation of words to a lower dimensional space where one projects words with a similar linguistic properties close to one another.

Even though our model implicitly induces a low dimensional embedding of feature vectors, in contrast to the works mentioned above, it is not the goal of our method. Nevertheless, these studies showed that embeddings generated by using low-rank constraint were able to capture useful semantic meanings of words and terms. In fact, Levy and Goldberg (2014) pointed that some popular neural-network-inspired word embedding models[1] are implicitly factorizing a word-context matrix. They also showed that explicitly using low-rank matrix factorization method can result in a better embedding for certain tasks.

Low-rank bilinear form Similar to our model, Madhyastha et al. (2014) and Madhyastha et al. (2015) use low-rank bilinear forms to induce their embeddings. They learned embeddings of the lexical space tailored to the target linguistic relation and showed that such method effectively learns using embeddings of a few dimensions. Like in our case, the low-dimensional latent projections in this paper are learned implicitly by imposing low-rank constraints on the predictions of the model. While the techniques they employed are similar to ours, as mentioned before, they are not novel techniques in machine learning. Both their and our studies are making use of available techniques and focusing on different properties of them for different hypotheses. The focus of their work is in inducing task-specific word embeddings from a more general ones, while in our case, we analyze the use of such techniques to perform multiclass classifications in sparse conjunctive feature spaces. Our hypothesis is that there are feature conjunctions that could be helpful for classification, but are unseen during training, thus by using low-rank constraint, we should be able to give weights to such–otherwise zero-weighted–conjunctions. Meanwhile, their hypothesis is that it is possible to induce low-dimensional word embeddings that are tailored for the target linguistic relation task. In our case, representations of an item are not necessarily word embeddings, and can be composed of–theoretically–unlimited number of elementary feature sets, hence unlimited dimension parameter tensor. The use of low-rank constraint in our case is crucial to give weights to unseen feature conjunctions, while in their case, unseen feature conjunctions do not exist. Not only such differences lead to different task settings and discussions, but practically,

[1] Such as Skip-Gram with Negative Sampling (SGNS) (Mikolov et al., 2013a,b) and Noise-Contrastive Estimation (NCE) embedding method (Mnih and Kavukcuoglu, 2013)

they also lead to different computational requirements, for which we propose an alternative implementation details to cater for bigger tensors that requires more memory. Our proposed implementation is specifically designed for sparse feature spaces, as presented in Section 3.3.2.1.

2.1.2 Low Rank Decomposition for Matrix/Tensor Completion

One can view the idea of giving weight to unobserved feature conjunctions as a problem of matrix completion of the parameter matrix. Different methods had framed NLP tasks as a matrix completion problem, and the use of low-rank constraints was shown to be effective in such cases. Some examples include general weighted automata (Balle and Mohri, 2012), Finite-State Transducers (FST) (Bailly et al., 2013a), Weighted Context Free Grammars (WCFG) (Bailly et al., 2013b), relation extraction (Chang et al., 2013, 2014; Fan et al., 2014; Nickel et al., 2012; Riedel et al., 2013; Singh et al., 2015; Yao et al., 2012) and entity classification (Yao et al., 2013).

We discuss some works related to classification tasks, specifically relation extraction and entity classification task.

2.1.2.1. Relation extraction

Relation extraction is a task of identifying semantic relationship between two objects such as nouns (Socher et al., 2012), named entities (Gormley et al., 2015; Yu et al., 2015) or Knowledge Base (KB) items (Bordes et al., 2011, 2013, 2014; Ngomo and Auer, 2011; Socher et al., 2013; Weston et al., 2013).

Among the different approaches to relation extraction, some low rank matrix completion approaches have also been used. For instance, Nickel et al. (2012) used it to address the problem of learning relations on Semantic Web's Linked Open Data(LOD). They performed a low-rank factorization of a sparse tensor that represent the labeling between two entities and a relation type. By performing such tensor completion, they were able to detect correlations between different relation types, and thus able to make decisions about unobserved relations between two entities.

Riedel et al. (2013), Yao et al. (2012) and Fan et al. (2014) also worked on relation extraction task by framing it as a matrix completion problem, where the rows of the matrix represent entity pairs and the columns represent the relations. In Riedel et al. (2013) and Yao et al. (2012), the columns are combinations of different relation schemas, such as relation in the schemas of pre-existing databases and surface form predicate relations (as in OpenIE (Etzioni et al., 2005)). In Fan et al. (2014), the columns are relational labels and textual features of the relation mentions. Recently,

Singh et al. (2015) combined both matrix and tensor factorization to incorporate per-entity information for a similar schema as in Riedel et al. (2013). Such per-entity information, they found, is useful to label unseen entity pairs.

Matricization It is interesting to note that while all the above work tried to complete values of entity-entity-relation triples, some represent them in a matrix instead of a tensor. In this thesis, we also propose such use of tensor matricization, i.e. representing a tensor as a matrix, and enforcing the low-rank constraint on the matrix.

2.1.2.2. Entity Classification

Yao et al. (2013) utilized the schema used in (Riedel et al., 2013) to address the problem of entity classification. Each row in the matrix corresponds to an entity and each column an entity type. In their case, the types include types appearing in natural language and those in pre-existing databases. They marked cells observed during training as true, and matrix completion will attempt to fill other cells that are unobserved by using probabilistic matrix factorization. By doing so, they were able to leverage various patterns of co-occurrences among entity types.

Even though we are also applying the low-rank constraint to entity classification, the notion of entity classification in our work is different. To put it simply, we define the task as classifying *mentions* of entities while for them it is a multi-label classification of entities based on pooled information of the mentions. Moreover, the utilization of the concept of matrix completion in our method is different; it is useful for completing a parameter matrix instead of an instance-class matrix. That is, we do not directly complete unknown labels of unobserved entities, but rather finding values of unobserved parameters, which indirectly will also help in classifying unobserved entities.

2.1.3 Low Rank Regularization for Controlling Model Capacity

In this section, we will discuss related work in NLP that uses low-rank constraints as a way to control their model parameters. Specifically, we will highlight some work that uses low-rank constraints in conjunctive feature spaces as well as those using nuclear norm penalty.

In this thesis, we use the low-rank constraint as a penalty in our learning objective to guide the learning algorithm to favor parameter matrix/tensor with lower rank. Our parameter space is a high-dimensional feature space that includes conjunctions of features from different features sets. To be able to use a convex optimization method, we relax such objectives using nuclear norm penalty.

2.1.3.1. High-dimensional conjunctive feature representation

Lei et al. (2014) used low-rank tensor learning in the context of dependency parsing, where like in our case dependencies are represented by conjunctive feature spaces. Their parameters are combinations of a sparse and low-rank tensor; each represents different sets of features. They induced the low-rank tensor from conjunctions of high-dimensional features of the head, modifier, and the dependency arc between the head and the modifier itself, by explicitly looking for a low-dimensional factorization of the tensor using a greedy alternating optimization.

In a more recent work, Yu et al. (2016) proposed a general framework using low-rank tensor for high-dimensional lexical features. They represent complex lexical features—comprised of parts of words, contextual information and labels—in a tensor that captures conjunction information among these parts. They reduced the tensor size by embedding each part of the features into a lower dimensional space, which results in smaller weight tensors. The embedding and the weight tensor is then learned jointly with a non-convex objective using stochastic gradient descent (SGD) with fixed rank. Such framework was able to achieve state-of-the-art results on tasks in relation extraction, PP-attachment, and preposition disambiguation.

Their setting is very similar to our work; our parameter space is also a combination of similar sets of features, i.e. words/mention, context and label. Their work helps to confirm our hypothesis that using low-rank constraints is suitable for a high dimensional conjunctive space. While the motivation is similar, our technical solution is different. In our case, we propose the technique of matricization of a tensor combined with a nuclear-norm relaxation to obtain a convex learning procedure.

2.1.3.2. Nuclear norm penalty

To impose a low-rank constraint, many methods, such as the Factorization Machine (Rendle, 2012), work with a fixed rank. In our case, we use a convex relaxation of the low-rank constraint by using nuclear norm regularization. Such relaxation was proposed by Srebro and Shraibman (2005) and has been used in a number of applications in machine learning. In NLP, for instance, Quattoni et al. (2014) proposed a model to learn latent-variable max-margin sequence taggers by combining a nuclear norm regularized optimization with a spectral technique. They showed that such regularizer constraint is an effective way to control the capacity of the model and that it outperforms standard l2 regularization. Madhyastha et al. (2014) and Madhyastha et al. (2015), which we have mentioned in Section 2.1.1, also uses nuclear norm relaxation in their learning objectives to generate embeddings and made a similar observation on comparison with ℓ_1 and ℓ_2 regularization.

Hutchinson et al. (2012) generalize a large class of ℓ_1 regularized models by adding a low-rank component to learn a language model. They decomposed the model parameters into a low-rank component that learns regularities in the training data and a sparse component that learns exceptions. They regularize the sparse component with ℓ_1, the low-rank component with nuclear-norm regularizer, and use ℓ_2 on both. They incorporated the low-rank component as a way to exploit similarities that might exist between different features, which a sparse solution cannot capture.

The methods and optimization method of these works are similar to our proposal, which is to use nuclear-norm constraint as learning penalty. Moreover, they also adopt a learning algorithm similar to ours; alternating between gradient step and penalty thresholding. Other than the application of such method to a different family of tasks, we also focus on the use of conjunctive feature spaces, an aspect not explored in other works. In our case, we assume the parameter space comes from a tensor space having at least three parts (word/mention, context and thus the integration of such space to our method is also an interesting aspect to explore.

2.2 Named Entity Classification and Related Tasks

One of the earliest cited papers in NE research is the work of Rau (1991) on extracting company names from text. Such task gained more attention after the Sixth Message Understanding Conferences (MUC-6) (Grishman and Sundheim, 1996) introduced a Named Entity sub-task where the goal is identifying and labeling the names of all the people, organization and geographic locations in a text, as well as identifying temporal terms, numeric expressions, monetary expressions and percentages. Henceforth, numerous scientific events and research had been dedicated to the task (Nadeau and Sekine, 2007).

In its development, NE research has expanded to encompass different problem definitions, including differences in NE types, corpus domains and languages. In general, we observe three main branches of NE studies[2]: (i) NE Recognition and Classification (NERC), (ii) NE Extraction (NEE) and (iii) NE Linking (NEL). Our task of NE classification is a sub-task of NERC.

In this section, we will provide a brief description of each task and their relationship to each other and to our work. Note that we will focus on the goals and settings of the different tasks, and not discuss the specific methods used to solve such tasks.

[2]In the literature, one might use different terms to refer to these tasks, for example, using Named Entity Recognition (NER) to refer to NERC or NEE. We clarify the intended use of such terms in this thesis by providing a brief description for each of them in this section.

2.2.1 Named Entity Recognition and Classification (NERC)

We define NERC as a task of identifying occurrences of named entities in a text and labelling them with one of the predefined classes. NERC is typically context-dependent, i.e. the same phrase or word in different contexts may refer to different NE types.

As the name suggested, we can further divide NERC into two sub-tasks: (i) recognition and (ii) classification. In Named Entity Recognition (NER), the goal is to identify text spans in a document that refers to an NE; this entails placing the exact beginning and end boundaries of each NE in a text sequence. In Named Entity Classification (NEC), we label each of the recognized NE into one of the predefined classes. One might attempt to solve both sub-tasks jointly (typically as a sequence tagging problem) or as separate elements (typically in a pipeline architecture).

In this thesis, we focus on the problem of NE classification, such as done in Fleischman (2001), Fleischman and Hovy (2002) and Elsner et al. (2009). In these papers, it is assumed that the named entities mentions have already been correctly extracted from the text and the task is to label them with correct entity types.

Fleischman and Hovy (2002) addressed the problem of subcategorization of person names into a more fine-grained types including *athlete*, *politician/government*, *clergy*, *businessperson*, *entertainer/artist*, *lawyer*, *doctor/scientist* and *police*. Their work extended a previous work on subcategorization of location into *country*, *city*, *street*, *territory*, *region*, *water*, *mountain* and *artifact* (Fleischman, 2001). In their settings, they assumed that all person/location names in the text have been correctly pre-identified by a different system.

Our domain is similar to Elsner et al. (2009), which focused on labeling identified named entity mentions as either a *person*, an *organization* or a *location*. They used pre-annotated corpus and extracted all strings in NE tags as their mentions. In our case, we add another category: *miscellaneous*, which refers to other named entities outside of person, organization and location, e.g. “russian”, “german”, “british”. We also introduce noise in the set of identified named entity mentions to include phrases that do not refer to a named entity (e.g. common phrases such as “year”, “thursday”, “government”). In a way, we might view our task as also performing a part of the NER sub-tasks; we also need to recognize which text spans are referring to named entities.

2.2.2 Named Entity Extraction (NEE)

In this thesis, we use the term NEE to refer to the task of compiling or expanding a list of named entities, such as extending an ontology (Alfonseca and Manandhar, 2002), extracting collections certain NE types (Etzioni et al., 2005) or completing

mappings of NE to classes of ontology and surface patterns (Yao et al., 2013). One can also refer to this task as a gazetteer expansion task.

The main difference of such tasks with NERC is that NEE typically use a pooled information of all NE occurrences in a text –or even a corpus– to make a decision. Meanwhile, an NERC system needs to make separate decisions for each possible NE mention in a text.

2.2.3 Named Entity Linking (NEL)

NEL in this thesis refers to the task of determining the identity of NE mentions in text, where the identity is represented by an object in a knowledge base. One can implement this as a continuation of the NERC task; after recognizing an NE in a text, we identify the real world object to which it refers. This task is different from NERC in that it requires connecting each NE mentions to a known object in the knowledge base. For example, in NERC, there might be two mentions of "John", both tagged as a *person*, but with NEL, one might be linked to a knowledge base item of "John Lennon" and another to "John Doe", two different individuals.

One of typical knowledge bases used by NEL systems is Wikipedia (Bunescu, 2006; Cucerzan, 2007; Ferragina and Scaiella, 2010; Kulkarni et al., 2009; Milne and Witten, 2008; Naderi et al., 2014a,b; Ratinov et al., 2011). Some other include Stanford TAP entity catalog (Guha and McCool, 2003), DBpedia Lexicalization Dataset (Bizer et al., 2009), and YAGO2 knowledge base (Hoffart et al., 2011a), which were used by NEL systems such as Dill et al. (2003), Mendes et al. (2011) and Hoffart et al. (2011b), respectively.

2.3 Template Filling and Related Tasks

Another task we explore in this thesis is the task of template filling. In this section, we will describe some of the tasks that are related to extraction of template slot-fillers. We observed two types of slot-filler tasks, one related to event/template completion –which is the type of slot-filler task we examine in this thesis– and another related to knowledge base population.

2.3.1 Event Template Completion

In this thesis, we consider the slot filling task of event template completion. Event templates are structured schemas that describe a specific event at a specific context. The example of events include a "terorrism event" described in a news article, or a "flight booking" event described in a dialog between a customer and an airline representative.

For example, Riloff (1996), Patwardhan and Riloff (2007), Huang and Riloff (2012) and Boros et al. (2014) addressed slot filling in the Fourth Message Understanding Conference (MUC-4) (Sundheim, 1992) data. The MUC-4 data contains events related to terrorism activities with slots such as perpetrator, weapon and victim.

Other studies, such as Ahn (2006); Ji et al. (2008); Li et al. (2013); Liao and Grishman (2010); Nguyen et al. (2016) performed event extraction on ACE (Automatic Content Extraction) (Doddington et al., 2004) dataset, which contains newswire articles describing events such as *Life/Die*, *Transaction/Transfer*, *Conflict/Attack*, *Movement/Transport*, etc.

More recently, Liu and Lane (2016) and Vu (2016) addressed a Spoken Language Understanding (SLU) problem of understanding speaker's intent on ATIS (Airline Travel Information Systems) dataset (Hemphill et al., 1990). The data set contains audio recordings of conversations related to flight reservations. The task is to extract information such as location of departure and arrival referred in the conversation.

A related task to event template completion is Automatic Event Schema Induction (AESI), which aims to automatically induce event templates from raw text. For example, works such as Chambers and Jurafsky (2011), Chambers (2013), Cheung et al. (2013) and Sha et al. (2016) learned to induce event templates from MUC-4 dataset.

The setting of the task in this thesis is similar to that of Boros et al. (2014). We frame the task as a classification problem, where given a candidate slot-filler, the system should recognize whether it is a slot-filler, and –if it is– classify it as one of the predefined slot types. The task we consider do not include the identification of the event type itself, and thus differs with the settings of most studies of event extraction on ACE and ATIS datasets.

2.3.2 Knowledge Base Population

In knowledge base population, slot filling plays a role of completing details or attributes of entities listed in the Knowledge Base (KB). One of the most known initiatives dedicated for KB population is the Text Analysis Conference Knowledge Base Population (TAC KBP) (Surdeanu, 2013), which defines the task as collecting information on certain attributes (or slots) of entities, which may be either persons or organizations. For instance, for an entity "Apple" that refers to the American multinational technology company[3], one of the slot that needs to be filled is the names of the founders. The KBP task is organized into two sub-tasks: (i) NEL and (ii) slot filling. NEL is important to identify mentions of KB items in text, as explained in Section 2.2.3. Information regarding the identified KB items can then be extracted from the text to fill their slots in the KB. Moreover, some slots can be filled by another

[3] `https://www.apple.com`

item in the KB, for which an NEL is required. The task of slot filling is then concerns with collecting information of the identified KB items. The task can also be framed as finding relations between entities, i.e. relation extraction task (Adel et al., 2016; Angeli et al., 2014; Roth et al., 2013). For example, the relation "org:founded by" between "Steve Jobs" and "Apple".

This task is similar to NEE in the sense that both typically use a pooled information from different sources of evidence to make decisions. For example, recent studies such as Das et al. (2017) tried to infer attributes from combining indirect evidences in the KB itself. For instance, one may infer "Seattle" as a slot-filler for the attribute of the city where "Melinda Gates" lives by observing that the knowledge base contains the path "Melinda"–*spouse*–"Bill"–*chairman*–"Microsoft"–*HQ-in*–"Seattle".

The difference with event template completion is that an event template completion system can only use local contexts of the slot-filler to make decisions about the particular mention of the slot-filler, meanwhile KBP can use information from any text resources or even from the KB itself. Moreover, while many KBP slot filling tasks are framed as relation extraction problem, event template completion are typically framed as a sequence labelling problem.

2.4 Supervision in Machine Learning for NLP

In this thesis, we focus on using machine learning techniques. One way to differentiate such systems is by the type of supervision they use for learning a model. Supervision in machine learning generally consists labeled examples. Different tasks of machine learning can be discriminated based on whether or not such labeled examples are used, meaning a learning method can be (i) unsupervised (using unlabeled data), (ii) supervised (using labeled data) or (iii) semi-supervised (using labeled and unlabeled data).

Supervised learning method Our approach employs a supervised learning method. The main challenge in training a supervised system is it typically requires the availability of an extensive collection of annotated data. Practically, such requirement is not always easy to meet. An effort of preparing a big set of annotated data can be costly, especially when dealing with a high number of named entities or slot types.

Minimal supervision Considering such challenge, we seek to minimize the supervision by requiring only a handful of examples for each entity/slot-filler type. We refer to such supervision as minimal supervision. We define a minimal supervision as a supervision that requires a small amount of annotation to produce a larger set of labeled training data that might contain noise or incorrect labelings.

In this section, we discuss the different types of supervision in NLP, focusing on tasks related to named entity and template slot-fillers.

2.4.1 Unsupervised Learning

The first one is unsupervised learning, for which only unlabeled text corpora are available for training. Given data items $x \in X$, the goal of unsupervised learning is to find interesting structure in the data X (Chapelle et al., 2010). For example, one might observe clustering of data items or estimate the underlying distribution which is likely to have generated the data.

Unsupervised learning tasks typically do not involve predefined classes or labels. For instance, some of the most popular unsupervised NLP tasks include document clustering (Steinbach et al., 2000), statistical language modelling (Bengio et al., 2003; Gotoh and Renals, 2003) and topic modelling (Blei, 2012).

In IE specifically, clustering and generative modelling have been used for named entity mentions disambiguation/coreference resolution (Elsner et al., 2009; Haghighi and Klein, 2007; Mann and Yarowsky, 2003) and event templates induction (Chambers and Jurafsky, 2011; Nguyen et al., 2015). An unsupervised NER task has also been attempted, for example by observing the time series distribution of a certain word in two newspapers (Shinyama and Sekine, 2004).

2.4.2 Supervised Learning

The second setting is supervised learning, whose goal is to map input items $x \in X$ to predefined output labels/targets $y \in Y$, given a training set made of pairs (x_i, y_i) (Chapelle et al., 2010). The supervised learning task is to model the distribution of pairs (x_i, y_i) in the training set. Typically, such task is evaluated by the ability of the model to correctly map new data items x in the test set to their corresponding label y.

Many NLP tasks fall into this category, such as chunking (Tjong Kim Sang and Buchholz, 2000), clause identification (Tjong Kim Sang and Déjean, 2001), semantic role labelling (Carreras and Màrquez, 2005). The IE tasks we consider in this thesis–NERC (Nadeau and Sekine, 2007) and template filling (Sundheim, 1992)–are also both a supervised learning task.

For both NERC and template filling, we can consider the task as (i) a pipeline of recognition task followed by classification task or (ii) a joint recognition and classification task. In both cases, we are mapping words/tokens or phrases to a predefined set of output labels, based on models learnt using the training set examples. In the first case, the recognition tasks' labels will refer to whether the word/phrase is a NE/slot-filler and the classification tasks' labels refer to the actual predefined labels of our interest. In the second case, the labels will refer to both information at the same time.

2.4.2.1. NERC

To recognize (and classify, in some cases) NE in a sentence, one needs to map each token in the sentence to a label that signifies if the token is part of an NE. Early studies on NERC (Borthwick et al., 1998; Chieu and Ng, 2002) learns classifiers of individual words/tokens of a text, and combine them with dynamic programming algorithm such as Viterbi to find the most likely sequence of labels. Other studies resort to a more structured prediction to find the sequence label, such as using Hidden Markov Model (HMM) (Bikel et al., 1997), Conditional Random Fields (CRF) (McCallum and Li, 2003a) and neural network (Lample et al., 2016; Ratinov and Roth, 2009). A hybrid model that combines different techniques has also been proposed, such as done by Florian et al. (2003), in which they combined robust linear classifier, Maximum Entropy (MaxEnt) classifier, transformation-based learning, and HMM.

In this thesis, we consider only the task of NE classification, given candidate NEs. We assume that the task of recognizing NEs in a sentence has already been done by previous system. Therefore, the task does not involve a sequence labelling problem and only concerns with mapping phrases to a set of labels. Such task is often a part of a pipeline NERC system, such as **?**, where the first module performs a sequence labelling task to identify the NEs and the second module classifies the identified NEs into their corresponding types. In our case, we assume the result of the first module to be noisy, i.e. it might contain non-entities, and thus we also need to recognize such noise and do not classify them as one of the predefined entity types. Having such classifier will lead to a more robust NERC system as it prevents more errors from the first module to propagate to the final result.

2.4.2.2. Template filling

In some studies, template filling has been approached as a relation classification task (Adel et al., 2016; Roth et al., 2013, 2015). In some other, it is considered a phrase classification or sequence labelling task (Boros et al., 2014; Liu and Lane, 2016; Patwardhan and Riloff, 2007; Surdeanu et al., 2010; Xu and Sarikaya, 2013), similar to the task of NERC.

In this thesis, we consider the second type of approach to template filling; a task of classifying candidate slot-fillers. Similar to our NE classification task setting, we also assume the set of candidate slot-fillers to be noisy. Instead of assuming such candidates to be available from previous system, we use a simple rule-based module to obtain the candidate slot-fillers, such as done in (Boros et al., 2014; Huang and Riloff, 2011, 2012).

2.4.3 Semi-Supervised Learning

Halfway between unsupervised and supervised learning is semi-supervised learning (SSL), which uses both labeled and unlabeled data. The training set, in this case, can be divided into two parts: the points $X_l := (x_1, \ldots, x_l)$, for which labels $Y_l := (y_1, \ldots, y_l)$ are provided, and the points $X_u := (x_{l+1}, ..., x_{l+u})$, the labels of which are not known (Chapelle et al., 2010). The goal of the model is similar to the supervised learning task; to map data items x to output labels/target y. The difference is that, unlike supervised learning, SSL takes into account the distribution of unlabeled items in X_u to learn the models.

A common way of performing SSL in NLP, is to train classifiers by estimating parameters of a generative model through iterative expectation-maximization (EM) techniques (Dempster et al., 1977), such as for text classification (Chapelle et al., 2010). Another way is by performing an iterative co-training (Blum and Mitchell, 1998) or bootstrap/self-training (Yarowsky, 1995) technique to expand the set of labeled training data by automatically label data items in the unlabeled set (Collins and Singer, 1999; Cucerzan and Yarowsky, 2002; Ji and Grishman, 2006; Kim et al., 2002; Niu et al., 2003). One can also use the unlabeled data to learn a good feature map of items x by creating auxiliary problems using unlabeled data and performing a joint empirical risk minimization on the auxiliary problems (Ando and Zhang, 2005). A more recent approach of semi-supervised learning is by incorporating word representations obtained from unlabeled data as a part of a supervised learning system, such as done by Collobert et al. (2011).

2.4.4 Labeled Examples from Minimal Supervision

The term minimal supervision in this thesis does not refer to a specific machine learning task in the way the previous learning supervision categories do. Rather, it refers to the way of building a labeled training set for either a supervised or a semi-supervised learning task. Traditionally, building labeled pairs (x_i, y_i) involves manually labelling each $x_i \in \mathcal{X}$ in the training set with its corresponding (y_i). The idea of minimal supervision is to either reduce or completely eliminate such effort.

For example, in knowledge base population, it is common to use occurrences of existing knowledge base instances (slot entity or relations) in unlabeled texts as training examples for a supervised learning task,(Mintz et al., 2009; Surdeanu et al., 2010). Such task is commonly known as *distance learning*. Another example is by manually creating "seed" examples of NE spellings to perform NEE (Li et al., 2010; Neelakantan and Collins, 2014a). In this case, all occurrences of such spellings in the corpus are considered as positive labeled training examples. Similarly, Collins and Singer (1999), Etzioni et al. (2005) and Patwardhan and Riloff (2007) uses only a handful of "seed" rules/extraction patterns to build the initial training set and

use it on SSL task with bootstrapping technique for NEE, NERC and template filling, respectively.

The tasks we consider in this thesis are supervised learning tasks with labeled training sets built using minimal supervision. Specifically, we follow the idea of creating "seed" examples of NEs or slot-fillers and use occurrences of such examples in unlabeled text to build our training set. Such a way of building the training set might lead to a noisy training set, but it presents a cheaper alternative to manually labelling a large size of examples.

Minimal Supervision vs Unsupervised Learning Due to the nature of creating the labeled training examples that does not require manual labelings of example pairs, systems using minimal supervision are sometimes referred as unsupervised in literatures. To avoid confusion with the term *unsupervised learning* used in this thesis, recall that we use the term *unsupervised*, *supervised* and *semi-supervised learning* to distinguish whether the distribution of labeled examples $(x_l, y_l) \in X_l$ and unlabeled examples $x_u \in X_u$ affect the resulting models rather than to distinguish the effort involved in creating the labeled examples X_l.

3 Low Rank Regularization for Sparse Conjunctive Feature Space

The content of this chapter is mainly published in Primadhanty et al. (2015).

We consider a problem of classifying items $x \in \mathcal{X}$ into one of the prespecified classes $y \in \mathcal{Y}$. In Natural Language Processing (NLP), items $\mathcal{X}$ can represent various things such as documents, paragraphs, sentences, phrases, words or even characters. Meanwhile, the classes $\mathcal{Y}$ can represent any set of labels such as topics, sentiments, entity types or semantic types.

In this thesis, we frame the classification problem as finding a discriminative model that models the probability of an item x belonging to a certain class y. Specifically, we adopt a linear feature-based model and represent x as combinations of several feature sets. For instance, one might represent a phrase in a sentence with features of its surrounding context and features of the phrase itself, as well as the combination between features of the two sets. We refer to such combination of features as *conjunctive features*.

To learn the model, we use a convex optimization method and impose a low-rank constraint as a mean of regularizing. The main goal of such method is to address the sparsity problem in high-dimensional conjunctive feature spaces.

This chapter is organized as follows. First, we present some notations that will be used in the remaining of the thesis in Section 3.1. In Section 3.2, we describe the model as well as introduce the notion of conjunctive feature space. We then present the framework for training such models using low-rank regularization and a convex learning method in Section 3.3.

3.1 Notations

In general, we use **bold** symbols to represent arrays. Specifically, bold lower-case alphabets ($\mathbf{x}$) represent vectors and bold upper-case alphabets ($\mathbf{X}$) represent matrices and higher order tensors. Scalar components/elements of arrays are denoted with their indices in square brackets. For example, $\mathbf{X}[i, j]$ refers to the element in row i and column j of matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, with $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, n\}$.

We use $\mathbf{x} \cdot \mathbf{y} \in \mathbb{R}$ to denote the dot product and $\mathbf{x} \otimes \mathbf{y} \in \mathbb{R}^{m*n}$ to denote the Kronecker product between $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{y} \in \mathbb{R}^n$. The Kronecker product between two vectors–$\mathbf{x}$ and $\mathbf{y}$–results in a block vector $[(\mathbf{x}[1] * \mathbf{y}), \ldots, (\mathbf{x}[n] * \mathbf{y})]$, that is

$$(\mathbf{x} \otimes \mathbf{y})[(i-1) * m + j] = \mathbf{x}[i] * \mathbf{y}[j] \quad ,$$

where $i \in \{1 \ldots m\}$ and $j \in \{1 \ldots n\}$.

3.2 Linear Feature-based Model

This section provides a general description of linear feature-based models and discusses the related parameter spaces in the domain of NLP.

Let us have an item x and a set of possible classes $y \in \mathcal{Y}$. Our goal is to model the conditional probability of y being the class of x:

$$\Pr(y|x;\boldsymbol{\theta}) = \frac{\exp\{s_\theta(x,y)\}}{\sum_{y' \in \mathcal{Y}} \exp\{s_\theta(x,y')\}} \quad , \tag{3.1}$$

where $s_\theta : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ is a real-valued scoring function parameterized by θ.

In the literature, it is common to employ a *linear feature-based model*. That is, one define the scoring function $s_\theta(x,y)$ of the model as a dot product between a representation $\boldsymbol{\phi}(x,y)$ and parameters $\boldsymbol{\theta}$:

$$s_\theta(x,y) = \boldsymbol{\phi}(x,y) \cdot \boldsymbol{\theta} \quad . \tag{3.2}$$

We discuss below the details of the representation $\boldsymbol{\phi}(x,y)$ and the parameters $\boldsymbol{\theta}$.

3.2.1 Conjunctive Feature Spaces

In practice, there are variations of feature spaces of $\boldsymbol{\phi}(x,y)$, which –as we will show below– are generally a *conjunctive feature* space. Typically, features that represent a candidate x and a class y can be grouped into separate *elementary features* that express different information of the pair $\langle x,y \rangle$. In a conjunctive feature space, representations $\boldsymbol{\phi}(x,y)$ are made up of the independent elementary feature sets and the conjunctions of the features between the sets. We discuss below several instances of such feature spaces.

Multiclass feature spaces In the typical setting of multiclass classification, the compatibility of an item x with a class y is measured based on two key elements: (i) features $\boldsymbol{\phi}_{\mathrm{x}}(x) \in \mathbb{R}^d$ that represent the item and (ii) the class' parameter vector

$\mathbf{w}_y \in \mathbb{R}^d$ that indicates the weight/importance of each feature for the class y. In this case, the scoring function is a dot product between the two:

$$\boldsymbol{\phi}(x, y) \cdot \boldsymbol{\theta} = \boldsymbol{\phi}_{\mathrm{x}}(x) \cdot \mathbf{w}_y \quad or \tag{3.3}$$

$$= \Big[\boldsymbol{\phi}_{\mathrm{y}}(y) \otimes \boldsymbol{\phi}_{\mathrm{x}}(x)\Big] \cdot \mathbf{w} \quad , \tag{3.4}$$

where $\mathbf{w} = [\mathbf{w}_1, \ldots, \mathbf{w}_{|\mathcal{Y}|}]$ and $\boldsymbol{\phi}_{\mathrm{y}}(y) \in \{0, 1\}^{|\mathcal{Y}|}$ is an indicator vector representing the current y:

$$\boldsymbol{\phi}_{\mathrm{y}}(y)[i] = \begin{cases} 1 & \text{if } i = y \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

Indicator feature spaces In NLP, some of the popular feature vectors to represent text spans are indicator feature vectors $\phi_x : \mathcal{X} \rightarrow \{0, 1\}^d$ that indicate the occurrence of different words in the text or whether certain characteristics of a word are observed (e.g. all-capitalized, is-capitalized, all-digits, alphanumeric, etc.). In many cases, we might want to define features that combine two or more of such features to capture a more fine-grained information. For example, it might be that knowing that a word is capitalized or if its alphanumeric by itself is not discriminative for the task at hand but knowing that the word is both capitalized AND alphanumeric is highly discriminative. In practice, such feature conjunctions are often engineered; one should decide what conjunctions are relevant for the problem. We can express all conjunctions of such feature sets as a product of $\boldsymbol{\phi}_{\mathrm{x}}(x) \otimes \boldsymbol{\phi}_{\mathrm{x}}(x)$. In this case, the parameter vector of a class $\mathbf{w}_y \in \mathbb{R}^{d*d}$ will contain the weights of each feature conjunction, and the scoring function can be written as

$$\boldsymbol{\phi}(x, y) \cdot \boldsymbol{\theta} = \Big[\boldsymbol{\phi}_{\mathrm{y}}(y) \otimes \boldsymbol{\phi}_{\mathrm{x}}(x) \otimes \boldsymbol{\phi}_{\mathrm{x}}(x)\Big] \cdot \mathbf{w} \quad . \tag{3.6}$$

Input tuple feature spaces Another common setting is to consider an input x as a tuple of different information about x, such as $x = \langle x_a, x_b \rangle$ where x_a and x_b are different views about x. For example, if x is a phrase, then x_a can be information about the phrase itself and x_b can be information about the context where x occurs (e.g. the document, the surrounding words, etc.). We can then define two feature functions $\boldsymbol{\phi}_{\mathrm{x_a}}(x_a) \in \mathbb{R}^{d_{x_a}}$ and $\boldsymbol{\phi}_{\mathrm{x_b}}(x_b) \in \mathbb{R}^{d_{x_b}}$ to represent x_a and x_b respectively. Similar to the previous case, conjunctive features for this case are also often engineered. The full conjunctive feature space of such features can be expressed as $\boldsymbol{\phi}_{\mathrm{x_a}}(x_a) \otimes \boldsymbol{\phi}_{\mathrm{x_b}}(x_b)$, and the parameter vector of a class $\mathbf{w}_y \in \mathbb{R}^{d_{x_a} * d_{x_b}}$ will contain the weights of each feature conjunction, and thus the scoring function can be written as

$$\boldsymbol{\phi}(x, y) \cdot \theta = \boldsymbol{\phi}(\langle x_a, x_b \rangle, y) \cdot \boldsymbol{\theta} \quad , \tag{3.7}$$

$$= \Big[\boldsymbol{\phi}_{\mathrm{y}}(y) \otimes \boldsymbol{\phi}_{\mathrm{x_a}}(x_a) \otimes \boldsymbol{\phi}_{\mathrm{x_b}}(x_b)\Big] \cdot \mathbf{w} \quad . \tag{3.8}$$

Note that in all of the instances mentioned above, the definition of the model (equation 3.1) remains the same, but with various definitions of the feature and parameter space.

In practice, we might have a more elaborate feature space, such as considering x as an input tuple with more than two elements. Nevertheless, we can see that the parameter space is always a conjunctive space, even in the simplest form (Equation 3.3). In our case, we would like to capitalize on such feature conjunctions without having to manually engineer the choices of conjunctions that are relevant for our problem. Our method takes the full conjunctive feature space and automatically decides on the relevant conjunctions by learning from the evidence in the training set. For feature conjunctions that are not observed during training, the method will consider their relevance based on their similarities to those that are observed. In the following sections, we will discuss the method's strategy of realizing such capabilities.

3.2.2 Parameter Tensor

Instead of regarding the parameters of the model as a vector, we regard them as a parameter tensor, where each dimension of the tensor corresponds to an elementary feature vector. For instance, in the case of having an input tuple (as in Equation 3.8), our model will have a parameter tensor $\mathbf{W} \in \mathbb{R}^{|\mathcal{Y}| \times d_{x_a} \times d_{x_b}}$ in place of the parameter vector $\mathbf{w} \in \mathbb{R}^{|\mathcal{Y}| * d_{x_a} * d_{x_b}}$. But instead of having the feature vector $\boldsymbol{\phi}(x, y)$ defined as a tensor as well, we use an unfolding of the parameter tensor and employ a bilinear form of the scoring function.

Matricization/unfolding A common framework for tensor computations is by turning the tensor (of order more than two) into a matrix and then perform matrix computations on the resulting matrix instead of the original tensor form. It is commonly known as the process of *matricization* or *tensor unfolding*. Given an n-th order tensor $\mathbf{T} \in \mathbb{R}^{d_1 \times \cdots \times d_n}$, we "unfold" it into a matrix $\mathbf{M} \in \mathbb{R}^{D_1 \times D_2}$, where $D_1 * D_2 = d_1 * \cdots * d_n$. To put it simply, we rearrange the components of the tensor into a matrix form. This is done by partitioning the tensor axes into two sets, one for matrix rows and another for columns.

In the case of our parameter tensor, unfolding the tensor is equal to grouping the elementary feature vectors into two groups, and associate one group with the columns of the resulting matrix, and the other group with the rows. For example, to create an unfolding $\mathbf{W}'$ from the parameter tensor $\mathbf{W} \in \mathbb{R}^{|\mathcal{Y}| \times d_{x_a} \times d_{x_b}}$, we can set the rows of $\mathbf{W}'$ to represent the classes and the columns to represent the features of the input, i.e. $\mathbf{W}' \in \mathbb{R}^{|\mathcal{Y}| \times (d_{x_a} * d_{x_b})}$.

For the remainder of this chapter, unless specified otherwise, we will denote as $\mathbf{W}$ the matricized version of the parameters θ of our models.

Bilinear scoring function With a matricized $\mathbf{W}$, we can rewrite the scoring function as

$$s_\theta(x, y) = \boldsymbol{\phi}_1(x, y)^\top \mathbf{W} \boldsymbol{\phi}_2(x, y) \quad , \tag{3.9}$$

where $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ is the weight matrix and $\boldsymbol{\phi}_1 : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^{d_1}$ and $\boldsymbol{\phi}_2 : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^{d_2}$ are feature functions that combine elementary feature functions according to the way we unfold $\mathbf{W}$. For example, if we use the matricization $\mathbf{W} \in \mathbb{R}^{|\mathcal{Y}| \times (d_a * d_b)}$ for the case of equation 3.8 above, we will have $\boldsymbol{\phi}_1(x, y) = \boldsymbol{\phi}_{\mathrm{y}}(y)$ and $\boldsymbol{\phi}_2(x, y) = \boldsymbol{\phi}_{\mathrm{x_a}}(x_a) \otimes \boldsymbol{\phi}_{\mathrm{x_b}}(x_b)$.

Note that regardless of the way we represent the parameters $\boldsymbol{\theta}$ (as vector, matrix or tensor), the model remains a linear feature-based model; the scoring function $s_\theta(x, y)$ is as a dot product between a representation $\boldsymbol{\phi}(x, y)$ and the parameters $\boldsymbol{\theta}$. In the case of our bilinear scoring function (Equation 3.9) above, the model corresponds to a linear feature-based model operating in the feature space of $\boldsymbol{\phi}_1 \otimes \boldsymbol{\phi}_2$, i.e. the score has one term for each feature conjunction:

$$s_\theta(x, y) = \sum_{i,j} \boldsymbol{\phi}_1(x, y)[i] * \boldsymbol{\phi}_2(x, y)[j] * \mathbf{W}[i, j] \quad , \tag{3.10}$$

where $\mathbf{W}[i, j]$ refers to the component of $\mathbf{W}$ that corresponds to the weight of the conjunctive feature $\langle \boldsymbol{\phi}_1[i], \boldsymbol{\phi}_2[j] \rangle$. Notice that it is trivial to include elementary features of $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$, in addition to their conjunctions, by having a constant component in each of the two representations set to 1.

In our method, the way of representing the parameters is important because the method benefits from exploiting the rank of a parameter matrix. In the following sections, we will see how the rank of the matrix plays an important role when learning the parameters and how the decision of the way we unfold the parameter tensor might affect the notion of the rank.

3.3 Low-Rank Learning

This section will discuss the framework for learning the linear feature-based model as explained in the previous section.

We learn the parameters of the model in a supervised manner; i.e. we use information from examples of labeled items to estimate the values of the parameters. Let $\mathcal{P} = \{(x_1, y_1), \cdots, (x_m, y_m)\}$ be a training set containing m examples of labeled items in the form of item-label pairs (x_i, y_i). Our learning goal is to estimate the values of the parameters that allow our model to correctly classify the pairs in $\mathcal{P}$.

We define such learning objective as finding parameters $\mathbf{W}$ that minimizes the "cost" of classification errors $L(\mathbf{W})$:

$$\underset{\mathbf{W}}{\operatorname{argmin}} L(\mathbf{W}) \quad . \tag{3.11}$$

Recall that we define our model as in Equation 3.1; it models conditional probabilities of y given x. Therefore, we define a negative log-likelihood loss function for our objective as follows:

$$L(\mathbf{W}) = \sum_{(x,y)\in\mathcal{P}} -\log \Pr(y|x;\mathbf{W}) \quad . \tag{3.12}$$

With such loss function, the learning objective will favor a model parameter $\mathbf{W}$ that sets a high conditional probability for the correct class y for each item x in the training set $\mathcal{P}$.

The ultimate goal of the model is not only to correctly predict classes of items that are in the training set $\mathcal{P}$, but also to be able to classify new items that are not in the training set (i.e. unseen data). A common challenge in learning a model in sparse high-dimensional feature spaces is avoiding the model from overfitting the training data (thus cannot generalize to unseen data). One way to avoid overfitting is by controlling the capacity of the model. which can be done by imposing a regularization term to the objective function:

$$\underset{\mathbf{W}}{\operatorname{argmin}} L(\mathbf{W}) + \tau R(\mathbf{W}) \quad , \tag{3.13}$$

where $R(\mathbf{W})$ is a regularizer and τ is a constant that trades off error and capacity.

Typically, a regularizer $R(\mathbf{W})$ is a penalty on the complexity of $\mathbf{W}$, such as ℓ_1 and ℓ_2 penalty. An ℓ_1 regularizer favours a sparse $\mathbf{W}$ by limiting the size of the parameters, while an ℓ_2 regularizer favours $\mathbf{W}$ with no extreme values by putting more penalty to larger parameter values. In our case, the regularization favors solutions where the rank of $\mathbf{W}$ is smaller.

In our case, the regularization will also allow the method to determine the values of parameters that are not observed in the training data, and thus generalizes the model to unseen data even more.

In the following sections we will discuss such regularization technique and the optimization scheme we employ to learn the parameters.

3.3.1 Low-Rank and Nuclear Norm Regularization

Different factors can affect the problem of overfitting. Among them are the number of parameters, the size of training data and the regularizer used. In our model, the

number of parameters is determined by the size of the conjunctive feature space. In NLP, such feature spaces are commonly large. Moreover, we are interested in training with minimal supervision, and thus the size of the training data will be small. With such settings, controlling the capacity of the model is crucial to avoid overfitting the training data.

A standard approach to control the capacity of a linear classifier is to use ℓ_1 or ℓ_2 regularizer. However, the main limitation of ℓ_1 and ℓ_2 regularization is that they cannot let the model give weights to conjunctions that were unobserved during training. With our loss function, the learning method will only optimize values of the parameters that are observed during training, so the values of the unobserved ones will remain zero. In principle, ℓ_1 and ℓ_2 work by thresholding and normalizing the values of the parameters. Therefore, during learning, the ℓ_1 or ℓ_2 penalty will only affect parameters with non-zero values, and thus will not affect the unobserved conjunctions. Without the ability to give weight to unseen conjunctions, it is unlikely that the model will generalize to novel examples where most of the conjunctions will be unseen in the training set.

One possible solution to unseen conjunctions is to impose a strong prior on the weight vector so that it assigns weight to unseen conjunctions. But such approach is non-trivial, we need to formulate how to build such a prior; deciding what kind of reasonable constraints we can put on unseen conjunctions.

Another common approach to handle high-dimensional conjunctive feature space is to reduce the size of the space by manually selecting features that are "relevant". But this can be time-consuming, and we might need to have different sets of features for different classification tasks. Ideally, we expect to have a learning algorithm that eliminates the need for such feature engineering and that it can automatically leverage rich conjunctive feature spaces.

Based on such observation, we propose the use of a low-rank regularizer, with which we can encourage the model to induce a low-dimensional projection of the elementary feature vectors. The projection allows us to map similar conjunctions closer, and thus give similar weights to similar conjunctions, including to those that are unseen. Such ability will help prediction by generalizing to novel examples where most of the conjunctions will be unseen in the training set.

3.3.1.1. Low-rank regularization

Regarding our parameters as matrices (as in Equation 3.9) allows us to control the capacity of the model via regularizers that favor parameter matrices with low rank.

Using negative log-likelihood loss function and low-rank regularizer, we rewrite our objective as

$$\underset{\mathbf{W}}{\operatorname{argmin}} \sum_{(x,y)\in\mathcal{P}} -\log \Pr(y|x;\mathbf{W}) + \tau \operatorname{rank}(\mathbf{W}) \quad . \tag{3.14}$$

While a matrix has a clear definition of rank, it is not the case for general tensors, and there exist various definitions in the literature. This is the main reason behind our choice of matricization; once the tensor has been turned into a matrix, we can use the standard definition of matrix rank. A main advantage of this approach is that we can make use of standard routines like singular value decomposition (SVD) to decompose the matricized tensor.

Intrinsic dimensionality To see the effect of low-rank regularizers, consider a singular value decomposition (SVD) $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ of the parameter matrix $\mathbf{W}$. $\mathbf{U} \in \mathbb{R}^{d_1 \times k}$ and $\mathbf{V}_y \in \mathbb{R}^{d_2 \times k}$ are orthonormal projections and $\mathbf{\Sigma} \in \mathbb{R}^{k \times k}$ is a diagonal matrix of singular values. We can then rewrite the scoring function as

$$s_\theta(x,y) = \left(\boldsymbol{\phi}_1(x,y)^\top \mathbf{U}\right) \mathbf{\Sigma} \left(\mathbf{V}^\top \boldsymbol{\phi}_2(x,y)\right) \quad . \tag{3.15}$$

We can see that the rank k is the intrinsic dimensionality of the inner product behind the score function; a low k implies that the score is defined as the inner product of few projected features. This means a low-rank regularizer will favor parameter matrices that have low intrinsic dimensionality.

The rank k of the matrix $\mathbf{W}$ is the number of non-zero singular values in $\mathbf{\Sigma}$. Therefore, an SVD decomposition is not a dimensionality reduction, but rather a change of basis for moving from a d_1 and d_2 dimensions to k dimensions, and thus no loss of information is suffered. A dimensionality reduction is done by removing the less significant (non-zero) singular values in $\mathbf{\Sigma}$, which entails a loss of information.

Weighting unseen conjuctions Projecting our feature vectors to a lower-dimensional vector means projecting similar elementary features to the same dimension and thus similar feature conjunctions will have similar weights. This ability is an important aspect of our method that will allow us to utilize unseen feature conjunctions for prediction.

Choice of unfolding Recall that our parameter matrix $\mathbf{W}$ is a matricized higher order tensor. In this case, we can view the way we unfold $\mathbf{W}$ as a way of implying the kind of latent projections that will be induced by the low-rank regularization. Depending on the task, some configurations might be more beneficial than others.

3.3.1.2. Nuclear norm regularization

Optimizing with a low-rank regularizer leads us to a non-convex problem. Therefore, we make use of a convex relaxation based on the nuclear norm (Srebro and Shraibman, 2005). The nuclear norm of a matrix $\mathbf{W}$, denoted $\|\mathbf{W}\|_*$, is the sum of its singular values:

$$\|\mathbf{W}\|_* = \sum_i \mathbf{\Sigma}_{i,i} \quad ,$$

where $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ is the singular value decomposition of $\mathbf{W}$. This norm has been used in a number of applications in machine learning as a convex surrogate for imposing low rank, e.g. in Srebro et al. (2005). With this relaxation, our final objective function is

$$\operatorname*{argmin}_{\mathbf{W}} \sum_{(x,y)\in\mathcal{P}} -\log \Pr(y|x;\mathbf{W}) + \tau\|\mathbf{W}\|_* \quad . \tag{3.16}$$

3.3.2 Optimization

A convex objective function (Equation 3.16) allows the use of different convex optimization methods. In this thesis, we use a simple optimization scheme known as forward-backward splitting (FOBOS) (Duchi and Singer, 2009), mainly for its ease of implementation and its flexibility to be adapted for different tasks.

In a series of iterations, this algorithm performs a gradient update followed by a proximal projection of the parameters, as shown in Algorithm 1. The proximal projection (line 6 of Algorithm 1) projects the parameters $\mathbf{W}$ to take into account the regularization penalty R that is defined by our objective function (see Equation 3.13). For example, for ℓ_1 it thresholds the parameters, for ℓ_2 it scales them, and for nuclear norm regularization it thresholds the singular values – as shown in Algorithm 2.

Hyperparameters The algorithm have three hyperparameters input: learning rate η, regularization factor τ and maximum number of iterations T. Setting the value of **maximum number of iterations** is trivial, it will not affect the result of the optimization in each iteration. Depending on the task, some models might need more iterations than others before converging. Practically, in our case, we implement such that we can continue the optimization process from the last result of previous optimization process. This allows us to begin the experiment with few iterations and then continue with more afterwards, in case we observe that it has not converged. The **learning rate** controls how big is the step size in each iteration. Notice here that the algorithm uses a decaying step size, so the step size will become smaller each iteration. So specifying a too small learning rate might cause the step size to be very close to zero in the final iterations. In general, using a too low learning

Algorithm 1: FOBOS

Data: η (learning rate), τ (regularization factor), T (max iterations)
Result: $\mathbf{W}_T$

initialize $\mathbf{W}_0$ as zero matrix ;
for $t = 0 \dots T\text{-}1$ **do**
    compute step size $\eta_t = \frac{\eta}{\sqrt{t+1}}$;
    compute gradient $\mathbf{g}(\mathbf{W}_t)$ of loss L at $\mathbf{W}_t$;
    update $\mathbf{W}_{t+0.5} = \mathbf{W}_t - \eta_t \mathbf{g}(\mathbf{W}_t)$;
    $\mathbf{W}_{t+1}$ = update $\mathbf{W}_{t+0.5}$ to take into account the regularization penalty R;
end

Algorithm 2: Nuclear Norm Proximal Projection for FOBOS

Data: η_t, τ, $\mathbf{W}_{t+0.5}$
Result: $\mathbf{W}_{t+1}$

compute SVD $\mathbf{W}_{t+0.5} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$;
define a new matrix $\widetilde{\mathbf{\Sigma}}$ with diagonal elements $\tilde{\sigma}_i = \max(\sigma_i - \eta_t\tau, 0)$;
set $\mathbf{W}_{t+1} = \mathbf{U}\widetilde{\mathbf{\Sigma}}\mathbf{V}^\top$;

rate the improvement will be slow, while using a too high learning rate will decay the loss faster, but can cause the parameters to move erratically and thus unable to settle in the optimal point. In some of our experiments, instead of setting the learning rate, we uses a line search to find the best step size to use at each iteration. The effect of **regularization factor** can be seen in difference between training and validation performances. When the validation performance is far below training, it indicates overfitting and thus a higher regularization factor might be needed. However, this might also indicate insufficient training data, which in many of our experiments–where we use only a small portion of the training data–is a possibility. One way of choosing a good regularization factor is a greedy approach; start with a low value and increase it until we don't see an increase in validation performance.

3.3.2.1. Sparse implementation of FOBOS

With nuclear norm regularization, the FOBOS algorithm requires the calculation of SVD decomposition of the $\mathbf{W}$ matrix in each iteration. Depending on the choice of technology used for implementing the algorithm, full SVD routines can be memory-intensive. Moreover, when the size $\mathbf{W}$ is large–which will be the case with high-dimensional feature spaces–it might be impossible to fit the full $\mathbf{W}$ matrix in memory. Therefore, we apply minor modifications to the algorithm for a sparse implementation version that does not require full SVD computation and does not require fitting the full $\mathbf{W}$ in memory.

Algorithm 3: Sparse Plus Low Rank (SPLR) SVD

Data: $\mathbf{L} \in \mathbb{R}^{d_1 \times k}$, $\mathbf{R} \in \mathbb{R}^{k \times d_2}$, $\mathbf{S} \in \mathbb{R}^{d_1 \times d_2}$, k, *maxIter* // $\mathbf{A} = \mathbf{S} + \mathbf{LR}$
Result: $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{V}$

initialize $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$ as zero matrices ;
for $i = 1 \dots k$ **do**
  $\mathbf{x_0}$ = random $\mathbb{R}^{d_2}$;
  for $j = 1 \dots maxIter$ **do**
    compute $\mathbf{x}_j$: // $\mathbf{x}_j = \mathbf{A}^\top \mathbf{A} \mathbf{x}_{j-1}$

$$\begin{aligned}\mathbf{x}_j =& \mathbf{S}^\top(\mathbf{S}\mathbf{x}_i) + \mathbf{S}^\top(\mathbf{L}(\mathbf{R}\mathbf{x}_i)) - \mathbf{S}^\top((\mathbf{U}\mathbf{\Sigma})(\mathbf{V}^\top\mathbf{x}_i)) + \\ & \mathbf{R}^\top(\mathbf{L}^\top(\mathbf{S}\mathbf{x}_i)) + \mathbf{R}^\top(\mathbf{L}^\top(\mathbf{L}(\mathbf{R}\mathbf{x}_i))) - \mathbf{R}^\top(\mathbf{L}^\top((\mathbf{U}\mathbf{\Sigma})(\mathbf{V}^\top\mathbf{x}_i))) - \\ & \mathbf{V}(((\mathbf{U}\mathbf{\Sigma})^\top)(\mathbf{S}\mathbf{x}_i)) - \mathbf{V}(((\mathbf{U}\mathbf{\Sigma})^\top)(\mathbf{L}(\mathbf{R}\mathbf{x}_i))) + \\ & \mathbf{V}(((\mathbf{U}\mathbf{\Sigma})^\top)((\mathbf{U}\mathbf{\Sigma})(\mathbf{V}^\top\mathbf{x}_i)));\end{aligned}$$

  end
  $\mathbf{v}_i = \mathbf{x}_j / \|\mathbf{x}_j\|$;
  $\mathbf{A}_i\mathbf{v}_i = \mathbf{S}\mathbf{v}_i + \mathbf{L}(\mathbf{R}\mathbf{v}_i) - (\mathbf{U}\mathbf{\Sigma})(\mathbf{V}^\top\mathbf{v}_i)$;
  $\boldsymbol{\sigma}_i = \|\mathbf{A}_i\mathbf{v}_i\|$;
  $\mathbf{u}_i = \mathbf{A}\mathbf{v}_i / \boldsymbol{\sigma}_i$;
  $\mathbf{U}[:, i] = \mathbf{u}_i[:, 0]$; $\mathbf{V}[:, i] = \mathbf{v}_i[:, 0]$; $\mathbf{\Sigma}[i, i] = \boldsymbol{\sigma}_i$;
end

In principle, the implementation is based on decomposing the $\mathbf{W}$ matrix and performing an SVD designed especially for matrices that can be decomposed as a sum of a sparse matrix and a low-rank matrix.

Decomposing $\mathbf{W}$ We define a decomposition of $\mathbf{W} \in \mathbb{R}^{d_1 \times d_2}$ with rank k into vectors with lower dimensions based on its SVD decomposition:

$$\mathbf{W} = (\mathbf{U}\mathbf{\Sigma})\mathbf{V}^\top \quad , \tag{3.17}$$

$$= \mathbf{L}\mathbf{R} \quad , \tag{3.18}$$

with $\mathbf{L} \in \mathbb{R}^{d_1 \times k}$ and $\mathbf{R} \in \mathbb{R}^{k \times d_2}$. For a sparse space, one can assume that the actual rank of $\mathbf{W}$ is much less than either d_1 or d_2. With some modifications to the original algorithm, we can avoid the need of creating the full $\mathbf{W}$, such that we only need to fit $\mathbf{L}$ and $\mathbf{R}$ into memory. For instance, we can calculate the score by first computing the projection of each side, i.e. $\langle \boldsymbol{\phi}_1(x, y)^\top \mathbf{L} \rangle$ and $\langle \mathbf{R}^\top \boldsymbol{\phi}_2(x, y) \rangle$.

Sparse plus low rank SVD To perform an SVD, we employ the power method (Golub and Van der Vorst, 2000) where we compute one singular vector and value at a time. The algorithm is designed to perform SVD on matrices that can be represented as a sum of a sparse matrix and a low rank matrix. We refer to such SVD as a Sparse

Algorithm 4: Modified FOBOS for sparse implementation

Data: η (learning rate), τ (regularization factor), T (max iteration), k (rank)
Result: $\mathbf{L}_T$, $\mathbf{R}_T$

initialize $\mathbf{L}_0$ and $\mathbf{R}_0$ as zero matrix ;
for $t = 0 \dots T\text{-}1$ **do**
  compute gradient $\mathbf{g}(\mathbf{L}_t, \mathbf{R}_t)$ of loss L at $\mathbf{L}_t$, $\mathbf{R}_t$;
  compute step size $\eta_t = \frac{\eta}{\sqrt{t+1}}$;
  compute sparse gradient matrix $\mathbf{S}_t = -\eta \mathbf{g}(\mathbf{L}_t, \mathbf{R}_t)$;
  compute SVD $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V} = \text{svdsplr}(\mathbf{L}_t, \mathbf{R}_t, \mathbf{S}_t, k)$;
  threshold $\widetilde{\mathbf{\Sigma}}$: $\tilde{\sigma}_i = \max(\sigma_i - (\min(\sigma_{i'} \in \mathbf{\Sigma})), 0)$;
  update $\mathbf{L}_{t+1} = \mathbf{U}\widetilde{\mathbf{\Sigma}}$;
  update $\mathbf{R}_{t+1} = \mathbf{V}^\top$;
end

Plus Low Rank (SPLR) SVD (Algorithm 3). In this case, it considers the input matrix $\mathbf{W}_{t+0.5}$ as a sum of two matrices:

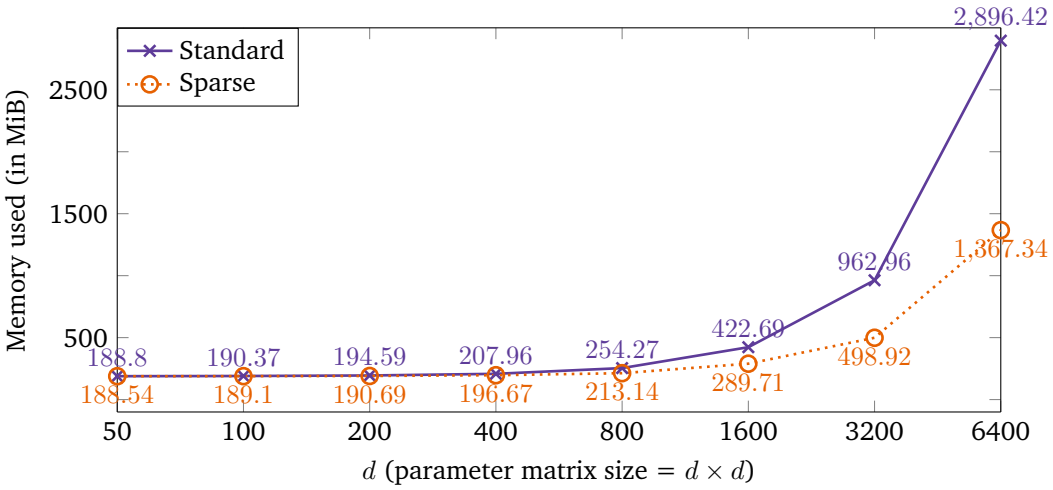
$$\begin{aligned}\mathbf{W}_{t+0.5} &= \mathbf{W}_t - \eta_t \mathbf{g}(\mathbf{W}_t) \quad , \\ &= (\mathbf{U}\mathbf{\Sigma})\mathbf{V}^\top - \eta_t \mathbf{g}(\mathbf{W}_t) \quad , \\ &= \mathbf{L}\mathbf{R} + \mathbf{S} \quad ,\end{aligned}$$

where $\mathbf{S}$ is the sparse gradient $\mathbf{g}(\mathbf{W}_t)$ matrix. The gradient matrix is sparse due to the fact that it is calculated based on our loss function (Equation 3.12) that only takes into account conjunctions that are seen in the training set. In practice, even though $\mathbf{S}$ is of size $d_1 \times d_2$, we can store it as a sparse matrix, for which almost every commonly used programming languages allows a memory-efficient storage.

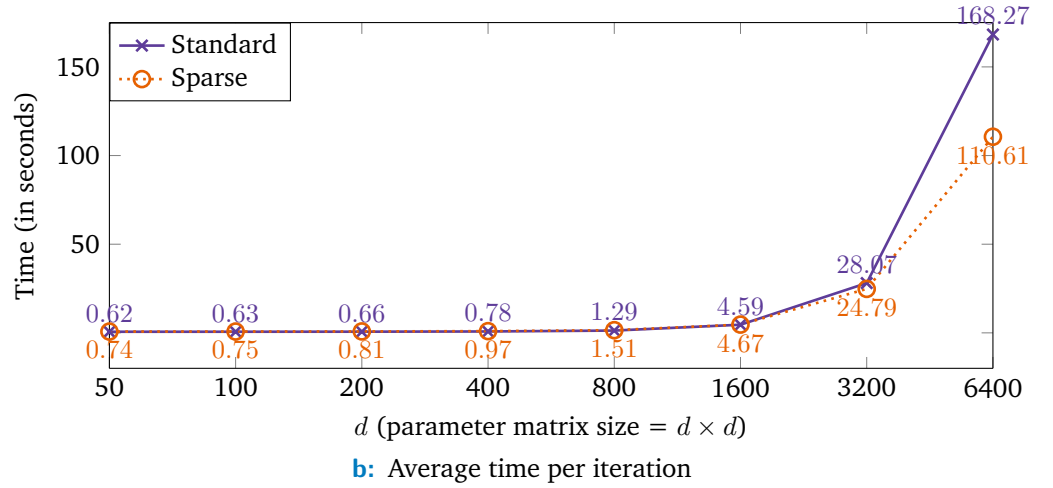
With the above modifications, we can rewrite the algorithm as shown in Algorithm 4. Note that we design this modification only with memory limitation in mind, disregarding its possible effect to speed. Duchi and Singer (2009) proposed a method for efficient implementation in high dimensions that relies on a lazy update for $\mathbf{W}$ elements whose gradient are zero. However, it is inapplicable for low-rank regularization since, after the projection, even elements with gradient zero might change in value and become non-zero.

3.3.2.2. Evaluating Sparse Implementation of FOBOS

In this section, we would like to know the effect of using the sparse implementation to the performance of the model. The goal is to compare the standard implementation and the sparse implementation. Knowing the limitations and benefits of both implementations can help in: (i) choosing the implementation to use according to the experimental constraints and (ii) further improve the implementation of our method.

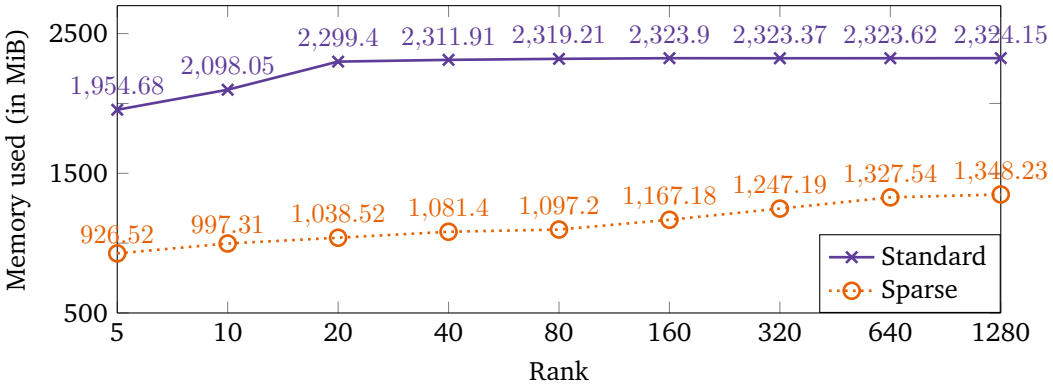


a: Maximum memory usage

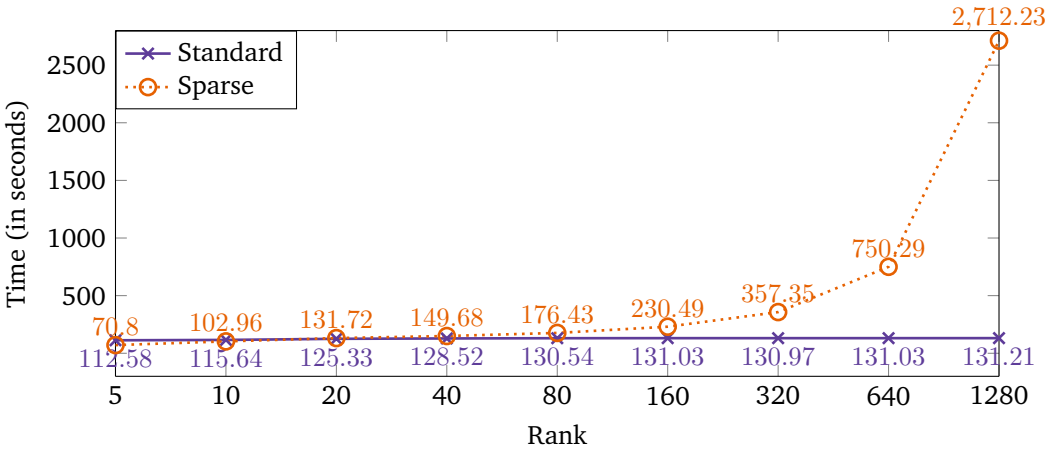


b: Average time per iteration

Fig. 3.1.: Maximum memory used and average time per iteration using standard implementation and sparse implementation for data with **varying matrix size**. Parameter matrix rank = 5, number of iterations = 5, maxIter for SPLR SVD = 1, number of training data = 100.



a: Maximum memory usage



b: Average time per iteration

Fig. 3.2.: Maximum memory used and average time per iteration using standard implementation and sparse implementation for data with **varying matrix gradient rank**. Parameter matrix size = 5000×5000, number of iterations = 5, maxIter for SPLR SVD = 1, number of training data = 100.

We have implemented both variants of the FOBOS algorithm (standard and sparse) in python[1] and train models in the form described in Chapter 4 from randomly generated training data. First, we compare the maximum memories used and time per iteration needed to train models with parameter matrix of varying sizes. In Figure 3.1, we can see that as the size of the parameter matrix expands, the difference of required memories between the two implementations becomes more significant. The time required to run one iteration, on the other hand, do not show a significant difference.

Nevertheless, it is important to note that in Figure 3.1, we fix the underlying rank of the parameter matrix to be quite small, i.e. 5. As we can see from Algorithm 3, the factorization for the sparse implementation performs one iteration for each rank, thus we can expect that it will take longer time to factorize matrices with higher rank. Such observation is shown in Figure 3.2, where this time we fix the parameter matrix size and vary the underlying rank instead. Figure 3.2b shows that the time required for the standard implementation is consistent throughout varying ranks, but it is almost exponential for the sparse implementation.

In terms of memory usage, we observe a consistent trend in which the sparse implementation requires less memory than the standard implementation. The standard implementation stores a dense parameter matrix and thus requires more or less the same memory regardless of the rank. Meanwhile the sparse implementation kept the factorized parameter matrices whose size depends on the rank, and thus we can see that the memory requirement increases as the rank increases, as shown in Figure 3.2a.

As for the performance differences, there is almost no difference for the data we use in the above experiments, as shown in Figure 3.3. We show the final score of the objective function after five iterations, and we can see that the difference between the two implementations is very small.

From these results, it is clear that there is a significant tradeoff between memory and time required to train models with different implementations. When memory is not an issue, the standard implementation is preferred. But when memory is limited, the sparse implementation can be used to cut down the memory usage up to more than half of the standard implementation. Note also that in the case of the sparse implementation, we need to specify the rank k as an input to the algorithm, which–as we can see in Figure 3.2–will affect the memory usage and time. In our experiment in the following section (see Figure 4.6), only few intrinsic dimensions, hence small rank, is required to obtain a good performance, and thus we can experiment using small values of k for a more efficient memory usage and a faster computation.

[1] https://www.python.org

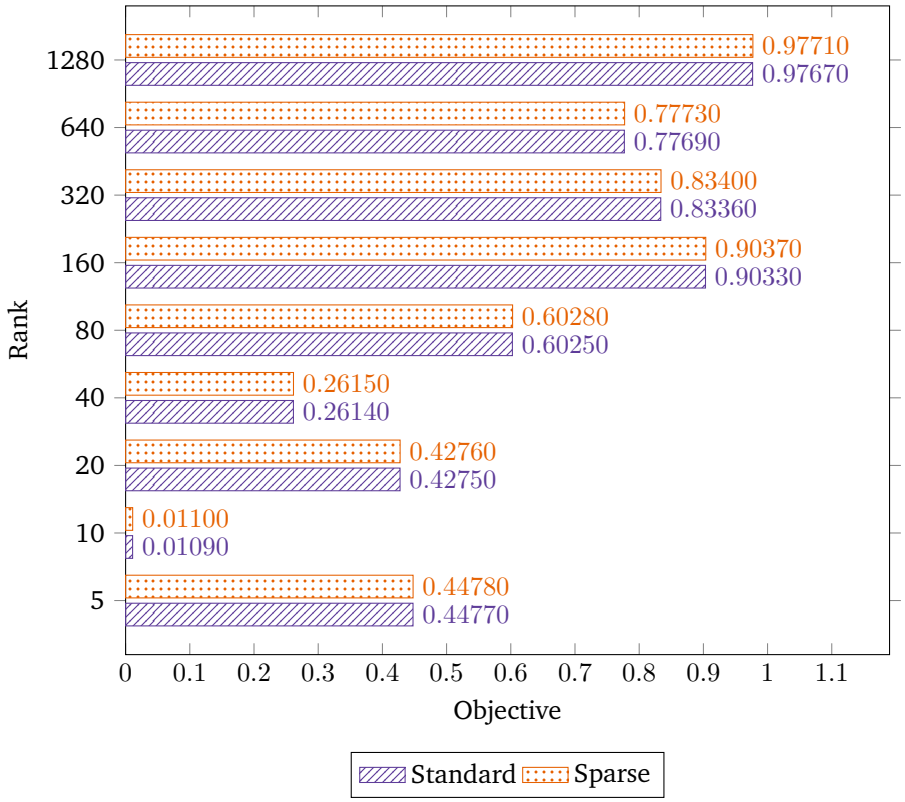


Fig. 3.3.: Final objective after 5 iterations using standard implementation and sparse implementation for data with varying parameter matrix gradient rank. Parameter matrix size = 5000×5000, maxIter for SPLR SVD = 1, number of training data = 100.

Of course, one thing to remember is that these implementations might not be the most efficient implementations, and with better implementation strategies, it is possible to improve the performances further.

3.4 Conclusion

In summary, this chapter describes the method of low-rank regularization for sparse conjunctive feature spaces. Our method combines available techniques to compose an effective way of utilizing sparse conjunctive feature spaces. The strategy is based on representing our parameter space as tensor and imposing a low-rank penalty to learn the parameter tensor. In this case, the notion of rank is defined as the rank of the matricized tensor.

The low-rank penalty serves as a way of controlling the capacity of the model while at the same time allowing the model to give weights to unseen feature conjunctions. With such capability, our model has an advantage in generalizing to new examples–where most of the conjunctions will be unseen in the training set–compared to standard regularizations such as ℓ_1 or ℓ_2.

Moreover, our method employs a convex learning objective, making it possible to train the model using different convex optimization methods. In this chapter, we present a simple optimization method and discuss its implementation details.

We detailed two implementation strategies that adopted either a dense or a sparse approach in storing and processing the parameter matrices. We showed empirical performance comparisons between the two implementations and observed that the sparse implementation is a plausible alternative for training models with larger parameter matrices when memory is limited.

4 Low Rank Regularization for Named Entity Classification

The content of this chapter is mainly published in Primadhanty et al. (2015).

In this chapter, we apply low-rank regularization on Named Entity (NE) classification task. We propose the use of "seed" set as a form of minimal supervision and introduce a new evaluation framework for such task that focuses only on unseen named entities. Empirical experiments show that low-rank regularization performs better than ℓ_1 and ℓ_2 regularizer on sparse conjunctive feature spaces.

The organization of this chapter is as follows. In Section 4.1, we will introduce the task of NE classification. We then continue by describing the specific low-rank models with minimal supervision for NE classification in Section 4.2 and report experimental results in Section 4.3. Finally we summarize our findings and conclude our analysis in Section 4.4.

4.1 Introduction

Sentence 1 **[Barack Hussein Obama II]** is the 44th and current President of the United States, ...

Sentence 2 Born in Honolulu, Hawaii, **[Obama]** is a graduate of Columbia University and Harvard Law School, ...

Fig. 4.1.: Sentences annotated with named entity of type PERSON.

Entities are instances of predefined categories that represent real life objects, such as names of organizations, persons, locations, etc. Information about entities and their relations can help in different Natural Language Processing (NLP) related tasks, such as question answering, ontology learning, biomedical information extraction or machine translation. Being able to identify such entities in text documents is key to understanding the content of the document. During the Message Understanding Conference-6 (MUC-6) (Grishman and Sundheim, 1996), such task was introduced as a sub-task of Information Extraction (IE), and hencefort gained popularity.

An entity mention is a phrase in a sentence that refers to an entity. For example, both sentences[1] 1 and 2 in Figure 4.1 contain a mention of entity "Barack Obama", even though the phrases that refer to the entity are not identical.

[1] Obtained from http://en.wikipedia.org/wiki/Barack_Obama

The categories in which entities belong are called entity types, which can be any possible grouping of the entities. For example we can define that the entity "Barack Obama" has an entity type PERSON together with other entities referring to a person name, or we can define that it has an entity type PRESIDENT together with other entities referring to a president of a country.

The goal of Named Entity Recognition and Classification task (NERC) is to identify mentions of entities in a text and label them with one of several predefined entity types. One can use an annotated training set and train a model that learns the characteristics, distributions, or a set of statistical rules and patterns that distinguish a named entity and use it to annotate other unlabeled data.

Named Entity classification task As the name suggests, it is possible to view NERC as two separate sub-tasks: (i) NE recognition and (ii) NE classification. One might devise a method that tries to do both at the same time, or develop two different strategies to tackle them. In this thesis, we focus only on the task of classification, that is, we assume that the candidate entities are given to the system, similar to Elsner et al. (2009); Fleischman (2001); Fleischman and Hovy (2002). Nevertheless, the classification task we consider introduces an extra challenge; we consider that the candidate entities also include "noise". That is, the system responsible in performing the NE recognition might also extract non-entities. Therefore, the goals of our classifier are twofold: determine whether a candidate is an entity and–if it is–determine to which entity type it belongs.

Evaluation of unseen entities There are two aspects of the problem of NERC, one is the ambiguity of entities with respect to their semantic class and another is recognition of unseen named entities. In corpora with unambiguous entities, the more crucial problem is on recognizing words that we have never seen during training as named entities. For example, in the case of recognizing treatments and conditions in a medical domain corpus, we can assume that in general there is not much ambiguity, i.e. a proper noun is either a condition (e.g. hemophilia, hepatitis, anemia), a treatment (e.g. aspirin, wellbutrin, adderall) or none of those. With little ambiguity amongst the entities, classifying seen entities is as trivial as memorizing the labels of said entities in the training data. Therefore, in such cases, the real challenge is in recognizing unseen entities. Moreover, when only a small number of annotated data is available, most of the named entities that we will want to predict in the unlabeled corpus will be unseen entities, and thus we would like to focus more on tackling this challenge. Accordingly, we propose an evaluation setting that focuses on measuring the performance of the model in classifying unseen entities.

4.1.1 Contributions

Finally, the specific contributions of this chapter are:

- We describe a specific instance of our low-rank model for the task of NE classification. We describe the conjunctive feature spaces we consider for this task, as well as the scoring function that we apply on such feature spaces.
- We present a minimal supervision strategy to build the training data for NE classification task. The minimal supervision strategy is based on the assumption that the corpus presents little ambiguity. We show how such ambiguity assumption holds in the dataset we use.
- We evaluate the performance of state-of-the-art NERC systems on unseen entities only. Such evaluation drops performances of the systems significantly, implying that classifying unseen entities is an important challenge.
- We conduct experiments to show that the proposed regularization framework is better for sparse conjunctive feature spaces than standard ℓ_1 and ℓ_2 regularization.

4.2 Low-Rank Named Entity Classification Models with Minimal Supervision

In this section, we describe a specific implementation of our low-rank regularization model for the task of named entity classification. First, in Section 4.2.1 we describe the setting of minimal supervision to build the training set used for learning the model. Then in Section 4.2.1, we describe the model.

4.2.1 Minimal Supervision for Named Entity Classification

To reduce annotation efforts, we propose the use of minimal supervision to gather training data. Our minimal supervision setting is based on the assumption that the data presents little ambiguity. With such assumption, we also propose to focus the evaluation of NE classifiers in its ability to classify unseen entities.

To do supervised learning of our model, a training set containing annotated data is required. Instead of annotating documents to build the training set, we propose the use of *seed* set. A seed set contains examples of named entities for each of our predefined entity types.

Generating training data This minimal supervision setting is based on the assumption that entities are unambiguous. In such case, we can expect every occurrence of the seed entities in the unlabeled text to be example mentions of the seeds. For example, having a seed "John Doe" for type *person*, we expect all occurrences of "John Doe" in the text to refer to a *person*. Therefore, we can can extract them as a positive training data add it to the training set. Meanwhile, for as training data for non-entities, we use single nouns that are not part of any examples in the seed set.

Assuming that each seed entity will have numerous occurrences in the unlabeled corpus, we can gather a considerable amount of training data by using only a handful examples. Such requirement is more practical compared to manually annotating the unlabeled documents.

In the literatures, this approach is also commonly used for a semi-supervised learning such as *bootstrapping* (self-training/co-training) (Blum and Mitchell, 1998; Mitchell, 1999; Yarowsky, 1995), in which the training set is iteratively expanded by adding extraction results of the classifiers in the previous iteration.

4.2.2 Low-rank Named Entity Classification Models

In this section, we describe a specific family of low-rank models for classifying entity. We define as input a candidate entity tuple $x = \langle l, e, r \rangle$ with its respective entity mention e and context information, which consists of its left l and right context r. The goal is to classify x into one entity class y in the set $\mathcal{Y}$. We will use log-linear models as described in Section 3.2 of the form shown in Equation 3.1, where θ are the parameters of this function that depends on the conjunctive feature space defined for the model. We will describe below the space that uses information from the left l and right r contexts of the input tuple and then proceed to extend it into utilizing the mention information e as well.

4.2.2.1. A Low-rank Model of Left-Right Contexts

We start from the observation that when representing tuple objects such as $x = \langle l, e, r \rangle$ with features, we often depart from a feature representation of each element of the tuple, i.e. elementary features. Hence, let ϕ_{l} and ϕ_{r} be two feature functions representing left and right contexts, with binary dimensions d_l and d_r respectively. For now, we will define a model that ignores the entity mention e and makes predictions using context features.

It is natural to define conjunctions of left and right features. Hence, in its most general form, one can define a matrix $\mathbf{W}_y \in \mathbb{R}^{d_l \times d_r}$ for each class, such that $\theta = \{\mathbf{W}_y\}_{y \in \mathcal{Y}}$. Borrowing notations from Equation 3.9, we define ϕ_1 and ϕ_2 as combinations of the elementary feature vectors ϕ_{l} and ϕ_{r} with the class indicator feature vector ϕ_{y} (Equation 3.5). We can then define the score function as

$$s_\theta(x, y) = \phi_1(x, y)^\top \mathbf{W} \phi_2(x, y) \quad , \tag{4.1}$$

$$s_\theta(\langle l, e, r \rangle, y) = \phi_{\text{l}}(l)^\top \mathbf{W} [\phi_{\text{r}}(r) \otimes \phi_{\text{y}}(y)] \quad , \tag{4.2}$$

with $\mathbf{W}$ being the matricized parameter tensor $\mathbf{W}' \in \mathbb{R}^{d_l \times d_r \times |\mathcal{Y}|}$ in the conjunctive feature space. To put it simply, $\mathbf{W}$ is a block matrix containing $\mathbf{W}_y$ of every y.

4.2.2.2. Adding Entity Features

The model above classifies entities based only on the context. Here we extend it to make use of features of the entity mention itself. Let $\mathcal{T}$ be a set of possible entity feature tags, i.e. tags that describe an entity, such as IsCapitalized, ContainsDigits, SingleToken, ... Let ϕ_{e} be a feature function representing entities. For this case, to simplify our expression, we will use a set notation and denote by $\phi_{\text{e}}(e) \subseteq \mathcal{T}$ the set of feature tags that describe e. Our model will be defined with one parameter matrix per feature tag, i.e. $\theta = \{\mathbf{W}_t\}_{t \in \mathcal{T}}$. The model form is:

$$s_\theta(\langle l, e, r \rangle, y) = \sum_{t \in \phi_{\text{e}}(e,y)} \phi_{\text{l}}(l)^\top \mathbf{W}_t [\phi_{\text{r}}(r) \otimes \phi_{\text{y}}(y)] \quad , \tag{4.3}$$

where in this case $\mathbf{W}$ consists of $\mathbf{W}_t$ of every t.

4.2.2.3. Learning with Low-rank Constraints

Both scoring functions in Equation 4.2 and Equation 4.3 are specialized instances of Equation 3.9, and thus learning the parameters (as described in Section 3.3) requires the matricization of the parameter tensor. In the case of Equation 4.2, we will have the parameter $\mathbf{W}$ as a matricized matrix corresponding to a tensor in the space of ϕ_{l}, ϕ_{r} and ϕ_{y}. In this case, we have several options of unfolding the parameter tensor; we can either group the context together, or separate them (combining one with the class parameter). Meanwhile, Equation 4.3 introduces an additional tensor dimension (i.e. ϕ_{e}) to the parameter tensor, thus increasing the options of possible unfoldings. We now have a tensor with up to four axes, namely left and right context representations, entity features, and entity classes.

Although the equations above show specific ways of unfolding the tensor, we can choose a different unfoldings for the same feature space. In general, different ways of partitioning the tensor axes will lead to different notions of intrinsic dimensions. In our case we choose the left context axes as the row dimension, and the rest of axes as the column dimension, i.e. $\phi_1 = \phi_{\text{l}}$ and $\phi_2 = \phi_{\text{r}} \otimes \phi_{\text{e}} \otimes \phi_{\text{y}}$. In preliminary experiments we tried variations, such as having right prefixes in the columns, and left prefixes, entity tags and classes in the rows. We only observed minor, non-significant variations in the results.

4.3 Experiments

In this section we evaluate our regularization framework for training models in high-dimensional sparse conjunctive feature spaces. We run experiments on learning entity classifiers with minimal supervision. We focus on classification of unseen entities to highlight the ability of the regularizer to generalize over conjunctions that

Class	10-30 Seed	Nb Mentions			
		10-30	40-120	640-1920	All
PER	clinton, dole, arafat, yeltsin, wasim akram, lebed, dutroux, waqar younis, mushtaq ahmed, croft	334	747	3,133	6,516
LOC	u.s., england, germany, britain, australia, france, spain, pakistan, italy, china	1,384	2,885	5,812	6,159
ORG	reuters, u.n., oakland, puk, osce, cincinnati, eu, nato, ajax, honda	295	699	3,435	5,271
MISC	russian, german, british, french, dutch, english, israeli, european, iraqi, australian	611	1326	3,085	3,205
O	year, percent, thursday, government, police, results, tuesday, soccer, president, monday, friday, people, minister, sunday, division, week, time, state, market, years, officials, group, company, saturday, match, at, world, home, august, standings	5,326	11,595	31,071	36,673

Tab. 4.1.: For each entity class, the seed of entities for the **10-30** set, together with the number of mentions in the training data that involve entities in the seed for various sizes of the seeds.

are not observed at training. We perform our experiments using CoNLL-2003 Shared Task data (Tjong Kim Sang and De Meulder, 2003), and compare the performance to ℓ_1 and ℓ_2 regularizers, as well as with state-of-the-art systems.

4.3.1 Task Definition and Evaluation Metric

We define a named entity classification task with minimal supervision. In this task, we are given a list of named entities (i.e. seed) for each entity class and a corpus of unlabeled documents as input. The goal is to learn from it and train a model to classify *possible* mentions of named entities in a text. During test time, we are given candidates of named entity mention and the context in which they appear. These candidates may also include non-entity mentions. Therefore, the goals of the classifier can be described as two steps classification; first classifying if a candidate is entity or non-entity, and then, for those that are classified as entity, classifying it as one of the predefined entity types.

We evaluate the performance of models by measuring their performance on unseen entities. We use of precision, recall and F1 score as the evaluation metric. Precision measures the proportion of predictions that are correct. Recall measures the proportion of the true annotations that are retrieved. F1 score is the harmonic mean of recall and precision.

	training	dev.	test
PER	6,516 (3,489)	1,040 (762)	1,342 (925)
LOC	6,159 (987)	176 (128)	246 (160)
ORG	5,271 (2,149)	400 (273)	638 (358)
MISC	3,205 (760)	177 (142)	213 (152)
O	36,673 (5,821)	951 (671)	995 (675)

Tab. 4.2.: The number of mentions (and number of unique candidates in parenthesis) for each entity type of CoNLL-2003 data used in our experiments.

4.3.2 Data and Setting

We use the CoNLL-2003 English data, which is annotated with four types: person (PER), location (LOC), organization (ORG), and miscellaneous (MISC). In addition, the data is tagged with parts-of-speech (PoS), and we compute word clusters running the Brown clustering algorithm (Brown et al., 1992) on the words in the training set.

Candidate entities Similar to Elsner et al. (2009), we extract all strings in NE tags as our entity mentions. We add single nouns that are not part of an entity as non-entities mentions. Accordingly, we add a new label "O" to denote non-entity type. Both entity and non-entities mentions will be referred to as *candidates* in the remaining of this section.

Minimal supervision setting To create the seed set for the minimal supervision setting, we select the most frequent non-ambiguous mentions in our training set. We lowercase all candidates and remove the ambiguous ones (i.e., those with more than one label in different mentions). Then we create supervision seeds by picking the N most frequent training candidates for entity types, and the M most frequent candidate non-entities. We create seeds of various sizes N-M, namely **10-30**, **40-120**, **640-1920**, as well as **all** of the candidates. For each seed set, the training set will contain all training candidates that involve entities in the seed. Table 4.1 shows the smaller seed, as well as the number of mentions for each seed size.

Evaluation For evaluation we use the development and test sections of the data. To evaluate on unseen entities, we exclude all candidates that match the training seed (i.e., that are in the **all** seed) from the development and test data. We do not remove ambiguous instances from the tests. As evaluation metric we use the average F1 score computed over all entity types, excluding the non-entity type. Note that this implies a baseline that always predicts non-entity (i.e. the most frequent class) will obtain an F1 score of zero.

After removing the ambiguous candidates from the training data, and removing candidates seen in the training from the development and test sets, the number of

mentions (and number of unique candidates in parenthesis) in the data used in our experiments is shown in Table 4.2

Features	Window	Bag-of-words		N-grams	
		Lexical	Cluster	Lexical	Cluster
Elementary features of left and right contexts	1	13.63	**14.59**	13.63	**14.59**
	2	**15.49**	13.86	13.08	**13.54**
	3	12.18	**14.45**	12.14	**13.28**
Only full conjunctions of left and right contexts	1	12.90	**13.75**	12.90	**13.75**
	2	8.59	**8.85**	12.31	**12.43**
	3	8.57	**10.59**	10.15	**10.49**
Elementary features and all conjunctions of left and right contexts	1	15.30	**16.98**	15.30	**16.98**
	2	**13.26**	12.89	14.28	**15.33**
	3	**11.87**	11.54	**13.94**	**13.15**

Tab. 4.3.: Average-F1 of classification of unseen entity candidates on development data, using the **10-30** training seed and ℓ_2 regularization, for different conjunctive spaces (elementary only, full conjunctions, all). **Bag-of-words** elementary features contain all clusters/PoS in separate windows to the left and to the right of the candidate. **N-grams** elementary features contain all n-grams of clusters/PoS in separate left and right windows (e.g. for size 3 it includes unigrams, bigrams and trigrams on each side).

Elementary features We refer to context as the sequence of tokens before (left context) and after (right context) a candidate mention in a sentence. Different classifiers can be built using different representations of the contexts. For example we can change the window size of the context sequence (i.e., for a window size of 1 we only use the last token before the mention and the first token after the mention). We can treat the left and right contexts independently of each other, we can treat them as a unique combination, or we can use both. We can also choose to use the word form of a token, its PoS tag, a word cluster, or a combination of these.

Table 4.3 compares different context representations and their performance in classifying unseen candidates using maximum-entropy classifiers trained with Mallet McCallum (2002) with ℓ_2 regularization, using the **10-30** seed. We use the lexical representation (the word itself) and a word cluster representation of the context tokens and use a window size of one to three. We use two types of features: bag-of-words features (1-grams of tokens in the specified window) and n-gram features (with n smaller or equal to the window size).

The performance of using word clusters is comparable, and sometimes better, to using lexical representations. Moreover, using a longer window, in this case, does not necessarily result in better performance. In the rest of the experiments we will use the elementary features that are more predictive and compact for our context: clusters and PoS tags in windows of size at most 2.

To represent an entity candidate we use standard traits of the spelling of the mention, such as capitalization, the existence of symbols, as well as the number of tokens

cap=1, cap=2 :
whether the first letter of the entity candidate is uppercase, or not

all-low=1, all-low=0 :
whether all letters of the candidate are lowercase letters, or not

all-cap1=1, all-cap1=0 :
whether all letters of the candidate are uppercase letters, or not

all-cap2=1, all-cap2=0 :
whether all letters of the candidate are uppercase letters and periods, or not

num-tokens=1, num-tokens=2, num-tok>2 :
whether the candidate consists of one token, two or more

dummy :
a tag that holds for any entity candidate, used to capture context features alone

Fig. 4.2.: The 12 entity tags used to represent entity candidates. The tags *all-cap1* and *all-cap2* are from Neelakantan and Collins (2014a).

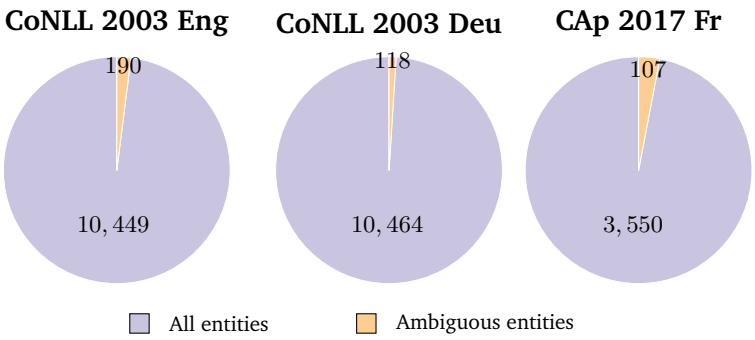


Fig. 4.3.: Proportion of ambiguous entity mentions in various NER datasets, in different languages; English (CoNLL 2003 Eng), German (CoNLL 2003 Deu) and French (CAp 2017 Fr). The numbers correspond to unique mentions (case-insensitive) tagged as named entities in the datasets.

in the candidate. See Figure 4.2 for the definition of the features describing entity candidates.

4.3.3 Ambiguity Assumptions on CoNLL-2003 Data

Our minimal supervision setting relies on the assumption that the data have little ambiguity. We are also evaluating on unseen entities assuming that seen entities will always have the same label, hence not an interesting problem to evaluate. In this section, we examine closer such assumptions on the data we use in our experiments; CoNLL-2003 data.

4.3.3.1. Unambiguous seed set for building training data

The ambiguity assumption in the seed set is that the same mentions of a named entity will always have the same label. Therefore, we extract all mentions–strings

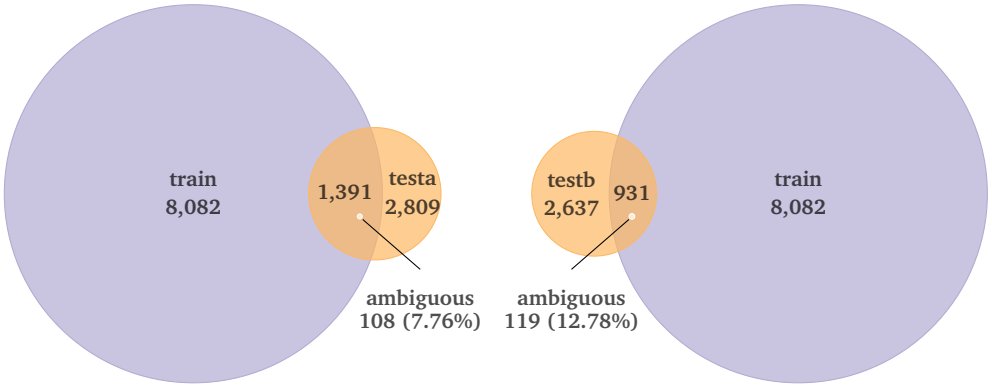


Fig. 4.4.: Proportion of ambiguous entity mentions in seen entities in the development and test set of CoNLL-2003 data. The numbers corresponds to unique mentions (case-sensitive) of entities tagged as PER, LOC, ORG or MISC in the dataset.

in NE tags–in the data and compare the labels of mentions of the same entity. We group the mentions in a case-insensitive manner, because our seed set is also used in the same way. We consider entities that have more than one label as an ambiguous entity. Here we only consider entities labeled as 'PER', 'LOC', 'ORG' and 'MISC', as in the original dataset.

Figure 4.3 shows the proportion of ambiguous entities in the CoNLL 2003 English corpus that we use in our experiments. We can see that there is less than 3% ambiguity, which implies that using a seed set in this corpus will result in a reliable training data; there is little chance that the seed will include a false example from the corpus. In addition, we also computed entity mention ambiguities in two other NERC datasets with different languages (CoNLL 2003 German Corpus and CAp 2017 French Corpus (Lopez et al., 2017)), as shown in Figure 4.3. The CoNLL 2003 German dataset is similar to the English dataset; it contains entities tagged with the same labels as the CoNLL 2003 English dataset. The CAp 2017 French dataset is comprised of tweets written in French, tagged with 13 labels including *Person*, *Location*, *Organization*, *Product*, *Event*, etc. For these two other datasets, we also observed that the amount of ambiguous entities is less than 4% of all unique entities in the corpus. This shows that the strategy of using seed set might also work in other datasets in different languages and domains.

4.3.3.2. Unambiguous evaluation for seen entities

The notion of unambiguity also motivates the evaluation on unseen entities. We assume that for seen entities, it is sufficient to memorize their labels from the training set to be able to correctly classify them in test time. In this case we compare labels of mentions between the training and development/test set. This time, we compare

the mentions in a case-sensitive manner. We simulate the real scenario of model prediction where the capitalization of a mention might change the prediction[2].

Figure 4.4 shows the proportion of seen entities, which is around 50% of the development set, and around 35% of the test set. Among those that are seen only 8-13% are ambiguous. This means, by just memorizing the labels in the train set, we can correctly classify around 90% of the seen entities, showing that ambiguity is not the main challenge in this dataset. Therefore, it is more informative to evaluate the system on its ability to classify unseen entities. Moreover, seeing how 50% or more of the development and test set are unseen in this corpus, having a model that performs well on those entities is crucial.

[2]The baseline of CoNLL 2003 NERC task was obtained by memorizing the most frequent label of mentions in the train set, and was also done in a case-sensitive manner, i.e. the capitalization of a mention can change its prediction result.

	F1	F1 unseen
Ma and Hovy (2016)	91.21	-
Luo et al. (2015)	91.20	-
Lample et al. (2016) (reproduction)	90.94 (90.62)	- (83.28)
Collobert et al. (2011) (reproduction)	89.59 (87.77)	- (78.15)
Florian et al. (2003)	88.76	79.66
Chieu and Ng (2002)	88.31	78.36
Klein et al. (2003)	86.31	75.34
Zhang and Johnson (2003)	85.50	73.64
Carreras et al. (2003a)	85.00	72.96
Curran and Clark (2003)	84.89	72.22
Mayfield et al. (2003)	84.67	72.07
Carreras et al. (2003b)	84.30	69.43
McCallum and Li (2003b)	84.04	71.27
Bender et al. (2003)	83.92	70.37
Munro et al. (2003)	82.50	68.00
Wu et al. (2003)	81.70	66.10
Whitelaw and Patrick (2003)	79.78	62.01
Hendrickx and van den Bosch (2003)	78.20	59.43
De Meulder and Daelemans (2003)	76.97	56.39
Hammerton (2003)	60.15	14.39
Our model		
ℓ_*	61.06	52.63
ℓ_* + gazetteers	78.51	60.26

Tab. 4.4.: Evaluation of NERC systems on CoNLL-2003 dataset.

	lex	pos	aff	pre	ort	gaz	chu	pat	cas	tri	bag	quo	doc	unl	cem	clu	kb
Ma and Hovy (2016)														+	+		
Luo et al. (2015)	+	+	+			+	+				+					+	+
Lample et al. (2016)														+	+		
Collobert et al. (2011)				+										+			
Florian et al. (2003)	+	+	+	+	+	+	+		+								
Chieu and Ng (2002)	+	+	+	+	+	+				+		+	+				
Klein et al. (2003)	+	+	+	+													
Zhang and Johnson (2003)	+	+	+	+	+	+	+										
Carreras et al. (2003a)	+	+	+	+	+	+	+	+		+	+						
Curran and Clark (2003)	+	+	+	+	+	+		+	+	+							
Mayfield et al. (2003)	+	+	+	+	+		+	+									
Carreras et al. (2003b)	+	+	+	+	+			+									
McCallum and Li (2003b)	+				+	+		+						+			
Bender et al. (2003)	+	+		+	+	+	+							+			
Munro et al. (2003)	+	+	+		+		+		+	+	+						
Wu et al. (2003)	+	+	+	+	+	+											
Whitelaw and Patrick (2003)			+	+					+								
Hendrickx and van den Bosch (2003)	+	+	+	+	+	+	+							+			
De Meulder and Daelemans (2003)	+	+	+		+	+	+		+					+			
Hammerton (2003)	+	+				+	+										
ℓ_*		+			+											+	
ℓ_*+*gaz*		+			+	+										+	

Tab. 4.5.: Features used by NERC systems on CoNLL-2003 dataset. Aff: affix information (n-grams); bag: bag of words; cas: global case information; cem: character embeddings; chu: chunk tags; clu: word clusters; doc: global document information; gaz: gazetteers; kb: knowledge base; lex: lexical features; ort: orthographic information; pat: orthographic patterns (like Aa0); pos: part-of-speech tags; pre:previously predicted NE tags; quo: flag signing that the word is between quotes; tri: trigger words; unl: unlabeled dataset .

4.3.4 Evaluation of Unseen Entities on State-of-The-Art Systems

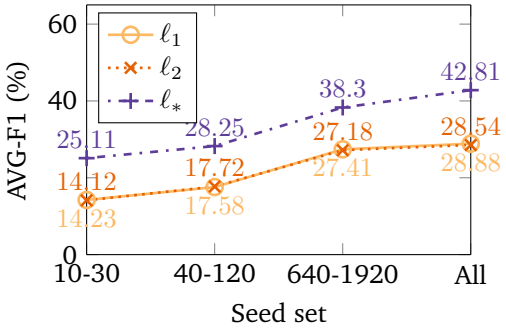
To see the performance of state-of-the-art systems in unseen entities, we re-evaluate their results as seen in Table 4.4. For Lample et al. (2016) and Collobert et al. (2011), we tried to reproduce the results using the tool[3] provided by the authors and obtained the F1 score as shown in brackets (i.e. 90.62 for Lample et al. (2016)). We were not able to reproduce results by the newest systems Luo et al. (2015) and Ma and Hovy (2016). We can see than evaluating on unseen entities causes the performances to drop at by least 10 percent points in F1 score, with the exception of the system by Lample et al. (2016) (their performance drops by 7 percent only). This suggests that part of the success of state-of-the-art models is in storing known entities together with their type (in the form of gazetteers or directly in lexicalized parameters of the model), and that named entity classification for *unseen* entities is still a challenging problem.

We list the different features that each system used in Table 4.5. Some approaches that has been used to help identify unseen entities include using gazetteers, exploiting similar structures of entity mentions and using labeled or unlabeled external textual resources, as described below.

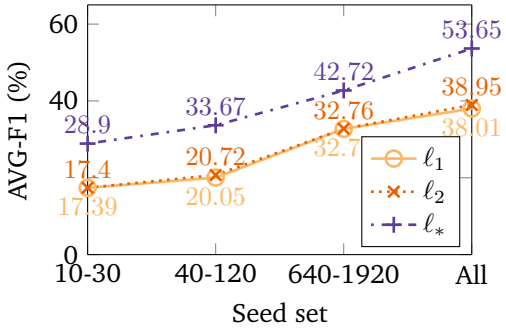
4.3.4.1. Strategies for unseen entities in SOTA

Gazetteers One of the most popular way of handling unseen entities is to use gazetteers, as done by most systems participated in CoNLL 2003 Shared Task. The list of known entities in a gazetteer can be helpful when its covers unseen entities. In Florian et al. (2003), gazetteers reduces the F-measure error by a factor of 15 to 21%. Unfortunately, in practice, gazetteers are not always available. Moreover, creating ones that are meaningful for the task is not always trivial. For example, De Meulder and Daelemans (2003) observed that their general-purpose gazetteers, taken partially from the internet and partially hand-crafted, are ineffective in increasing performance. They needed to create new gazetteers that are specific to the dataset to increase the performance. Hendrickx and van den Bosch (2003) also have a similar observation, incorporating gazetteers in their model help only in some of the different test sets that they have. In addition, for entity types in a very dynamic domain–e.g. electronic consumer goods, movies, etc– where there can be new names of entities every month, a system that heavily relies on gazetteer will not be able to recognize such new names unless the gazetteers are maintained continuously.

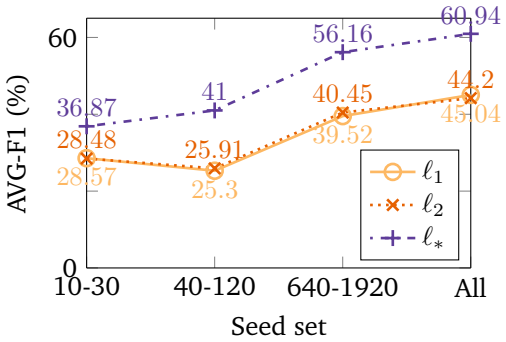
[3]Lample et al. (2016): https://github.com/clab/stack-lstm-ner, Collobert et al. (2011): https://ronan.collobert.com/senna/



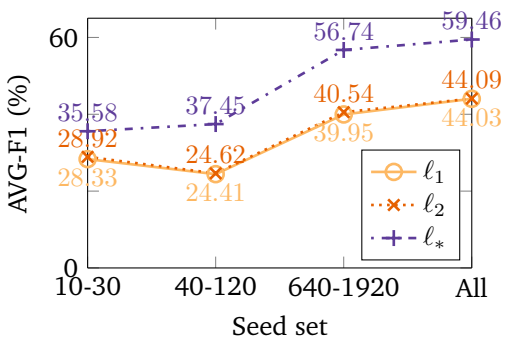
a: Only full conjunctions of left-right contexts (cluster), window size = 1



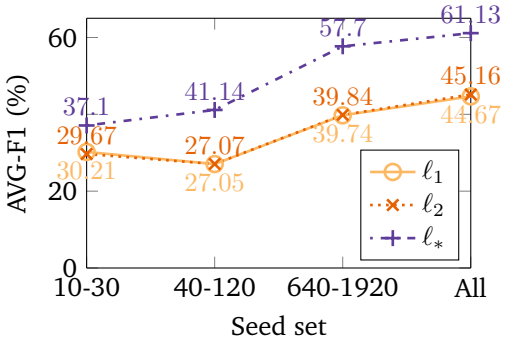
b: Only full conjunctions of entity tags and left-right contexts (cluster), window size = 1



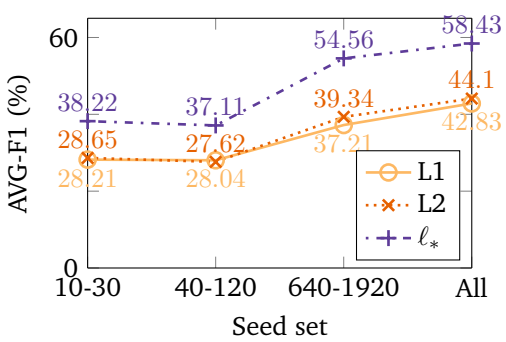
c: Elementary features and all conjunctions of entity tags and left-right contexts (cluster), window size = 1



d: Elementary features and all conjunctions of entity tags and left-right contexts (cluster), window size = 2



e: Elementary features and all conjunctions of entity tags and left-right contexts (cluster & PoS), window size = 1



f: Elementary features and all conjunctions of entity tags and left-right contexts (cluster & PoS), window size = 2

Fig. 4.5.: Average F1 of classification of unseen entity candidates on development data, with respect to the size of the seed. ℓ_1, ℓ_2 and ℓ_* refers to models with different regularizations. Each plot corresponds to a different conjunctive feature space with respect to window size (1 or 2), context representation (cluster with/out PoS), using entity features or not, and combining or not full conjunctions with lower-order conjunctions and elementary features.

Entity mention structure similarities Another popular approach to generalize to unseen entities is by exploiting similar structure of entity names in certain domains. For example, Munro et al. (2003) observed that bi-gram 'ae' and 'gg'–in CoNLL 2003 training set and more generaly within English–almost always indicates a named entity, due to etymological history of entity names in the language. Such characteristics can be utilized to extract unseen entities with similar character n-grams, prefix or suffix with seen ones. Most systems listed in Table 4.4–except McCallum and Li (2003b) and Hammerton (2003)–incorporates such features in their systems. Such features relies on the assumption that structure similarities exist between entities in the domain. One cannot always assume names in each entity type to share etymological history or other constraints that lead to such structure regularities.

External labeled textual resources Some systems, such as Florian et al. (2003) and Luo et al. (2015) also uses external textual resources that are labeled with the same or similar named entities. Such resources can be used, for example, to train other systems whose output can help the final classification, to expand the training set, or to calculate important statistics as features of the model. An example of commonly used labeled textual resources are Wikipedia articles, in which named entity mentions are tagged by hyperlinks to their respective entities.

External unlabeled textual resources More recent systems are using external unlabeled textual resources to create representation of words, such as Ma and Hovy (2016) and Lample et al. (2016). A large corpus is typically required to obtain an reliable independent representation of words. Therefore, it is common to use pre-trained representations or embeddings of words, which were trained using an external corpus that are available in large quantity and do not necessarily related to the corpus that we will train and test our models.

4.3.5 Comparing Regularizers

In this section, we present a comparison of models using the nuclear norm regularizer with baselines models. To examine the efficacy of such regularizer, we use standard regularizers of ℓ_1 and ℓ_2 for our baseline models. In all of the models (proposed and baselines), we use the same feature sets and simply change the regularization penalty.

To train each model, we validate the regularization parameter and the number of iterations on development data, trying a wide range of values. The best performing configuration is then used for the comparison.

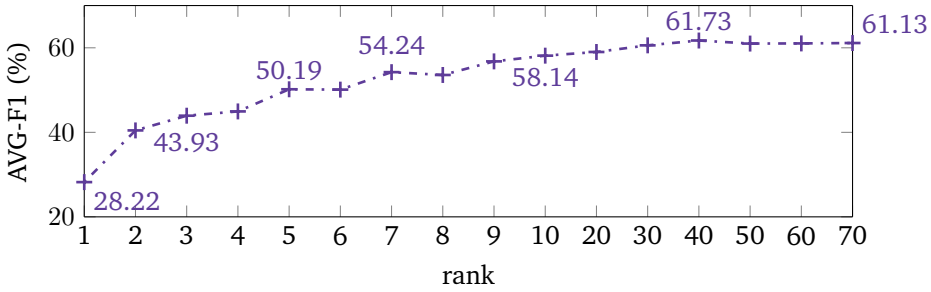


Fig. 4.6.: Avg. F1 on development set for increasing dimensions, using the low-rank model in Figure 4.5e trained with **all** seeds.

4.3.5.1. Results

Figure 4.5 shows results on the development set for different feature sets, including contexts in the form bag of of word cluster and Part of Speech (tags) of the words surrounding the named entity mentions, and orthological information of the mentions. We started representing context using cluster labels, as it is the most compact representation obtaining good results in preliminary experiments. We tried several conjunctions: a conjunction of the left and right context, as well as conjunctions of left and right contexts and features of the candidate entity. We also tried all different conjunction combinations of the contexts and the candidate entity features, as well as adding PoS tags to represent contexts.

We observe that for most conjunction settings our regularizer performs better than the ℓ_1 and ℓ_2 regularizers. Moreover, in many cases, our model is able to achieve a comparable result with ℓ_1 and ℓ_2 using only 10% of the seed set. This is very encouraging for a minimal supervision setting; we can train competitive models with much less annotation effort.

Using the best model from each regularizer, we evaluated on the test set. As can be seen in Figure 4.5, the best model we obtained uses all training data, with features of context in the form of word cluster and PoS tags (window size of 1 to the left and to the right), as well as ortographic information (see Figure 4.2) of the named entity mentions. The model was trained with learning rate value $\eta = 100$ and regularization factor $\tau = 20$, which was chosen by trying different value combinations (ranging from 0.0001 to 1000) and evaluating them on the development set. Table 4.6 shows the test results of the model. For all seed sets, the nuclear norm regularizer obtains the best average F1 performance. This shows that encouraging the max-entropy model to operate on a low-dimensional space is effective.

Moreover, Figure 4.6 shows model performance as a function of the rank. The model obtains a good performance even if only a few intrinsic dimensions are used.

		PER			LOC			ORG			MISC			AVG F1
		P	R	F1	P	R	F1	P	R	F1	P	R	F1	
10-30	ℓ_1	66	65	66	**15**	**24**	**19**	59	19	**29**	23	30	26	35
	ℓ_1	66	65	65	15	23	18	**61**	19	28	23	31	26	35
	ℓ_*	**72**	**75**	**74**	15	22	18	49	**21**	18	**31**	**38**	**34**	**39**
40-120	ℓ_1	72	44	55	13	40	20	49	31	38	22	36	27	35
	ℓ_2	72	45	55	**14**	**42**	**21**	**49**	32	38	22	31	25	35
	ℓ_*	**75**	**61**	**68**	13	21	16	49	**36**	**41**	**30**	**47**	**37**	**40**
640-1920	ℓ_1	80	62	70	24	**44**	31	56	48	51	20	30	24	44
	ℓ_2	79	66	71	27	44	33	**60**	49	**54**	22	32	26	46
	ℓ_*	**81**	**81**	**81**	**52**	**44**	**48**	54	**54**	54	**29**	**51**	**37**	**55**
All	ℓ_1	76	72	74	33	36	34	57	46	51	28	29	29	47
	ℓ_2	**77**	71	74	34	**37**	36	58	**50**	54	29	33	31	48
	ℓ_*	74	**91**	**81**	**65**	36	**46**	**72**	45	**55**	**37**	**44**	**40**	**56**

Tab. 4.6.: Results (**P**recision/**R**ecall/**F1**) on the test set for models trained with different sizes of the seed, using the parameters and features that obtain the best evaluation results on the development set. ℓ_1, ℓ_2 and ℓ_* refers to different regularizations. Only test entities unseen at training are considered. Avg. F1 is over PER, LOC, ORG and MISC, excluding O.

4.3.5.2. Analysis of weight of unseen conjunctions

To see the effect of low-rank regularization in giving weights to unseen conjunctions, we examined the parameter matrix of the low-rank model in Figure 4.5f with the **10-30** seed. We show in Figure 4.7 the proportions of zero and non-zero valued parameters in the parameter matrix of the resulting model. Based on the conjunctions seen in the training data, only around 1% of parameter matrix will have non-zero values, but in the final model, 27% of the matrix have non-zero values. This illustrates the ability of the low-rank regularization to give weights to unseen conjunctions. We emphasize that using ℓ_1 or ℓ_2 regularization can not set non-zero weights to these parameters associated with unseen conjunctions.

4.3.6 Comparison with State-of-The-Art Systems

In the previous section we compare the use of low-rank regularization with the standard ℓ_1 and ℓ_2 regularizer. In this section, we want to compare our low-rank regularization model with state-of-the-art NERC models.

Since our model is a classification model, we assume the availability of pre-identified boundaries of entity candidates. Nevertheless, we also add some noise in addition

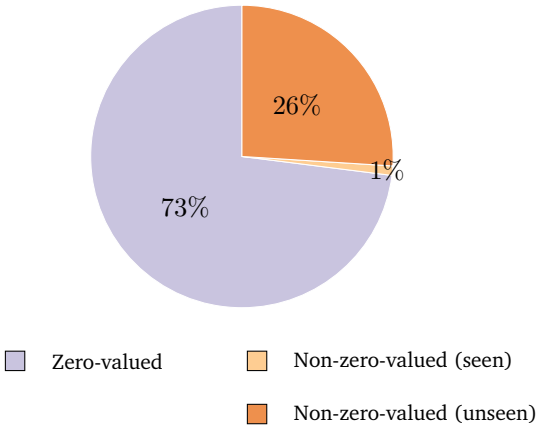


Fig. 4.7.: Weights of conjunctive features in the parameter matrix of Figure 4.5f trained with the **10-30** seed.

to the correct boundaries. The noise is added by also selecting single word nouns that are not a part of the correct boundaries. Note that this is the same candidate selection strategy we use for our training data. We compare both the standard evaluation as well as the unseen evaluation in Table 4.4. Our model uses only simple features of the left and right context (cluster & PoS) with window size of 1 and the simple entity mention features as described in Section 4.3.2 and Table 4.5. In this experiment we use all training data to be comparable with state-of-the-art settings.

We also experimented with using gazetteers as a post-processing step to our model results. We change labels of candidates to match those found in gazetteers. We compare the candidates and the gazetteer items by case-sensitive exact match. We use only the official gazetteer provided for the CoNLL 2003 Shared Task and no other external resources. We can see that the use of gazetteer improves our results significantly (up to 18% F1 score in standard evaluation and up to almost 10% F1 score in unseen evaluation).

We can see that the results of our model is quite far from the best state-of-the-art results. It is important to note, nevertheless, that our model uses a very simple set of elementary features compared to most state-of-the-art systems, as seen in Table 4.5. It is evident that there are many ways of improving the performance, including enriching the feature set, using better gazetteer and exploiting structure similarities of the entity mentions. The gazeetteer we used is a very limited set of named entity list that were found to be unhelpful in some state-of-the-art systems. We can improve the gazeetteer by expanding it with lists from other labeled external resources. As noted before, entity mentions in this corpus possess some structure similarities and thus the use of character ngrams, as used by state-of-the-art systems, can be helpful to improve the performance.

Another aspect to note is that our model does not do a joint NERC, instead, we assume the entities are recognized in advance. Most systems in Table 4.4 do a joint NERC, which may also help in classifying the named entities. Similar to ours, Carreras et al. (2003b) did a separate NER and NEC. Assuming a perfect recognition of named entities, they reported an accuracy of 95.14% in the development set. To compare, we also did an experiment where we train an NEC classifier without the "other" class, and obtained an accuracy of 70.96%. Adding gazetteer post-processing, we obtained 82.33% accuracy. Although this result is far from state-of-the-art performance, it is again important to note that Carreras et al. (2003b) uses much richer elementary features as well as gazetteers to train the system.

The use of dense representations such as word and character embeddings have been recently proven to be effective for this task, such as shown by Ma and Hovy (2016) and Lample et al. (2016). Ma and Hovy (2016) also analyze the performance on their system on several sets of unseen entities: those not in training but in word embeddings, those in training but not in word embeddings, and those that are in neither, and showed that their approach is useful in all of those cases. Therefore, utilizing such dense features, especially combining them with discrete sparse features where unseen conjunctions are present, would be an interesting future work. Such setting would lead to more challenges such as the increase size of the feature space, and thus a more efficient implementation strategy is required to run the experiments.

Even with the simple features and gazetteers, we are able to get better performances compared to some systems that use better features and resources. An interesting future work is to build a complete NERC system and try to use the same feature sets and external resources as state-of-the-art methods to do a fair comparison. With our method, it should be possible to use all possible elementary features (as much as the memory limitation allows) and have the learning method do the feature selection automatically, thus do not require an extensive feature engineering.

4.4 Conclusion

In this chapter, we apply our low-rank regularization model for the task of Named Entity classification.

We specify different aspects of the model that apply to this task. We presented the conjunctive feature spaces and their corresponding scoring function. We have also described the strategy of minimal supervision that can be employed for NE classification.

Our experiments were performed on CoNLL-2003 data and some analysis regarding the ambiguity of the data was presented in the chapter. It was shown that for this dataset, the more crucial problem is on classifying unseen entities.

We showed how the performance of state-of-the-art NERC systems drops significantly when evaluated on unseen entities. This suggests that classifying unseen entities remains a challenge.

Finally, we showed that our low-rank regularization method works better for sparse conjunctive feature spaces. The model showed encouraging results to train competitive models using less supervision. Our models were also shown to be able to obtain a good performance using only a few intrinsic dimension (i.e. rank). Moreover, we also visualized how our method is able to give weights to unseen conjunctions in the parameter matrix.

With simple elementary feature sets we use in our experiments, we obtain significantly inferior results compared to current state-of-the-art results on the same dataset. We identified some differences of feature sets and external resources of our current models with state-of-the-art models. An interesting future work is to use a more comparable feature sets and external resources to have a more fair comparison with state-of-the-art results. Some notable aspects we explore in this thesis is the use of convex method for low-rank constraint and the focus on small datasets, which we have not seen very much explored in state-of-the-art systems.

5 Low Rank Regularization for Template-Filling

In this chapter, we apply our low-rank model on another task of Information Extraction (IE), namely template filling, using similar settings to our NE classification task. The particular instance of template filling task we consider adds extra challenges compared to our NE classification task. First, classifications of mentions in this task introduce a new notion of ambiguity; mentions of the same entity are not always a slot-filler. Second, the type of data annotation we examine presents an additional challenge in compiling the training set. The annotations provided for this data are at the document level, while the supervision our method expects is at the mention level. Finally, in contrast to the NE classification setting, we do not assume that candidate slot-fillers are given.

Similar to our approach in NE classification, we employ a "seed" set for minimal supervision and focus on classifying unseen slot-fillers. We again empirically compare the performance of our model with standard ℓ_2 regularizers.

The organization of this chapter is as follows. We define the task of template filling in Section 5.1 and describe the strategy to train the slotfiller classifiers in Section 5.2. Section 5.3 reports the experimental results, and we conclude in Section 5.4.

5.1 Introduction

In this thesis, the term template filling refers to extracting text spans from a document that helps describe an event. We consider an event to have a pre-specified template of slots that represent different types of information that are relevant for the event. Template filling is useful for language understanding by representing the information of an article in a structured manner.

For example, one can devise a system to understand news of terrorism-related events, where we define templates that represent events such as *bombing* and *murder*. We can define the template *bombing* as having slots that describe the *perpetrator*, the *number of victims* and the *location* of the bombing. Meanwhile for the event of *murder*, the template might have a slot to describe the *weapon* being used. Another example is an automated airline customer service system where we can have events of *booking a flight* with slots such as *origin*, *destination* and *date of departure*.

An extension of template filling task is to also induce the templates from a textual corpus, such as done in Chambers and Jurafsky (2011). In this thesis, the templates

are predefined; we know the type of slots we are interested in, and the task is to identify entities relevant to the slots. Note that the term entity in this chapter does not always refer to a named entity, but rather a more general notion of an object. For example, mentions such as "bomb" or "gun" can be extracted for the slot *weapon*. In contrast with the NE classification task, this time we do not assume that the candidate slot-filler mentions are given or pre-identified by other systems.

This task of template filling is similar to slot filling task of Knowledge Base Population (KBP), as mentioned in Section 2.3.2. In both tasks, the goal is to extract slot-fillers for predefined slots of information from unlabeled text, which implies identifying candidate slot-fillers and determine their relationship (slot-type) with either a KB item or an event. The difference is that in the task of template filling, we can only use local contexts of the slot-filler to make decisions about the particular mention of the slot-filler, meanwhile KBP can use information from any text resources or even from the KB itself. While many KBP slot filling tasks are framed as relation extraction problem, event template completion are typically framed as a sequence labelling problem. Moreover, slot filling in KBP implies a sub-task of NEL as well, which is not the case for the template filling task we consider.

Similar to the NE classification task, the notion of classification in this task is more than classifying an identified slot-filler into its correct class, rather, the model needs to decide on two types of classification: (i) whether a mention is a slot-filler, and if so, (ii) of which slot type. Compared to the NE classification task, the particular instance of template filling task we consider introduces two additional challenges: contextual extraction and document-level supervision.

Contextual extraction In our NE classification, we simulate "noisy" NE recognizer –whose extraction might include non-entities– by adding non-entity mentions. In the task of template filling, such task is inherent. Even in the case of non-ambiguous slot types, mentions of the same entity are not always a slot-filler. For example, one individual might be mentioned in two news article, but only mentioned as a victim of an event in one of them. Such extraction thus relies heavily on the context of the mention. Therefore, unlike in NE classification, memorizing non-ambiguous seen entities is insufficient to classify the candidate mentions.

Document-level supervision We assume the corpus to have a document-level supervision rather than mention-level supervision. That is, we are given the text and the summary of events described in the text–in the form of complete event templates. This is to contrast with a mention-level supervision where slot-filler mentions are marked in the text, such as the type of supervision in the NE classification task. A document-level of supervision presents a challenge on building the training set because our model requires the training set to be in the form of mention-class pairs, which are not available when the mentions are not identified.

Entity-level evaluation Another difference with the NE classification task is on the evaluation setting. For this task, we consider an entity-level evaluation. That is, we evaluate the ability of the model to identify the correct entities for each slot, regardless of which mention of the entities was used for extracting them. So rather than evaluating the classification result of each mentions of the same entity, we only evaluate whether the model is able to classify at least one of them as a filler for the correct slot.

5.1.1 Contributions

The specific contributions of this chapter are as follow:

- We describe a specific instance of our low-rank model for the task of template filling.
- We present a strategy to reduce the noise of training sets that are built using document-level supervision. The strategy is based on a simple heuristics that takes into account distances between possible mentions of slot-fillers.
- We explore the MUC-4 dataset and the respective experimental settings in an attempt to provide reproducible results for the task of template filling on MUC-4 dataset. We also present details of the evaluation procedure to allow for an objective comparison for future studies.
- We conduct experiments to use our regularization framework and compare it to standard ℓ_2 regularizations for template-filling task.

5.2 Low-Rank Template Filling Model with Document-Level Supervision

In this section, we describe a specific implementation of the low-rank regularization model for template-filling task. Specifically, we present a strategy to build training data from document-level supervision to train the model.

The supervision we consider for this task is a document-level supervision such as used in the template filling task of the Message Understanding Conference (MUC-4) (Sundheim, 1992) dataset. In contrast with the mention-level supervision of the NE classification task, we do not have the documents labeled with occurrences of slot-fillers. Instead, for every document, we are given set of templates that correspond to events described in the document. The slots of the templates, in this case, are filled with their respective slot-fillers from the text.

We are interested to work with such type of supervision because we consider it to be more practical –thus more realistic to expect– in real-life scenarios. While this difference in supervision might not make a difference when the text is short or

when the slot-fillers are only mentioned once in the text, in long texts with multiple mentions of the slot-filler, annotating the mentions in the text will require more effort. One can imagine that it is much more straightforward to require annotators to answer questions such as "who are the victims?" rather than asking them to mark the exact occurrences of the mention in the text that describes the individuals as the victim. In some texts, the individuals might be mentioned multiple times, for which case the annotators should carefully select and agree on which of the mentions are indicative of them being victims.

In our model, we consider as input a list of (mention, label) pairs, in which a mention implies its location in the text is known and thus we have information about its context. With such supervision setting we assume for this task, the mentions need to be first identified within the text. In this case, one can consider a simple string matching and extract all matches in the text to be positive examples. Such simple rule, however, might lead to noisy training data. For example, consider a mention of a person's name, e.g. "John Doe", and a template containing slots for *weapon* and *victim*. Although as a slot-filler the word "John Doe" might be unambiguous (i.e. it will always be a victim and never a weapon), in a document containing multiple events, it can be mentioned as a victim of one event but only as a witness in the other events. Extracting all the mentions as positive examples will then lead to the model learning the wrong contexts. Note that such problem does not occur for our task of NE classifier with minimal supervision–we can safely assume "John Doe" will always refer to a named entity of type *person* regardless of the role it plays in an event.

We propose to reduce the noise by employing a simple heuristic that considers other slot-fillers in proximity. When there are several mentions of a slot-filler in a text, we choose ones that are closer to other possible slot-filler mentions of the same event. We assume that descriptions of an event are usually in the same part of the document. When there is more than one event, each event –hence slot-fillers– is usually described in different parts of the text.

For example, for an event containing two slot-fillers A and B, each having several possible mentions in the document; $\{a_1, \ldots, a_{|a|}\}$ and $\{b_1, \ldots, b_{|b|}\}$ respectively. Instead of adding all mentions a_i and b_j to the training set, we add only one mention of A and one mention of B. Let us number all words in a document in order of their position from the start of the document, i.e. first word in the document is numbered 1, second word 2, etc. A *location* of a phrase or mention is then defined as the position number of the first word of the phrase. We define $\delta(a_i, b_j)$ to be the distance between locations of mention a_i and b_j. Choosing the mentions for training is then finding a_i and b_j that minimizes δ:

$$\underset{a_i, b_j}{\operatorname{argmin}}\, \delta(a_i, b_j) \quad , \tag{5.1}$$

where $1 \leq i \leq |a|$ and $1 \leq j \leq |b|$. For events with more than two slot-fillers, we consider the sum of distances δ between all pairs of slot-fillers of the event.

5.3 Experiments

In this section, we describe our experiments of using low-rank models on the task of template filling with high-dimensional conjunctive feature space. We run experiments on learning classifiers with minimal supervision and focus the evaluation on unseen slot-fillers. We perform our experiment using MUC-4 data (Sundheim, 1992) and compare the performance with standard ℓ_2 regularizers.

5.3.1 Task Definition and Evaluation Metric

We consider the task of extracting slot-fillers from a document given few examples of known slot-fillers in other documents. Given a document, the goal is to propose appropriate strings that refer to entities that play certain roles in the events described in the document. In other words, the goal is to know the participants of an event and their roles in the event. Therefore, in contrast to the task of NE classification, the expected result is not an annotated test document, but rather a list of participants and their roles in the event. In fact, in this task, it does not matter if we are able to identify either one mention or several mentions of a participant in the document; extractions of multiple mentions of the same participant for the same role is considered as one extraction.

Supervision The training supervision is a document-level annotation, i.e. the location of the slot-filler is not marked in the text, as described in Section 5.2. During test time, we are given unlabeled documents with no pre-identified candidate slot-fillers.

5.3.1.1. Evaluation

Following state-of-the-art systems, the evaluation metric used is precision/recall/F1 score on all target slots in the test set. In addition, we also evaluate the model on unseen slot-fillers, which are new slot-fillers that have not been observed in train. As mentioned before, the evaluation setting assumed for this task is an entity-level evaluation. We consider each slot-filler to be an entity, which can be mentioned multiple times in a document. When one or multiple mentions of the entity is classified as a slot-filler for certain slot type y, we consider it as one extraction of the entity for slot y. We then evaluate on the ability of the model to extract the correct entities for each of the pre-specified slots by using the standard precision, recall and F1 of the extraction result.

In general, we follow the evaluation setting of Huang and Riloff (2012); if the role filler has the correct label and appears in any event template for the document, then it is correct. We describe some details of the evaluation procedure below.

Head noun matching rule In line with state-of-the-art systems, we use the head noun rule to judge if a candidate slot-filler matches the correct slot-filler as specified in the template's *key*–list of actual slot-fillers we need to extract. Instead of requiring the model to extract the exact match of a slot-filler, we consider the predicted and the actual slot-filler as a match when their head nouns match. The head noun is defined as the right-most word in the phrase or, in the case of the phrase containing the word "of", the right-most word before the word "of", e.g. "captain" for "the captain of the team".

Rules on multiple slot fillers A slot can be described as having several slot-fillers, and each slot-filler can have several options of mention phrases that refer to the same entity. The model is required to extract all slot-fillers, but not necessarily all mention phrases. One extraction that matches one of the mention phrase options–based on the head noun–is sufficient to be considered a successful extraction of the slot-filler.

Evaluating optional slot-fillers A document might have optional events or optional slot-fillers. We consider all slot-fillers in an optional event to be optional. We evaluate optional slot-filler by including the correctly predicted ones, but not penalizing those that are not extracted. That is, when a model correctly extract an optional slot-filler, both precision and recall will improve, but when the model is unable to extract an optional slot-filler, the recall will not be affected.

5.3.2 Data and Setting

We use MUC-4 data(Sundheim, 1992) that contains newswire articles about terrorism events in Latin America. An example of the data is shown in Figure 5.1[1]. The dataset contains a set of event templates, set of labeled templates, as well as their mappings to the respective text. We do some data preprocessing to the text, such as sentence splitting, tokenization, parsing (using Stanford CoreNLP (Manning et al., 2014) and SENNA (Collobert, 2011; Collobert et al., 2011)) and word sense disambiguation (using *It Makes Sense (IMS)* (Zhong and Ng, 2010)).

One document might be labeled with zero templates (i.e. do not describe an event), one template or more than one template, not necessarily of equal types. The training corpus (DEV) contains 1300 documents, but only 733 documents are labeled with at least one event. Documents with no event are articles such as speeches or news related to political events that do not involve a terrorist attack. The development (TST1+TST2) and test (TST3+TST4) sets each contain 200 documents.

[1]Visualization of the annotation is taken from `https://github.com/brendano/muc4_proc`

DEV-MUC3-0047

ACCORDING TO MILITARY SOURCES
AND PEOPLE WHO ARRIVED TODAY IN AYACUCHO FROM THE LA MAR AREA, 44
PEASANTS HAVE BEEN KILLED IN TWO TOWNSHIPS OF LA MAR PROVINCE,
AYACUCHO DEPARTMENT.

YESTERDAY A [[COLUMN OF [[SHINING PATH]$_1$]$_2$ TERRORISTS]$_1$]$_2$ ARRIVED IN THE VILLAGE OF CHINCHIPE, IN THE JUNGLE PROVINCE OF LA MAR, AND SHOT 16 PEASANTS WHO WERE MEMBERS OF THE PEASANT PATROLS THAT OPPOSE THE TERRORISTS.

THE [[[[SHINING PATH]$_1$]$_2$ GUERRILLAS]$_1$]$_2$, WHO BURNED MURDERED PEASANTS' [HOUSES]$_2$, CHARGED THEM WITH COLLABORATING WITH THE ARMY. THE TOWN OF CHINCHIPE IS 220 KM NORTH OF AYACUCHO.

TRAVELLERS ARRIVING FROM LA MAR PROVINCE SAID THAT ON 23 SEPTEMBER 28 BODIES OF ALLEGED PEASANTS WERE FOUND NEAR THE TOWN OF CHULLAS.

THE BODIES SHOWED SIGNS OF [TORTURE]$_3$ AND BULLET WOUNDS BUT THE IDENTITY OF THE [MURDERERS]$_3$ APPARENTLY COULD NOT BE DETERMINED.

a: Document *DEV-MUC3-0047* from MUC-4 dataset, annotated with phrases that matches a slot-filler in the respective event templates.

message id	"DEV-MUC3-0047"
message template	"1"
incident type	"ATTACK"
incident instrument_id	"-"
perp individual_id	["COLUMN OF SHINING PATH TERRORISTS", "SHINING PATH GUERRILLAS"]
perp organization_id	["SHINING PATH"]
phys_tgt id	"-"
hum_tgt name	"-"

message id	"DEV-MUC3-0047"
message template	"2"
incident type	"ARSON"
incident instrument_id	"*"
perp individual_id	["COLUMN OF SHINING PATH TERRORISTS", "SHINING PATH GUERRILLAS"]
perp organization_id	["SHINING PATH"]
phys_tgt id	["HOUSES"]
hum_tgt name	"-"

message id	"DEV-MUC3-0047"
message template	"3"
incident type	"ATTACK"
incident instrument_id	["TORTURE"]
perp individual_id	["MURDERERS"]
perp organization_id	"-"
phys_tgt id	"-"
hum_tgt name	"-"

b: The event templates of document *DEV-MUC3-0047* from MUC-4 dataset.

Fig. 5.1.: Document *DEV-MUC3-0047* and its corresponding event templates from MUC-4 dataset.

PerpInd	PERP: INDIVIDUAL ID
PerpOrg	PERP: ORGANIZATION ID
Victim	HUM TGT: DESCRIPTION (without colon clauses) HUM TGT: NAME
Target	PHYS TGT: ID
Weapon	INCIDENT: INSTRUMENT ID

Fig. 5.2.: List of slot types used in this thesis and their respective slot names in MUC-4 keys.

5.3.2.1. Details of MUC-4 Data

Due to the complexity of the dataset, state-of-the-art systems that were evaluated with the same dataset have used various configurations of the same data for their experiments. Many time, details in configuration choices are not well described in the literature, making it difficult to do an objective comparison. To allow for easier comparison in the future, we try to present as much details of the settings we assume in our experiments.

Template and slot types The dataset defines six template types: *Attack*, *Kidnapping*, *Bombing*, *Arson*, *Robbery* and *Forced Work Stoppage*. Each template consists of 25 slot types. In this thesis, we will only consider events of *Attack*, *Kidnapping*, *Arson* and *Bombing*. For each event, we only consider the string-valued slots of: *PerpInd* (individual perpetrator), *PerpOrg* (organizational perpetrator), *Victim*, *Target* and *Weapon*. The respective template slot names for each type is shown in Figure 5.2.

Entities and mentions in key file In MUC-4 dataset, we consider a slot as having multiple entities separated by line break and an entity can have alternate mention strings separated by a forward slash "/" as one entity each line in the slot. For example, consider the following slot:

```
19. HUM TGT: DESCRIPTION            "JESUIT PRIESTS" / "JESUITS"
                                    "WOMEN"
```

In this case, we have two entities to fill the slot *Victim*: (i) "JESUIT PRIESTS" / "JESUITS" and (ii) "WOMEN". The first entity have two alternate mention strings: "JESUIT PRIESTS" and "JESUITS".

Optional templates and entities We assume as optional template those that are marked as "OPTIONAL" in the "MESSAGE: TEMPLATE" slot, such as:

```
1.  MESSAGE: TEMPLATE               2 (OPTIONAL)
```

We assume as optional entities those that are marked with a question mark prefix, such as:

```
9.  PERP: INDIVIDUAL ID              ? "MANUEL RODRIGUEZ PATRIOTIC FRONT" / "FPMR"
```

Handling of colon clauses Some slots in MUC-4 keys can contain colon separated clauses, such as:

```
18. HUM TGT: NAME                    "BERNARDETTE PARDO"
                                     "CARLOS CORRALES"
                                     "JORGE SAENZ"
19. HUM TGT: DESCRIPTION             "U.S. JOURNALIST": "BERNARDETTE PARDO"
                                     "CAMERAMAN": "CARLOS CORRALES"
                                     "MOVIE ACTOR": "JORGE SAENZ"
```

In our setting, we assume that such colon clauses in the "HUM TGT: DESCRIPTION" slot to be a description of the "HUM TGT: NAME" slot, and thus do not include them in either training nor evaluation.

5.3.2.2. Training Settings

We extract examples of slot-fillers using heuristics as mentioned in Section 5.2. For examples of non slot-fillers, we use Noun Phrases (NP) in the text that do not overlap with possible slot-filler mentions in the text. We extract NP based on the parse tree we obtain during preprocessing. We consider all overlapping NP as one example, thus resulting in a large set of non-slot-filler examples. In our experiments, we *undersample* the set of non-slot-filler examples to include only the most frequent ones in the training set.

5.3.2.3. Candidate slot-fillers

In this task, we do not have access to pre-identified candidate slot-fillers during test time. Therefore, a template filling system needs to also identify such candidates from the unlabeled test corpus. For our system, we simply extract all Noun Phrases (NP) from the text. We use SENNA to parse the text and obtain all NPs as candidate entities. This simple rule gives us a recall of around 96% in the test set.

5.3.3 MUC-4 Data Characteristics

In this section, we explore the characteristics of the slot-fillers in the MUC-4 dataset. Specifically we examine the ambiguity of the slot types and the mentions.

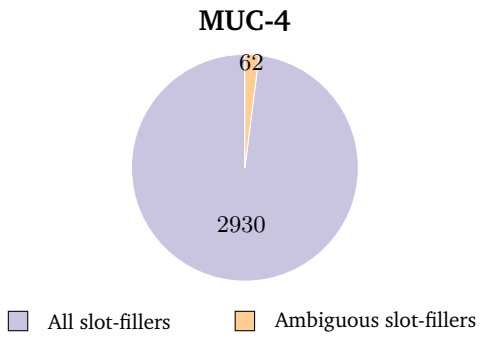


Fig. 5.3.: Proportion of ambiguous slot-filler mentions in MUC-4 dataset. The numbers corresponds to unique mentions of slot-fillers tagged as *PerpInd*, *PerpOrg*, *Victim*, *Target* or *Weapon* in the event template of *Attack*, *Kidnapping* and *Bombing*.

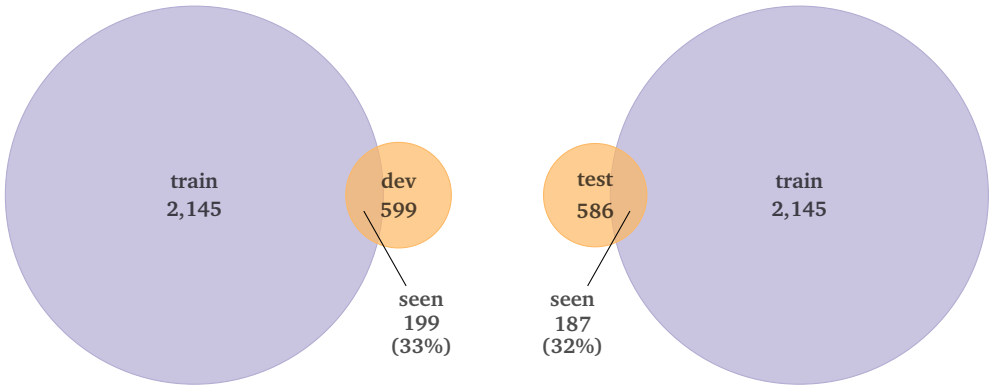


Fig. 5.4.: Proportion of seen/unseen slot-fillers in in the development and test set of MUC-4 data. The numbers corresponds to unique mentions of slot-fillers tagged as *PerpInd*, *PerpOrg*, *Victim*, *Target* or *Weapon* in the event template of *Attack*, *Kidnapping* and *Bombing*.

5.3.3.1. Slot types ambiguity

Figure 5.3 shows the ambiguity of slot-fillers in the train, development and test set of MUC-4 dataset. We consider slot-fillers from the key files–i.e. the annotated event templates–of the dataset and only consider slot types and templates as described in Section 5.3.2.1. We do not include optional slot-fillers. For the purpose of this comparison, we extract every alternative string of each slot filler as one instance of slot-filler. The idea is to see if a string mention is always classified as the same slot-type in different event templates. As we can see in Figure 5.3, only around 2% of slot-fillers are ambiguous in this dataset. Thus if the task is simply to classify pre-identified slot-fillers, the model can simply memorize the labels in training set. But in our case, we do not have access to pre-identified slot-fillers at test time, rather, we are trying to classify all Noun Phrases in the test set.

5.3.3.2. Proportion of unseen slot-fillers

Figure 5.4 shows the proportion of seen and unseen slot-fillers in the development and test set of MUC-4 dataset. Similar to the Figure 5.3, we also consider each alternative string of slot-fillers in the key files as one instance of slot-filler and do not include optional slot-fillers. Matching between the instances are measured by exact matches between the two strings. We can see that even without minimal supervision, more than 60% of slot-fillers in test set are unseen. Therefore, the ability of the model to generalize to such unseen slot-fillers is important.

5.3.4 Most Frequent Class Baseline

To see if memorizing labels of seen slot-filler is sufficient for the template filling task on MUC-4 data, we perform a simple baseline experiment that memorizes labels of seen slot-fillers.

For this experiment, we simply memorize all the mentions of slot-fillers in training[2] and their most frequent class. The Noun Phrases (NP) in the test set are then classified based on this dictionary. Those that have not been seen in training are not extracted as a slot-filler. The comparison of the mentions in train and test set is based on exact match of the full phrase. We experimented with the full corpus (without minimal supervision) and obtained an F1 score of **61%** with a low precision of 59%. This suggests that many of the predictions based on memorizing seen labels are incorrect despite the low ambiguity of mentions as seen in section 5.3.3.1. This shows that mentions that have been seen as a slot-filler in train do not necessarily occur as a slot-filler in other documents, implying the importance of context for the classification, using features or representations of the mention itself will be insufficient. Moreover, the proportion of unseen slot-fillers shown in Section 5.3.3.2 suggests that it is important for the model to generalize to new slot-fillers in order to improve performance.

5.3.5 Comparing Regularizers

We compare the performance of our low-rank regularized models with those of using ℓ_2 regularizers to see if low-rank regularization applies to another task other than named entity classification (Chapter 4).

We ran a small experiment with two training data: (i) **4-40**: using only 4 most frequent mentions of slot-fillers for each slot types and most frequent 40 non slot-filler mentions, which leads to a total of 11,691 training samples (the full training set we compiled contains 54,857 samples) and (ii) **All**: using all mentions of slot-fillers and randomly sampled 20,000 non-entity mentions. For these experiments, we use word embeddings as the elementary features. Specifically, we use GloVe (Pennington

[2]Including the optional slot-fillers.

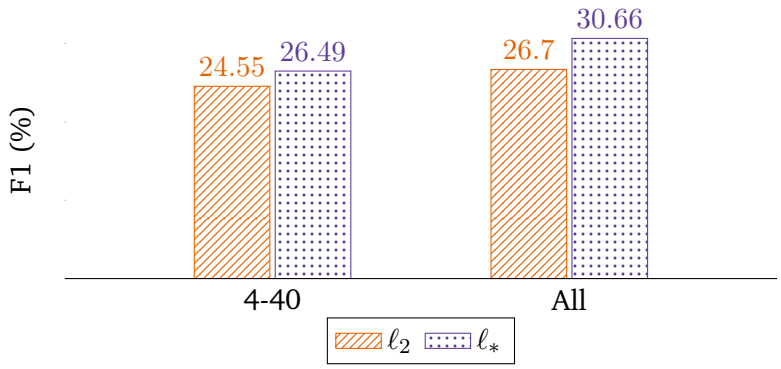


Fig. 5.5.: F1 score of unseen slot-fillers in in test set of MUC-4 data, using ℓ_2 and nuclear-norm (ℓ_*) regularization. **4-40**: models trained with only the top 4 mention examples of slot-fillers for each slot types and top 40 non slot-fillers mention examples. **All**: models trained with all mention examples of slot-fillers and uniformly sampled 20,000 non slot-filler examples.

et al., 2014) embeddings to represent the mention and contexts. The contexts we consider are words surrounding the mention with a window size of 3 (i.e. 3 words before and 3 words after the mention). Similar to (Boros et al., 2014), we combine word embeddings of multiple word phrases by performing an addition of each word's embedding.

Although in this case the feature spaces are no longer sparse, the comparison of ℓ_* regularization with standard ℓ_2 regularization is still interesting in this case, such as observed by Madhyastha et al. (2014). Moreover, the use of tensors of elementary features still creates a conjunctive feature space, for which it is interesting to see whether low-rank constraint is useful in such case.

The hyperparameters are tuned on the development set, and the maximum number of iterations are set to 100 for all models. The learning rate (η) was automatically tuned by doing a linesearch on each iteration. We experimented varying values of regularization factor, ranging from 0.0000001 to 2. The best ℓ_2 model is achieved using regularization factor (τ) of 0.1, and the best ℓ_* model is used a regularization factor of 1.

The results shown in Figure 5.5 are the result of classifying slot fillers in the test set using the two models. We can see from this results that the model is also helpful in this classification task.

5.3.6 Comparison with State-of-The-Art Systems

While it would be interesting to compare the performance of our model to state-of-the-art methods, we leave it for future work. In order to have a competitive results in slot-filling, we need to build a slot-filling system, which requires more than classifying noun phrases. Most slot-filling systems are either sequence models (such

	PerpInd	PerpOrg	Target	Victim	Weapon
Huang and Riloff (2012)	129	74	126	201	58
Ours (total)	174	126	152	242	69
Ours (non-optional)	122	87	111	186	63
Ours (optional)	52	39	41	56	6

Tab. 5.1.: Number of slot fillers in the MUC-4 Test Set.

as Liu and Lane (2016)) or modular systems with multiple classification steps (such as Huang and Riloff (2012)).

Moreover, unfortunately we have not been able to replicate the evaluation conditions on the MUC-4 dataset of systems in the the literature. This is essentially due to the nature of MUC-4 dataset being annotated in document-level and the lack of documentation in the evaluation for slot-filler extraction evaluation in this dataset, thus it is difficult to know for sure if the evaluations metric used by each state-of-the-art methods are the same. For example, Huang and Riloff (2012) reported the number of slot-fillers in MUC-4 test set, but do not provide descriptions in details what they consider as one slot-filler. In our case, we obtain the number of slot-fillers as in shown in Table 5.1. Such discrepancies can stem from ambiguity in assuming what constitutes as one slot-filler, for example, the following questions can be raised from the data (as described in Section 5.3.2.1):

- do we consider each alternate mention of an entity as one slot-filler?
- do we include optional slot-fillers?
- do we include slot-fillers in an optional template?
- how do we consider colon clauses?

Evaluation discrepancies for this dataset have also been uncovered before by O'Connor (2013). In our case, we describe our evaluation details in Section 5.3.1.1.

With such considerations, for this experiment, we do not build a system and leave comparison with state-of-the-art systems for future work. Instead we have focused in analyzing how nuclear-norm regularization compares to ℓ_2 regularization for unseen slot-fillers. For future work, we would like to utilize such regularization in a complete end-to-end system and experiment with a more well defined task and dataset to compare it with state-of-the-art systems.

5.4 Conclusion

In summary, in this chapter we applied our low-rank regularization model for the task of template filling. We explored a dense conjunctive feature space by using dense elementary feature sets than in NE.

We have also presented a heuristic rule to build a less noisy training set from the provided document-level supervision and unlabeled documents.

Finally, we compared our regularization model with ℓ_2 regularizer to explore the efficacy of low-rank regularization in dense conjunctive spaces. Our experiments are meant to obtain preliminary results to show how our methods is applicable to a different classification tasks. We use very limited training data, as well as features. We showed that the use of low-rank constraint is useful in comparison with ℓ_2 in this case as well.

Comparison with state-of-the-art methods is challenging for this dataset, as described in Section 5.3.2, due to evaluation discrepancies between state-of-the-art methods.

Conclusion and Future Work

6

In summary, this thesis investigates a **low-rank regularization** framework for **linear feature-based models** in **high-dimensional sparse conjunctive feature spaces**. We applied such method for two tasks of Information Extraction (IE): Named Entity Classification (NEC) and template filling. The focus is to examine the efficacy of such regularization to extract unseen entities/slot-fillers on settings where minimal supervision is used.

This chapter summarizes our contributions in that regard and presents possible areas of future work.

6.1 Contributions

In summary, our contributions are: (i) we construct a regularization scheme for training linear feature-based classifiers in high-dimensional sparse conjunctive feature spaces and (ii) we conduct experiments on learning entity classifiers and extracting slot-fillers, both with minimal supervision.

6.1.1 Development of Low-Rank Regularization Method for Sparse Conjunctive Feature Spaces

We have developed a low-rank regularization framework for linear feature-based models in sparse conjunctive feature spaces. Our formulation is based on using tensors to parameterize the classifiers, and then control the capacity of the model using low-rank regularization. We show that by imposing a low-rank penalty, we implicitly induce a low-dimensional embedding of the elementary feature vectors. Such formulation enables the method to give weights to unseen conjunctive features and allows the model to utilize them at test time. Overall, our formulation results in a convex procedure for training the model parameters. Even though the thesis focuses on specific IE tasks, the scheme can be easily adapted for other tasks.

6.1.2 Experiments on Named Entity Classification and Template Filling

We empirically examined the method for named entity classification and template filling tasks. We framed the tasks as a task of classifying text spans. For both cases, we assumed the setting of minimal supervision by using only a handful of examples for each entity/slot types. The training data was then compiled based on mentions

of the examples in the text. To evaluate the models, we introduced the use of an evaluation perspective that focuses on measuring the performance extracting entities/slot-filler candidates that are not provided during training.

Our experiments gave preliminary results to give light on the use of low-rank regularization in limited settings, which can be extended to a full NERC or template filling systems using richer feature spaces in future works. Our results showed that the proposed regularization is better for sparse conjunctive feature spaces than standard ℓ_1 or ℓ_2 regularization. It facilitates the extraction of unseen candidates by giving weights to unseen feature conjunctions, an ability not possessed by neither ℓ_1 nor ℓ_2. In some cases, this allows the low-rank regularized models to use fewer examples to achieve comparable results with ℓ_1 or ℓ_2 regularized models. For example, in the case of named entity classification, we can use less than 10% of seed examples to achieve comparable results using low-rank regularization.

These results make us conclude that encouraging the model to operate in a low-dimensional space is an effective way of controlling its capacity and ensure good generalization for high-dimensional sparse conjunctive feature spaces.

Unfortunately, we were not able to do more comprehensive experiments in large scale, such as using more elementary feature sets, nor were we able to build a complete end-to-end system to compare with state-of-the-art. More engineering effort is required, which was not feasible in the timeline of this thesis.

6.2 Future work

Throughout the research, we took note of several ideas that might improve our approach. We present some prominent ones below.

6.2.1 Incorporation of Unlabeled Data in Semi-Supervised learning

Our current model is trained in a supervised manner, i.e. values of parameters are learned solely based on annotated examples (features/label pairs) given during training. To improve the model, we would like to exploit information from unlabeled data by doing semi-supervised learning. Semi-supervised learning is useful whenever there are far more unlabeled data than labeled, which is likely to occur if obtaining data points is cheap, but obtaining the labels costs a lot of time, effort, or money (Chapelle et al., 2010). With a minimal supervision setting, such conditions are true, and thus it is worthwhile to investigate the use of semi-supervised learning for our case.

Semi-supervised learning vs supervised learning Technically, our models utilize unseen conjunctions that are not a part of the labeled training data. But it is important to note that this is not a semi-supervised learning. The values of these conjunctions depend only on the distribution of seen conjunctions from labeled data, thus makes it a supervised learning. Meanwhile, using semi-supervised learning, these values will also depend on the distributions of such conjunctions in the unlabeled data. We present below two examples of possible semi-supervised extensions to our learning method. In practice, we should also take a look at other semi-supervised approaches as well before deciding which is most suitable.

Self-training A simple and direct extension of our method is to use a semi-supervised approach called self-training. First, we train a classifier with a small amount of labeled data. We then use the classifier to classify unlabeled data, which are then added to the training set. One might also use some criteria to choose which newly labeled data should be included. The classifier is then retrained using the new training set, and then we repeat the steps. Self-training has been applied to different natural language processing tasks, such as Yarowsky (1995), Riloff et al. (2003), Maeireizo et al. (2004) and Ji and Grishman (2006).

Co-training Another possible approach is to use co-training (Blum and Mitchell, 1998; Mitchell, 1999). Other studies (Collins and Singer, 1999; Cucerzan and Yarowsky, 2002; Jones, 2005; Kim et al., 2002; Neelakantan and Collins, 2014b) have used co-training for the task of information extraction from text. The idea of co-training is to have two (or more) classifiers that "learn" from each other. Each classifier labels unlabeled data and adds the newly labeled data into the training set of the other classifier. The classifiers are then re-trained, and the process is repeated. One way of having different classifiers is to use different families of models, for example, Kim et al. (2002) combines Memory-based Learning, Sparse Network of Winnows and Maximum Entropy Model. Another option is to split the features into two (or more) sets. An important assumption with such method is that the feature sets are individually sufficient to train a good classifier and are conditionally independent given the class. Although in our setting we have separate elementary feature vectors that can be easily grouped into sub-sets, our method assumes that there are interactions between all of these elementary feature vectors. Splitting them into sub-groups will remove some of the interactions on the model. Therefore, the strategy of dividing these features into subsets that adhere to the co-training assumptions could, by itself, become an interesting research question.

6.2.2 Enforcing Low-Rank Tensor Regularization

In this thesis, we chose one tensor unfolding for each model and imposed a low-rank constraint on the matrix. An alternative direction would be to impose the constraint to the tensor form directly. In our current approach, the latent embeddings contain

a combined projection of several elementary feature vectors. Tensor decomposition would allow the projection of each elementary feature vectors into their latent representations. Therefore, it is compelling to investigate how such difference would affect the model. The use of tensor decomposition has been used in other IE related works such as Chang et al. (2014) and Nickel et al. (2012). Meanwhile, the use of low-rank tensor regularization has been employed in other fields of machine learning, such as Dong et al. (2014), which generalizes a low-rank matrix regularized method into a low-rank tensor regularized method for a computer vision application.

6.2.3 Combining sparse regularization and low-rank regularization

As shown in Hutchinson et al. (2012), the combination of a sparse regularization and a low-rank regularization could be beneficial. The idea is to keep two versions of the parameter matrix; the sparse version and the low-rank version. Using a low-rank regularization is useful in capturing regularities in training data, and thus good for generalizing the model. Imposing such constraint, however, removes the sparsity of the matrix. In some cases, it could be a good idea to keep the sparsity of the matrix to learn exceptions. For instance, we might want to memorize features with very strong discriminative quality. A very simple example is a feature that represents the full phrase of a mention. Since in our case mentions are unambiguous, we might want the model to simply memorize the classification of mentions it saw in the training set, regardless of the context. But at the same time, we would also like to learn the regularities of the context so we can classify mentions that are unseen during training. In this case, using a combination of the two matrix regularization could be beneficial to improve the model.

A

Low-Rank Regularization for Sparse Conjunctive Feature Spaces: An Application to Named Entity Classification

The following pages contain the paper Primadhanty et al. (2015) as published in ACL 2015. This paper is related to the main contents of Chapter 3 and 4.

InToEventS: An Interactive Toolkit for Discovering and Building Event Schemas

B

The following pages contain the paper Ferrero et al. (2017) as published in EACL 2017. This paper is related to –but not part of– the task of Chapter 5.

ATTENTION ¡
For reasons of copyright, pages 90 to 93 of the thesis, containing the texts mentioned above, should be consulted on the web pages of the editor

https://www.aclweb.org/portal/search/profileplus/InToEventS%3A%20An%20Interactive%20Toolkit%20for%20Discovering%20and%20Building%20Event%20Schemas

Bibliography

Heike Adel, Benjamin Roth, and Hinrich Schütze. Comparing convolutional neural networks to traditional models for slot filling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 828–838, San Diego, California, June 2016. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N16-1097`.

David Ahn. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8. Association for Computational Linguistics, 2006.

Enrique Alfonseca and Suresh Manandhar. Extending a lexical ontology by a combination of distributional semantics signatures. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, EKAW '02, pages 1–7, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44268-5. URL `http://dl.acm.org/citation.cfm?id=645362.650887`.

Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(Nov):1817–1853, 2005.

Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D Manning, Christopher Ré, Julie Tibshirani, Jean Y Wu, Sen Wu, and Ce Zhang. Stanford's 2014 slot filling systems. *TAC KBP*, 695, 2014.

Raphaël Bailly, Xavier Carreras, and Ariadna Quattoni. Unsupervised spectral learning of finite state transducers. In *Advances in Neural Information Processing Systems*, pages 800–808, 2013a.

Raphaël Bailly, Xavier Carreras Pérez, Franco M Luque, and Ariadna Julieta Quattoni. Unsupervised spectral learning of wcfg as low-rank matrix completion. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 624–635. Association for Computational Linguistics, 2013b.

Borja Balle and Mehryar Mohri. Spectral learning of general weighted automata via constrained matrix completion. In *NIPS*, pages 2168–2176, 2012.

Oliver Bender, Franz Josef Och, and Hermann Ney. Maximum entropy models for named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 148–151. Edmonton, Canada, 2003.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Daniel M. Bikel, Scott Miller, Richard Schwartz, and Ralph Weischedel. Nymble: A high-performance learning name-finder. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, pages 194–201, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics. doi: 10.3115/974557.974586. URL `http://dx.doi.org/10.3115/974557.974586`.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165, 2009.

David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4):77–84, 2012.

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

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA, 1998. ACM. ISBN 1-58113-057-0. doi: 10.1145/279943.279962. URL `http://doi.acm.org/10.1145/279943.279962`.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Conference on artificial intelligence*, number EPFL-CONF-192344, 2011.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795, 2013.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

Emanuela Boros, Romaric Besançon, Olivier Ferret, and Brigitte Grau. Event role extraction using domain-relevant word representations. In *EMNLP*, pages 1852–1857, 2014.

Andrew Borthwick, John Sterling, Eugene Agichtein, and Ralph Grishman. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *IN PROCEEDINGS OF THE SIXTH WORKSHOP ON VERY LARGE CORPORA*, pages 152–160, 1998.

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

Razvan Bunescu. Using encyclopedic knowledge for named entity disambiguation. In *In EACL*, pages 9–16, 2006.

Xavier Carreras and Lluís Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164. Association for Computational Linguistics, 2005.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. Learning a perceptron-based named entity chunker via online recognition feedback. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 156–159. Edmonton, Canada, 2003a.

Xavier Carreras, Lluís Màrquez, and Lluís Padró. A simple named entity extractor using adaboost. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 152–155. Edmonton, Canada, 2003b.

Nathanael Chambers. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, volume 13, pages 1797–1807, 2013.

Nathanael Chambers and Dan Jurafsky. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics, 2011.

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. Multi-relational latent semantic analysis. In *EMNLP*, pages 1602–1612, 2013.

Kai-Wei Chang, Scott Wen-tau Yih, Bishan Yang, and Chris Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, October 2014.

Olivier Chapelle, Bernhard Schlkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125, 9780262514125.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. Probabilistic frame induction. In *Proceedings of NAACL-HLT*, pages 837–846, 2013.

Hai Leong Chieu and Hwee Tou Ng. Named entity recognition: A maximum entropy approach using global information. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1072228.1072253. URL `http://dx.doi.org/10.3115/1072228.1072253`.

Michael Collins and Yoram Singer. Unsupervised models for named entity classification. In *Proceedings of the joint SIGDAT conference on empirical methods in natural language processing and very large corpora*, pages 100–110. Citeseer, 1999.

Ronan Collobert. Deep learning for efficient discriminative parsing. In *AISTATS*, volume 15, pages 224–232, 2011.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. *EMNLP-CoNLL 2007*, page 708, 2007.

Silviu Cucerzan and David Yarowsky. Language independent ner using a unified model of internal and contextual evidence. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, COLING-02, pages 1–4, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118853.1118860. URL `http://dx.doi.org/10.3115/1118853.1118860`.

James R. Curran and Stephen Clark. Language independent ner using a maximum entropy tagger. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 164–167. Edmonton, Canada, 2003.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *In EACL*, 2017.

Fien De Meulder and Walter Daelemans. Memory-based named entity recognition using unannotated data. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 208–211. Edmonton, Canada, 2003.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A Tomlin, et al. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *Proceedings of the 12th international conference on World Wide Web*, pages 178–186. ACM, 2003.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1, 2004.

Weisheng Dong, Guangming Shi, Xin Li, Yi Ma, and Feng Huang. Compressive sensing via nonlocal low-rank regularization. *IEEE Transactions on Image Processing*, 23(8):3618–3632, 2014.

John Duchi and Yoram Singer. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934, 2009.

Micha Elsner, Eugene Charniak, and Mark Johnson. Structured generative models for unsupervised named-entity clustering. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 164–172. Association for Computational Linguistics, 2009.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial intelligence*, 165(1):91–134, 2005.

Miao Fan, Deli Zhao, Qiang Zhou, Zhiyuan Liu, Thomas Fang Zheng, and Edward Y Chang. Distant supervision for relation extraction with matrix completion. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 839–849. Association for Computational Linguistics, June 2014.

Ronen Feldman and James Sanger. *The text mining handbook: advanced approaches in analyzing unstructured data*. Cambridge university press, 2007.

Paolo Ferragina and Ugo Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1625–1628. ACM, 2010.

Germán Ferrero, Audi Primadhanty, and Ariadna Quattoni. Intoevents: An interactive toolkit for discovering and building event schemas. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 104–107, Valencia, Spain, April 2017. Association for Computational Linguistics. URL `http://aclweb.org/anthology/E17-3026`.

Michael Fleischman. Automated subcategorization of named entities. In *ACL (Companion Volume)*, pages 25–30, 2001.

Michael Fleischman and Eduard Hovy. Fine grained classification of named entities. In *In Proc. of the 19th International Conference on Computational Linguistics*, pages 1–7, 2002.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 168–171. Edmonton, Canada, 2003.

Gene H Golub and Henk A Van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1):35–65, 2000.

Matthew R. Gormley, Mo Yu, and Mark Dredze. Improved relation extraction with feature-rich compositional embedding model. In *Proceedings of EMNLP*, 2015.

Yoshihiko Gotoh and Steve Renals. Statistical language modelling. In *Text-and Speech-Triggered Information Access*, pages 78–105. Springer, 2003.

Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *Coling*, volume 96, pages 466–471, 1996.

R Guha and Rob McCool. Tap: A semantic web test-bed. *Web Semantics: Science, Services and Agents on the World Wide Web*, 1(1):81–87, 2003.

Aria Haghighi and Dan Klein. Unsupervised coreference resolution in a nonparametric bayesian model. In *Annual meeting-Association for Computational Linguistics*, volume 45, page 848, 2007.

James Hammerton. Named entity recognition with long short-term memory. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 172–175. Edmonton, Canada, 2003.

Charles T Hemphill, John J Godfrey, George R Doddington, et al. The atis spoken language systems pilot corpus. In *Proceedings of the DARPA speech and natural language workshop*, pages 96–101, 1990.

Iris Hendrickx and Antal van den Bosch. Memory-based one-step named-entity recognition: Effects of seed list features, classifier stacking, and unannotated data. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 176–179. Edmonton, Canada, 2003.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*, pages 229–232. ACM, 2011a.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 782–792. Association for Computational Linguistics, 2011b.

Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57. ACM, 1999.

Ruihong Huang and Ellen Riloff. Peeling back the layers: Detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1137–1147. Association for Computational Linguistics, 2011.

Ruihong Huang and Ellen Riloff. Modeling textual cohesion for event extraction. In *AAAI*, 2012.

Brian Hutchinson, Mari Ostendorf, and Maryam Fazel. A sparse plus low rank maximum entropy language model. In *INTERSPEECH*, pages 1676–1679. Citeseer, 2012.

Heng Ji and Ralph Grishman. Data selection in semi-supervised learning for name tagging. In *Proceedings of the Workshop on Information Extraction Beyond The Document*, IEBeyondDoc '06, pages 48–55, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. ISBN 1-932432-74-4. URL http://dl.acm.org/citation.cfm?id=1641408.1641414.

Heng Ji, Ralph Grishman, et al. Refining event extraction through cross-document inference. In *ACL*, pages 254–262, 2008.

Rosie Jones. *Learning to extract entities from labeled and unlabeled text*. PhD thesis, University of Utah, 2005.

Jae-Ho Kim, In-Ho Kang, and Key-Sun Choi. Unsupervised named entity classification models and their ensembles. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1072228.1072316. URL http://dx.doi.org/10.3115/1072228.1072316.

Dan Klein, Joseph Smarr, Huy Nguyen, and Christopher D. Manning. Named entity recognition with character-level models. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 180–183. Edmonton, Canada, 2003.

Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. Collective annotation of wikipedia entities in web text. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 457–466. ACM, 2009.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270, 2016.

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

Rémi Lebret and Ronan Collobert. Word embeddings through hellinger pca. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E14-1051.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P14-1130.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

Qi Li, Heng Ji, and Liang Huang. Joint event extraction via structured prediction with global features. In *ACL (1)*, pages 73–82, 2013.

Xiao-Li Li, Lei Zhang, Bing Liu, and See-Kiong Ng. Distributional similarity vs. pu learning for entity set expansion. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 359–364, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1858842.1858908.

Shasha Liao and Ralph Grishman. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics, 2010.

Bing Liu and Ian Lane. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016*, 2016.

Cédric Lopez, Ioannis Partalas, Georgios Balikas, Nadia Derbas, Amélie Martin, Coralie Reutenauer, Frédérique Segond, and Massih-Reza Amini. Cap 2017 challenge: Twitter named entity recognition. *arXiv preprint arXiv:1707.07568*, 2017.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. Joint named entity recognition and disambiguation. In *Proc. EMNLP*, pages 879–880, 2015.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany, August 2016. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P16-1101.

Pranava Swaroop Madhyastha, Xavier Carreras, and Ariadna Quattoni. Learning Task-specific Bilexical Embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 161–171. Dublin City University and Association for Computational Linguistics, August 2014. URL http://www.aclweb.org/anthology/C14-1017.

Pranava Swaroop Madhyastha, Xavier Carreras, and Ariadna Quattoni. Tailoring word embeddings for bilexical predictions: An experimental comparison. In *International Conference on Learning Representations 2015, Workshop Track*, 2015.

Beatriz Maeireizo, Diane Litman, and Rebecca Hwa. Co-training for predicting emotions with spoken dialogue data. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 28. Association for Computational Linguistics, 2004.

Gideon S Mann and David Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 33–40. Association for Computational Linguistics, 2003.

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

James Mayfield, Paul McNamee, and Christine Piatko. Named entity recognition using hundreds of thousands of features. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 184–187. Edmonton, Canada, 2003.

Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 188–191, Stroudsburg, PA, USA, 2003a. Association for Computational Linguistics. doi: 10.3115/1119176.1119206. URL http://dx.doi.org/10.3115/1119176.1119206.

Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 188–191. Edmonton, Canada, 2003b.

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. 2002.

Pablo N Mendes, Max Jakob, Andrés García-Silva, and Christian Bizer. Dbpedia spotlight: shedding light on the web of documents. In *Proceedings of the 7th international conference on semantic systems*, pages 1–8. ACM, 2011.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013b.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Hlt-naacl*, volume 13, pages 746–751, 2013c.

David Milne and Ian H Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518. ACM, 2008.

M. Mintz, S. Bills, R. Snow, and D. Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

Tom Mitchell. The role of unlabeled data in supervised learning. In *Proceedings of the sixth international colloquium on cognitive science*, pages 2–11. Citeseer, 1999.

Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in neural information processing systems*, pages 2265–2273, 2013.

Robert Munro, Daren Ler, and Jon Patrick. Meta-learning orthographic and contextual models for language independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 192–195. Edmonton, Canada, 2003.

David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, January 2007. URL `http://www.ingentaconnect.com/content/jbp/li/2007/00000030/00000001/art00002`. Publisher: John Benjamins Publishing Company.

Ali Naderi, Horacio Rodrıguez, and Jordi Turmo. Binary vector approach to entity linking: Talp in tac-kbp 2014. In *In the Seventh Text Analysis Conference, Gaithersburg, MD USA*, 2014a.

Ali M Naderi, Horacio Rodriguez, and Jordi Turmo. The talp participation at erd 2014. In *Proceedings of the first international workshop on Entity recognition & disambiguation*, pages 89–94. ACM, 2014b.

Arvind Neelakantan and Michael Collins. Learning dictionaries for named entity recognition using minimal supervision. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 452–461, Gothenburg, Sweden, April 2014a. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E14-1048.

Arvind Neelakantan and Michael Collins. Learning dictionaries for named entity recognition using minimal supervision. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 452–461, Gothenburg, Sweden, April 2014b. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/E14-1048.

Axel-Cyrille Ngonga Ngomo and Sören Auer. Limes: a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2312–2317. AAAI Press, 2011.

Kiem-Hieu Nguyen, Xavier Tannier, Olivier Ferret, and Romaric Besançon. Generative event schema induction with entity disambiguation. In *ACL (1)*, pages 188–197, 2015.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. Joint event extraction via recurrent neural networks. In *Proceedings of NAACL-HLT*, pages 300–309, 2016.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st World Wide Web Conference 2012, WWW*, pages 271–280, Lyon, France, April 2012. doi: 10.1145/2187836.2187874. URL http://doi.acm.org/10.1145/2187836.2187874.

Cheng Niu, Wei Li, Jihong Ding, and Rohini K. Srihari. A bootstrapping approach to named entity classification using successive learners. In *In Proceedings of the 41st Annual Meeting of the ACL*, pages 335–342, 2003.

Brendan O'Connor. Learning frames from text with an unsupervised latent variable model. *arXiv preprint arXiv:1307.7382*, 2013.

Siddharth Patwardhan and Ellen Riloff. Effective information extraction with semantic affinity patterns and relevant regions. In *EMNLP-CoNLL*, volume 7, pages 717–727, 2007.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Audi Primadhanty, Xavier Carreras, and Ariadna Quattoni. Low-rank regularization for sparse conjunctive feature spaces: An application to named entity 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)*, pages 126–135, Beijing, China, July 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P15-1013.

Ariadna Quattoni, Borja Balle, Xavier Carreras, and Amir Globerson. Spectral regularization for max-margin sequence tagging. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1710–1718. JMLR Workshop and Conference Proceedings, 2014. URL http://jmlr.org/proceedings/papers/v32/quattoni14.pdf.

Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/W09-1119`.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1375–1384. Association for Computational Linguistics, 2011.

L. F. Rau. Extracting company names from text. In *[1991] Proceedings. The Seventh IEEE Conference on Artificial Intelligence Application*, volume i, pages 29–32, Feb 1991. doi: 10.1109/CAIA.1991.120841.

Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/N13-1008`.

Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049, 1996.

Ellen Riloff. Information extraction as a stepping stone toward story understanding. *Understanding language understanding: Computational models of reading*, pages 435–460, 1999.

Ellen Riloff, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32. Association for Computational Linguistics, 2003.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. Effective slot filling based on shallow distant supervision methods. In *Proceedings of NIST KBP workshop*, volume 1, 2013.

Benjamin Roth, Nicholas Monath, David Belanger, Emma Strubell, Patrick Verga, and Andrew McCallum. Building knowledge bases with universal schema: Cold start and slot-filling approaches. In *Proceedings of the Eighth Text Analysis Conference (TAC2015)*, 2015.

Alexandre Salle, Marco Idiart, and Aline Villavicencio. Matrix factorization using window sampling and negative sampling for improved word representations. In *The 54th Annual Meeting of the Association for Computational Linguistics*, page 419, 2016.

Lei Sha, Sujian Li, Baobao Chang, and Zhifang Sui. Joint learning templates and slots for event schema induction. In *Proceedings of NAACL-HLT*, pages 428–434, 2016.

Yusuke Shinyama and Satoshi Sekine. Named entity discovery using comparable news articles. In *Proceedings of the 20th international conference on Computational Linguistics*, page 848. Association for Computational Linguistics, 2004.

Sameer Singh, Tim Rocktaschel, and Sebastian Riedel. Towards combined matrix and tensor factorization for universal schema relation extraction. In *NAACL Workshop on Vector Space Modeling for NLP (VSM)*, June 2015.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *Learning Theory*, pages 545–560. Springer Berlin Heidelberg, 2005.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2005.

Michael Steinbach, George Karypis, Vipin Kumar, et al. A comparison of document clustering techniques. In *KDD workshop on text mining*, volume 400, pages 525–526. Boston, 2000.

Beth M Sundheim. Overview of the fourth message understanding evaluation and conference. In *Proceedings of the 4th conference on Message understanding*, pages 3–21. Association for Computational Linguistics, 1992.

Mihai Surdeanu. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *TAC*, 2013.

Mihai Surdeanu, David McClosky, Julie Tibshirani, John Bauer, Angel X Chang, Valentin I Spitkovsky, and Christopher D Manning. A simple distant supervision approach for the tac-kbp slot filling task. In *TAC*, 2010.

Erik F Tjong Kim Sang and Sabine Buchholz. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning-Volume 7*, pages 127–132. Association for Computational Linguistics, 2000.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada, 2003.

Erik F. Tjong Kim Sang and Hervé Déjean. Introduction to the conll-2001 shared task: Clause identification. In Walter Daelemans and Rémi Zajac, editors, *Proceedings of CoNLL-2001*, pages 53–57. Toulouse, France, 2001.

Ngoc Thang Vu. Sequential convolutional neural networks for slot filling in spoken language understanding. In *Interspeech 2016*, 2016.

Jason Weston, Antoine Bordes, Oksana Yakhnenko, and Nicolas Usunier. Connecting language and knowledge bases with embedding models for relation extraction. In *Conference on Empirical Methods in Natural Language Processing*, pages 1366–1371, 2013.

Casey Whitelaw and Jon Patrick. Named entity recognition using a character-based probabilistic approach. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 196–199. Edmonton, Canada, 2003.

Dekai Wu, Grace Ngai, and Marine Carpuat. A stacked, voted, stacked model for named entity recognition. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 200–203. Edmonton, Canada, 2003.

Puyang Xu and Ruhi Sarikaya. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 78–83. IEEE, 2013.

Limin Yao, Sebastian Riedel, and Andrew McCallum. Probabilistic databases of universal schema. In *Proceedings of the AKBC-WEKEX Workshop at NAACL 2012*, June 2012.

Limin Yao, Sebastian Riedel, and Andrew McCallum. Universal schema for entity type prediction. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 79–84, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2411-3. doi: 10.1145/2509558.2509572. URL `http://doi.acm.org/10.1145/2509558.2509572`.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics, 1995.

Mo Yu, Matthew R. Gormley, and Mark Dredze. Combining word embeddings and feature embeddings for fine-grained relation extraction. In *Proceedings of NAACL*, 2015.

Mo Yu, Mark Dredze, Raman Arora, and Matthew R Gormley. Embedding lexical features via low-rank tensors. In *Proceedings of NAACL-HLT*, pages 1019–1029, 2016.

Tong Zhang and David Johnson. A robust risk minimization based named entity recognition system. In Walter Daelemans and Miles Osborne, editors, *Proceedings of CoNLL-2003*, pages 204–207. Edmonton, Canada, 2003.

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