Improving Structured Prediction for Named-Entity Recognition

by

Hanieh Poostchimohammadabadi

A thesis submitted in partial fulfillment for

the degree of DOCTOR OF PHILOSOPHY

School of Electrical and Data Engineering

Faculty of Engineering and Information Technology

University of Technology Sydney

JANUARY 2019

This thesis is dedicated to

my *father*.

Acknowledgments

I would like to sincerely thank my supervisor Prof. Massimo Piccardi for his support and encouragement during this project. At many stages in the course of this research project I benefited from his advice, particularly so when exploring new ideas. His positive outlook and confidence in my research inspired me and gave me confidence. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly.

I would like to thank my family:

- my Mom and Dad for their love, endless support and encouragement. His memory will be always with me;
- my siblings, specially my sister who has always advised me in making right decisions and my older brother whose memory will be eternal;
- my husband for encouraging and reassuring me in all moments of despair.

This research has been funded by the Capital Markets Cooperative Research Centre and is supported by Semantic Sciences Pty Ltd.

Hanieh Poostchi

January 2019, Sydney

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, Hanieh Poostchimohammadabadi declare that this thesis, is submitted in fulfilment of the requirements for the award of the degree of Doctor of Philosophy, in the School of Electrical and Data Engineering, Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has no been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

Production Note:

Signed: Signature removed prior to publication.

Date: 21.01.2019

Abstract

Natural language processing aims to provide an understanding of human utterances adequate to automatically answer questions, translate documents or retrieve information based on its meaning. At the foundation of these capabilities are text analysis tasks such as named-entity recognition (NER) which aims to identify all "named entities" in a text such as people, locations, organizations, numerical expressions and others.

Great effort has been devoted to NER since its inception in 1996. However, the investigation has been mostly focused on languages with large amounts of digital resources such as English, German, Dutch and Spanish. Indeed, NER is still a challenging task for the many languages with low (i.e., little, scarce, scattered) digital resources and manually-annotated corpora. To abridge this gap, in the beginning of this thesis we have targeted NER for a language with scarce annotated resources, namely Persian, that is spoken by a population of over a hundred and ten million people world-wide. To this end, we have provided and published the first manually-annotated Persian NER corpus and introduced an initial NER pipeline that leverages a word embedding and a sequential max-margin classifier. The experimental results show that the proposed approach has been capable of achieving promising MUC7 and CoNLL scores while outperforming two alternatives based on a CRF and a simple RNN. Upon the introduction of the BiLSTM-CRF in 2015, we have mode forward our research by exploring combinations of various word embeddings with the BiLSTM-CRF architecture, with the best combination beating our initial results by more than 12 percentage points.

Building on the achievements of the BiLSTM-CRF in NER, in this thesis we intro-

duce the BiLSTM-SSVM, an equivalent neural model where training is performed using a structured hinge loss. The typical loss functions used for evaluating NER are entity-level variants of the F_1 score such as the CoNLL and MUC losses. Unfortunately, the common loss function used for training NER - the cross entropy - is only loosely related to these evaluation losses. For this reason, we propose a training approach for the BiLSTM-CRF that leverages a hinge loss bounding the CoNLL loss from above. In addition, we present a mixed hinge loss that bounds either the CoNLL loss or the Hamming loss based on the density of entity tokens in each sentence. The experimental results over four benchmark languages (English, German, Spanish and Dutch) show that training with the mixed hinge loss has led to small but consistent improvements over the cross entropy across all languages and four different evaluation measures.

Another interesting NLP component that has been covered in this thesis is cluster naming. Cluster naming is the assignment of representative labels to clusters of documents or words. Once assigned, the labels can play an important role in applications such as navigation, search and document classification. However, finding appropriately descriptive labels is still a challenging task. Accordingly, we have proposed various approaches for assigning labels to word clusters by leveraging word embeddings and the synonymy and hypernymy relations in the WordNet lexical ontology. Experiments carried out using the WebAP document dataset show that one of the approaches stands out in the comparison and is capable of selecting labels that are satisfactorily aligned with those chosen by a pool of four, independent human annotators.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural Language Processing (NLP), a field at the crossroad of linguistics and machine learning, concerned with the analysis and synthesis of natural language. Mainly, it aims to design systems capable of extracting concise and meaningful information from text. Machine translation, question answering, summarization and sentiment analysis are all examples of high-level (close to the meaning) NLP tasks. At the foundation of these tasks are lower-level (close to the syntax) components such as part of speech (POS) tagging, chunking, named-entity recognition and semantic role labeling. Amongst them, *Named-Entity Recognition* (NER), introduced in the sixth Message Understanding Conference (MUC-6) [Grishman and Sundheim, 1996] and also known as *entity extraction* or *entity identification*, aims to identify all informative named entities (NE) such as the names of people, locations and organizations and numeric expressions in unstructured text. An entity may consists of a single token or multiple tokens, where each token is assigned a tag prefixed by an indicator of the beginning (B-), the inside (I-), or outside (O) of an entity. This annotation scheme is known as IOB2 format, standing for inside, outside and beginning (version 2) [Ramshaw and Marcus, 1995]. Figure 1.1 shows an example sentence annotated with named entities in the IOB2 format.

Early research on NER was mostly devoted to handcrafted rule-based systems which are intrinsically language-dependent, and thus laborious to be extended to new languages. As a consequence, later studies were mainly focused on language-independent machine learning techniques that attempt to learn statistical models for NER from data [Nadeau and Sekine, 2007]. However, supervised training of statistical models requires creation

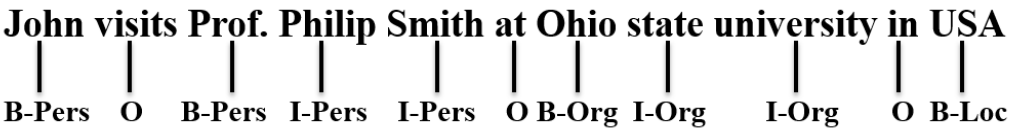

Figure 1.1: An example sentence annotated with named entities in the IOB2 format, where "Pers", "Org", and "Loc" stand for *person*, *organization* and *location* entity types.

of gold standard datasets, such as [Tjong Kim Sang, 2002] for Spanish and Dutch and [Tjong Kim Sang and De Meulder, 2003] for English and German, which initially limited supervised NER investigations to languages with high digital resources. Gradually, replacement of manually-annotated gold standards with very large "silver standard" corpora is mollifying the scarcity of supervised data. Silver standards are NE annotated corpora derived from processing Wikipedia's text and meta-information alongside entity databases such as Freebase [Nothman *et al.*, 2013; Al-Rfou *et al.*, 2015]. This enabled combinations of semi-supervised and distant supervision approaches for other languages [Althobaiti *et al.*, 2015]. However, low digital-resource languages still face a significant scarcity of public repositories. For instance, only 8.8% of Wikipedia articles in Hindi are identified as entity-based articles in Freebase [Al-Rfou *et al.*, 2015]. In this research, our very first aim was to enable supervised NER for a low digital-resource language, namely Persian, by providing the first manually-annotated Persian NE dataset. The Persian language, despite accounting for more than a hundred and ten million speakers around the globe, has been rarely studied for NER [Khormuji and Bazrafkan, 2014] and even text processing [Shamsfard, 2011].

In NLP, most classification problems, such as chunking, POS tagging, slot-filling and NER, are forms of sequential labeling problems. Sequential labeling predicts a sequence of class labels, $y = \{y_1, \dots y_t, \dots y_T\}$, based on a corresponding sequence of measurements, $x = \{x_1, \dots x_t, \dots x_T\}$. A widespread model for sequential labeling is the hidden Markov model (HMM) that factorizes the joint probability of the measurements and the labels, $p(x, y)$, by arranging the latter in a Markov chain (of order one or above) and conditioning the measurement at frame t on only the corresponding label. For an HMM of order one, $p(x, y)$ is expressed as:

$$p(x, y) = p(y_1) \prod_{t=2}^{T} p(y_t|y_{t-1}) \prod_{t=1}^{T} p(x_t|y_t)$$

where $p(y_1)$ is the probability of the initial class, terms $p(y_t|y_{t-1})$ are the transition probabilities and terms $p(x_t|y_t)$ are the emission, or measurement, probabilities. With limited modifications (exponential family for the emission probabilities, denormalized factors), this model becomes the widespread conditional random field (CRF), allowing for discriminative training [Sutton and McCallum, 2012].

Given its inherently sequential nature, NER has often been tackled by sequential classifiers ranging from conventional models like HMMs [Zhou and Su, 2002], linear-chain CRFs [Lafferty *et al.*, 2001; Finkel *et al.*, 2005], structural support vector machines (SSVMs) [Poostchi *et al.*, 2016] and combinations of feed-forward neural networks with CRF [Collobert *et al.*, 2011] to the more recent deep recurrent neural network (RNN) architectures including simple RNNs [Mesnil *et al.*, 2013; Mesnil *et al.*, 2015], bidirectional long short-term memory networks (BiLSTM) [Huang *et al.*, 2015], their combination with CRF (called BiLSTM-CRF [Huang *et al.*, 2015; Lample *et al.*, 2016; Ma and Hovy, 2016]) and the state-of-the-art language model (LM) augmented sequence taggers such as TagLM [Peters *et al.*, 2017] and ELMo [Peters *et al.*, 2018].

Conventional sequential prediction models require real-valued feature vectors as word representations to provide effective predictions. The feature vector can be as simple as a binary vector of text features like '*word is all uppercased*' or a more complex, real-valued vector capturing semantic and syntactic aspects of the word. Word2vec [Mikolov *et al.*, 2013a], GloVe [Pennington *et al.*, 2014], Hellinger-PCA (HPCA) [Lebret and Collobert, 2014] and fastText [Bojanowski *et al.*, 2017] are well-known examples of unsupervised word embeddings applied successfully to the NER task. The neural network approaches have the ability to deliver end-to-end systems for NER. With these approaches, an implicit word embedding is automatically extracted in the network's early layers by initializing the training with random values or a preliminary embedding.

In this thesis, we have explored the combination of various pre-trained word embedding models (including HPCA, GloVe and word2vec) with a range of structured prediction methods (including CRF, structural SVM, Jordan-RNN and the BiLSTM-CRF) for the sequential labeling task of NER using our manually-annotated Persian NE dataset. The models that we have trained are all language-independent tools and can be straight-

forwardly redeployed for other languages.

Moreover, as the main contribution of this thesis, we have introduced the BiLSTM-SSVM, an equivalent neural model to the BiLSTM-CRF where training is performed using a structured hinge loss. The proposed training approach leverages a hinge loss bounding the non-decomposable NER-dedicated loss function, called CoNLL-F_1 score, from above. The CoNLL-F_1 score is a strict version of the F_1 score where a prediction is counted as a true positive only if the entire named-entity mention is segmented and classified correctly. We also have presented a mixed hinge loss that bounds either the CoNLL loss or the Hamming loss based on the density of entity tokens in each sentence. The experimental results over four benchmark languages (English, German, Spanish and Dutch) show that training with the mixed hinge loss has led to small but consistent improvements over the cross entropy, the common loss function used for training NER, across all languages and four different evaluation measures.

Finally, as an extra project suggested by our industry partner, we have worked on automatic cluster labeling. Our industry partner aimed to develop a tool for navigation of a hierarchy of documents. In turn, this requirement called for an illustrative and concise description of the contents of each node of the hierarchy that could be easily understood by a human user. Cluster labeling is the assignment of representative labels to clusters of documents or words [Wang *et al.*, 2014] and can play an important role in applications such as navigation, search and document classification. The most common approaches choose the labels from amongst the most frequent or most central cluster members [Manning *et al.*, 2008]. However, as these approaches can only select cluster labels from the terms and phrases that explicitly appear in the collection, they possibly fail to provide an appropriate level of abstraction [Lau *et al.*, 2011]. For this reason, we have explored several approaches for assigning labels to word clusters by leveraging word embeddings and the synonymy and hypernymy relations in the WordNet lexical ontology. A hypernymy relation represents an asymmetric relation between a class and each of its instances. A hypernym (e.g., *fish*) has a broader context than its hyponyms (*salmon, grouper, bass* etc). The proposed approaches use different criteria for labeling clusters of keywords by a representative selection of their hypernyms. Experiments carried out using the WebAP

document dataset [Keikha *et al.*, 2014] have shown that one of the approaches stand out in the comparison and is capable of selecting labels that are reasonably aligned with those chosen by a pool of human annotators.

1.1 Objectives and Deliverables

A founding aspect of this PhD research is that it has been performed in collaboration with a sponsoring industry partner, namely Semantic Sciences Pty. Ltd.[1], whose aim is to develop and commercialize unstructured data analysis software. The main product of this company, called Sintelix, is a series of text analytic components that enables customers to integrate unstructured text data into their systems. Accordingly, the ultimate goal of this PhD thesis was to develop original research outcomes while fulfilling the industry partner's expectations for accurate, high-level NLP tools such as language-independent NER frameworks and an efficient cluster naming pipeline.

To this end, first, we developed an original NER pipeline based on structural SVM with promising overall performance, and delivered it to our industry partner. Then, with the appearing of the BiLSTM-CRF with character-level encoding in the literature (2016), we delivered them a BiLSTM-CRF NER package that was consistent with their platform's implementation. Eventually, we have made a research contribution on NER by proposing and implementing a novel training algorithm that can target specific performance measures for NER and achieve higher accuracy.

Another interesting NLP component that was appealing to the industry partner for its commercialization potential is automatic cluster naming (aka labeling). Cluster labeling is the assignment of representative labels to clusters of documents or words and can play an important role in applications such as navigation, search and document classification. For this purpose, we have proposed various approaches for assigning labels to word clusters by leveraging word embeddings and the synonymy and hypernymy relations in the WordNet lexical ontology. The pipeline was implemented and delivered to the industry partner as the final deliverable.

[1] `http://www.sintelix.com/`

1.2 Contributions

In this thesis, we first explore a range of structured prediction methods (including CRF, SSVM, Jordan-RNN and BiLSTM-CRF) for the sequential labeling task of NER. We then present the BiLSTM-SSVM, an equivalent neural model to the BiLSTM-CRF where training is performed using a structured hinge loss. The proposed training approach leverages a hinge loss bounding the CoNLL loss from above. In addition, we present a mixed hinge loss that bounds either the CoNLL loss or the Hamming loss based on the density of entity tokens in each sentence.

As a side project, we also investigate the problem of cluster naming and propose a number of approaches for labeling a cluster of words with a set of representative verbal terms.

Our main contributions include:

1. Achieving reasonable NER accuracy for a low digital-resources language, namely Persian, by providing new digital resources including datasets, annotation and collation of existing resources (Chapter 3);

2. Investigating the performance of conventional sequential labeling classifiers, namely CRF, structural SVM, and recurrent deep neural network frameworks such as the Jordan-RNN the and BiLSTM-CRF, for Persian NER (Chapter 3 and 4);

3. Outperforming the state-of-the-art NER systems (on standard English, German, Spanish and Dutch NER datasets) by optimizing a sequential classifier to minimize the CoNLL loss (Chapter 5);

4. Proposing simple and fast approaches for cluster naming that provide labels which are satisfactorily aligned with those of human annotations (Chapter 6).

Overall, the first three contributions form an innovative training algorithm that can target performance measures specific for the NER task and seems sizable for a doctoral dissertation.

1.3 Publications

The outcomes have been published in and presented at high rank NLP conferences and local workshops. An integrated version of the two initial publications at COLING 2016 and LREC 2018 has been invited to be published in forms of a book chapter. Moreover, the main outcome has been submitted to the most prestigious journal of the NLP community, the Transactions of ACL. The list of published papers and under-review publications are as follows:

- Journal paper
  - Hanieh Poostchi and Massimo Piccardi, "BiLSTM-SSVM: Training the BiLSTM to Minimize the CoNLL Loss," (Under-review at the Transaction of ACL).
- Book chapter
  - Hanieh Poostchi and Massimo Piccardi, 'Persian Named Entity Recognition with Structured Prediction Methods," The Handbook of Persian Computational Linguistics and Natural Language Processing, De Gruyter publication (The publisher has scheduled July 2019 for the submission date).
- Conference papers
  - Hanieh Poostchi, Ehsan Zare Borzeshi, Mohammad Abdous, Massimo Piccardi, "PersoNER: Persian Named-Entity Recognition", 26^{th} International Conference on Computational Linguistics (COLING 2016), Osaka, Japan, December 11-16, 2016.
  - Hanieh Poostchi, Ehsan Zare Borzeshi, Massimo Piccardi, "BiLSTM-CRF for Persian Named-Entity Recognition, ArmanPersoNERCorpus: the First Entity-Annotated Persian Dataset", 11^{th} edition of the Language Resources and Evaluation Conference (LREC 2018), Miyazaki, Japan, 7-12 May, 2018.
  - Hanieh Poostchi and Massimo Piccardi, "Cluster Labeling using Word Embeddings and WordNet's Hypernymy," 16^{th} Annual Workshop of The Aus-

tralian Language Technology Association (ALTA), Otago University, Dunedin, New Zealand, 10-12 December, 2018.

- Presentations
  - Hanieh Poostchi, Ehsan Zare Borzeshi, Massimo Piccardi, Daniel McMichael "Multilingual Named Entity Recognition", Improving Health through Data Analytics (CMCRC 2015 annual conference), Sydney, Australia, 12 November, 2015.
  - Hanieh Poostchi, Ehsan Zare Borzeshi, Massimo Piccardi, "Natural Language Processing", Disrupting Health Markets Through Data (CMCRC 2016 annual conference), Sydney, Australia, 23 November, 2016.
  - Hanieh Poostchi, Ehsan Zare Borzeshi, Massimo Piccardi, "PersoNER: Persian Named-Entity Recognition", 14^{th} Annual Workshop of The Australian Language Technology Association (ALTA), Monash University, Melbourne, Australia, 5-7 December, 2016.
  - Hanieh Poostchi, Ehsan Zare Borzeshi, Massimo Piccardi, "Multilingual Named Entity Extraction & Meta-Search based Resource Discovery", Emerging Trends in Digital Health (CMCRC 2017 annual conference), Sydney, Australia, 17-18 May, 2017.
  - Hanieh Poostchi, Ehsan Zare Borzeshi, Massimo Piccardi, "BiLSTM-CRF for Persian Named-Entity Recognition, ArmanPersoNERCorpus: the First Entity-Annotated Persian Dataset", 15^{th} Annual Workshop of The Australian Language Technology Association (ALTA), Queensland University of Technology, Brisbane, Australia, 6-8 December, 2017.

1.4 Thesis Outline

The thesis is organized into seven chapters that are summarized as follows:

Chapter 1 The current chapter, introducing the accomplished research.

Chapter 2 This chapter presents the background knowledge for the proposed contributions. It discusses the concept of named-entity recognition and provides literature review on step-by-step and end-to-end NER frameworks. It covers several unsupervised word embedding methods, conventional sequential classifiers and state-of-the-art deep (recurrent) neural network architectures for sequential labeling. Moreover, it discusses the concept of automatic cluster naming and provides the literature review on cluster labeling and hypernym-hyponym taxonomy construction.

Chapter 3 In this chapter, we present and release ArmanPersoNERCorpus, the first manually-annotated Persian NE dataset, and proposed an NER pipeline for the Persian language. The main components of the pipeline are word embedding by Hellinger PCA and classification by a structural SVM-HMM classifier. Experiments conducted over the ArmanPersoNERCorpus dataset have achieved higher overall F_1 scores than those of a CRF and a Jordan-RNN. The released dataset can be used for further development of Persian NER systems and for evaluation of systems trained on silver-standard corpora, and the achieved accuracy will provide a baseline for future comparisons. We refer the interested readers to [Poostchi *et al.*, 2016] for the published version of this chapter.

Chapter 4 Following the achievements in the previous chapter, in this chapter, we present an approach for Persian NER based on a deep learning architecture, the BiLSTM-CRF, and release four different Persian word embeddings based on GloVe, CBOW, skip-gram and HPCA. The proposed approach has achieved an $F1$ score that is 12.32 percentage points higher on average than the previous results. The released word embeddings could find future use in other Persian NLP tasks including translation, question answering and summarization. We refer the interested readers to [Poostchi *et al.*, 2018b] for the published version of this chapter. In addition, the integration of chapters 3 and 4 has been invited to be published as a book chapter in the Handbook of Persian Computational Linguistics and Natural Language Processing by publisher De Gruyter.

Chapter 5 In this chapter, we present the BiLSTM-SSVM, a training approach for the BiLSTM-CRF based on a hinge loss minimization. In this approach, the hinge loss is

used as an upper bound for three evaluation losses, namely Hamming, CoNLL and a combination of the two. The required loss-augmented inference is challenging in the case of a non-decomposable loss such as CoNLL, and, for this reason, in this chapter we have proposed an articulated dynamic programming algorithm that can perform the loss-augmented inference for the CoNLL loss and any other loss similarly based on entity-level error counting. Since the CoNLL loss is of the F_1 type, we have also argued that it may be a promising training objective for sentences with relatively sparse entities. For this reason, we have proposed a training objective that bounds the CoNLL loss for sentences with low entity density, and the Hamming loss otherwise ("mixed hinge"). Experiments conducted over four benchmark languages (English, German, Spanish and Dutch) have shown that training with the mixed hinge loss has achieved slightly higher accuracies than with the cross entropy for all languages. These results suggest that training with objectives closer to the evaluation measures can be an effective strategy, and that using different losses for sentences with different sufficient statistics should be explored further. This chapter is under review by the Transactions of the ACL.

Chapter 6 This chapter describes an additional project that has been carried out for the industry partner. The project has explored various approaches for labeling keyword clusters based on the hypernyms and synonyms from the WordNet lexical ontology. The proposed approaches map both the keywords and their hypernyms to a word embedding space and leverage the notion of centrality in the cluster. Experiments carried out using the WebAP dataset have shown that one of the approaches has outperformed all the others in terms of precision at k for all values of k, and it has provided labels which are reasonably aligned with those of a pool of annotators. We refer the interested readers to [Poostchi *et al.*, 2018a] for the published version of this chapter.

Chapter 7 We conclude our dissertation with a summary of the main findings and suggestions for possible future extensions.

Chapter 2

Literature Review

2.1 Named-Entity Recognition

Early research on NER was mostly devoted to handcrafted rule-based systems which are intrinsically language-dependent, and thus laborious to be extended to new languages. As a consequence, recent studies are mainly focused on language-independent machine learning techniques that attempt to learn statistical models for NER from data [Nadeau and Sekine, 2007]. Moreover, replacement of manually-annotated gold standards with very large "silver standard" corpora mollifies the scarcity of supervised data. Silver standards are NE annotated corpora derived from processing Wikipedia's text and meta-information alongside entity databases such as Freebase [Nothman *et al.*, 2013; Al-Rfou *et al.*, 2015].

Existing NER approaches mainly divide over two categories: in the first, the task is decoupled into an initial step of word embedding, where words are mapped to feature vectors, followed by a step of word/sentence-level classification. The second category leverages recurrent neural networks (RNNs) to deliver end-to-end systems for NER. With this approach, an implicit word embedding is automatically extracted in the network's early layers by initializing the training with random values or a preliminary embedding. While it is possible to learn an implicit word embedding with any given task loss function and classifier, this approach was de facto popularized by [Collobert *et al.*, 2011]. In this chapter, we cover some of the prominent works that have been published recently in these domains of research.

2.1.1 Unsupervised Word Embedding Methods

Word embedding implies mapping of words into feature vectors. The feature vector can be as simple as a binary vector of text features like '*word is all uppercased*' or a more complex, real-valued vector capturing semantic and syntactic aspects of the word. These vectors can be leveraged as features in a variety of NLP applications like NER. Word2vec [Mikolov *et al.*, 2013a], GloVe [Pennington *et al.*, 2014], Hellinger-PCA [Lebret and Collobert, 2014] and fastText [Bojanowski *et al.*, 2017] are well-known examples of unsupervised word embeddings applied successfully to the NER task.

Primarily, the distance or angle between pairs of word vectors were used to assess the substantial quality of those word mappings against distances assigned by human annotators [Pennington *et al.*, 2014]. However, a new evaluation scheme that prefers mappings with dimensions of meaning was introduced in [Mikolov *et al.*, 2013c]. It suggested that a model should has the encoding ability of illustrating the analogy *"king is to queen as man is to woman"* with the vector equation $king - queen = man - woman$.

Any word embedding method basically requires a general statistic such as term-frequency (tf), term-frequency inverse-document-frequency (tf-idf), bag of words (bow) or word co-occurrence to characterize words in a collection of documents. Out of those, word co-occurrence statistics have the ability to represent a word by the frequencies of its surrounding words which well aligns with the requirements of NER. In the following, we cover few particular word embedding methods that have taken lots of attention recently.

2.1.1.1 Hellinger-PCA

Lebret and Collobert [2014] have shown that a simple spectral method analogous to PCA can produce word embeddings as useful as those of neural learning algorithms such as word2vec. Given an unsupervised training corpus and a vocabulary, V, the co-occurrence matrix, $C_{|V|\times|D|}$, in [Lebret and Collobert, 2014] is computed as:

$$C(v_i, d_j) = p(d_j|v_i) = \frac{n(v_i, d_j)}{\sum_d n(v_i, d)} \tag{2.1}$$

where $v_i \in V; i = 1 \ldots |V|$ and $d_j \in D \subseteq V; j = 1 \ldots |D|$. $n(v_i, d_j)$ is the count of occurrences of context word d_j in the neighborhood of reference word v_i. Thus, $C(v_i, :)$

represents discrete probability distribution $p(d|v_i)$ and is used to characterize v_i. Since words are represented as discrete distributions, Lebret and Collobert [2014] argued that it is more appropriate to measure their distances in a Hellinger space. Accordingly, $H(C)$ is the transformation of C into Hellinger space where the distance between any two discrete probability distributions, P and Q, is given by:

$$dist(P, Q) = \frac{1}{\sqrt{2}}||\sqrt{P} - \sqrt{Q}||_2. \tag{2.2}$$

Eventually, PCA is applied to reduce the dimensionality of $H(C) \in \mathbb{R}^{|V|\times|D|}$ to $h(C) \in \mathbb{R}^{|V|\times m}$, where $m \ll |D|$.

2.1.1.2 GloVe

***Glo**bal **Ve**ctors* is a global log-bilinear regression model with a weighted least-square objective that combines advantages of global matrix factorization and local context windows methods [Pennington *et al.*, 2014]. If entry C_{ij} of the co-occurrence matrix C counts the occurrence of word j in the context of word i, then $P_{ij} = P(j|i) = C_{ij}/C_i$ will be the probability of this co-occurrence. Here, $C_i = \sum_{k=1}^{|V|} C_{ik}$ is the number of times any word appears in the context of word i and $|V|$ is the length of the vocabulary list. Interesting experiments were conducted to figure out how certain aspects of meaning can be extracted directly from co-occurrence probabilities. The outcomes showed that only the ratio of probabilities cancels out the noise from non-discriminative words successfully. Accordingly, the ratios of co-occurrence probabilities are used as starting point for word vector learning. The ratio P_{ik}/P_{jk} depends on three words i, j, and k:

$$F(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{2.3}$$

where $w \in \mathbb{R}^d$ are word vectors and $\tilde{w} \in \mathbb{R}^d$ are context word vectors. One way to capture the information that present the ratio P_{ik}/P_{jk} in the word vector space is to choose F as a function that depends on the difference of the two target word vectors:

$$F(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{2.4}$$

While the right-hand side of (2.4) is a scalar, the arguments of F on the left-hand side are vectors. Taking the dot product of the arguments of F results into:

$$F\big((w_i - w_j)^T \tilde{w}_k\big) = \frac{P_{ik}}{P_{jk}} \tag{2.5}$$

The word list w and context word list $\tilde{w}$ are free to exchange their roles. Therefore, to restore the symmetry, F needs to be a homomorphism:

$$F\big((w_i - w_j)^T \tilde{w}_k\big) = \frac{F(w_i^T \tilde{w}_k)}{F(w_j^T \tilde{w}_k)} \tag{2.6}$$

Substituting (2.5) into (2.6) results into,

$$F(w_i^T \tilde{w}_k) = P_{ik} = \frac{C_{ik}}{C_i} \tag{2.7}$$

The solution to (2.6) is $F = \exp$, or

$$w_i^T \tilde{w}_k = \log(P_{ik}) = \log(C_{ik}) - \log(C_i) \tag{2.8}$$

Indeed, the training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Finally, the weighted least square regression model is introduced by the following cost function:

$$J = \sum_{i,j=1}^{|V|} f(C_{ij})(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - \log C_{ij})^2 \tag{2.9}$$

where f is a weighting function. Since, $\log C_i$ in (2.8) is independent of j, it is absorbed into the bias term b_i for w_i. Moreover, an additional bias $\tilde{b}_j$ for $\tilde{w}_j$ restores the symmetry.

2.1.1.3 Word2Vec

Word2vec is a feedforward neural network language model with the aim to develop a generative model for estimating continuous representations of words that preserves the linear regularities among words [Mikolov *et al.*, 2013b]. This method has introduced two model architectures: i) predicting context words surrounding a given word, **skip-gram**, and ii) predicting word surrounding a context, **continuous bag-of-words** (**CBOW**).

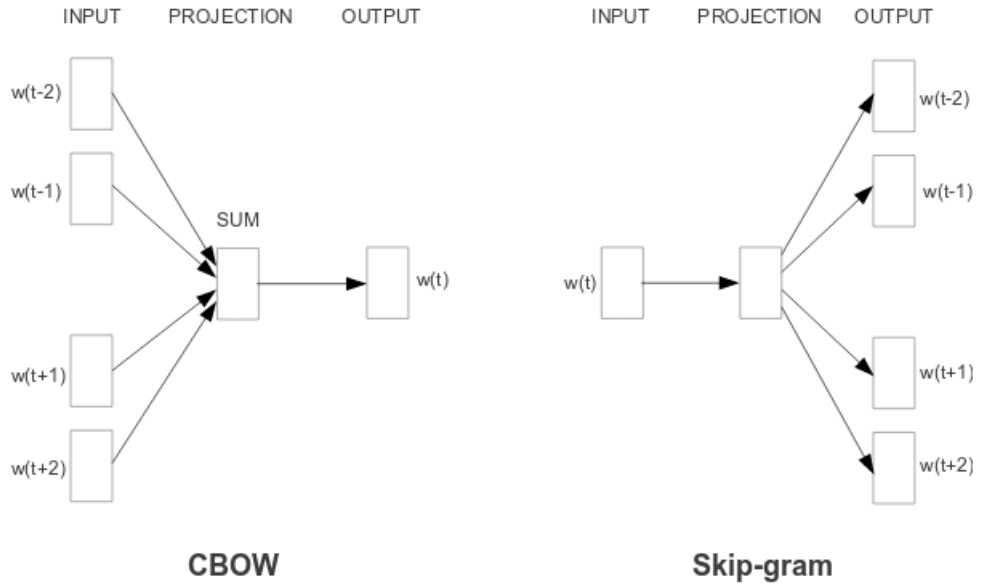

Figure 2.1: The CBOW (left) and the skip-gram (right) model architectures (b orrowed from [Mikolov *et al.*, 2013b]).

As shown in Figure 2.1 (right), the training objective of the *skip-gram* model is to learn word vector representations that are useful for predicting the nearby words in a sentence [Mikolov *et al.*, 2013c]. That is, maximizing the average log probability

$$max \quad \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0} \log p(w_{t+j}|w_t) \tag{2.10}$$

where $w_1, w_2, ..., w_T$ is a sequence of training words and c, a function of the center word w_t, is the length of the training context. A neural network architecture consists of an input projection layer, an output layer and a softmax function designed to solve (2.10). The following softmax function defines $p(w_{t+j}|w_t)$ in the basic skip-gram formulation:

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W} \exp({v'_w}^T v_{w_I})} \tag{2.11}$$

where W is the length of the vocabulary list, and v_w and v'_w are the input and output vector representations of w, respectively. However, since the cost of computing $\nabla \log p(w_O|w_I)$ is proportional to W (which is often in range of 10^5 to 10^7), this formulation is computationally impractical.

In the skip-gram model with negative sampling, a simplified version of noise con-

trastive estimation [Mnih and Teh, 2012] has been applied to approximate the full softmax in a computationally efficient manner. The negative sampling is defined by the objective

$$\log \sigma({v'_{w_O}}^T v_{w_I}) + \sum_{i=1}^{k} \mathbb{E}_{w_i \sim P_n(w)}[\log \sigma(-{v'_{w_i}}^T v_{w_I})] \tag{2.12}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ and k is the number of negative samples for each data sample. The objective is to discriminate the target word w_O from the noise distribution $P_n(w)$ draws using logistic regression; which becomes possible by replacing every $\log P(w_O|w_I)$ with 2.12. For a word, all surrounding words (content words) can be considered as positive examples and negative samples can be sampled randomly from the dictionary.

The *CBOW* model, illustrated in figure 2.1 (left), is similar to skip-gram except that the roles of the input and output are reversed. In this model, words in sequences from past and future are input and they are trained to predict the current word as the network output.

2.1.1.4 FastText

Most of continuous word representation techniques represent each word by a distinct vector and ignore the internal structure of words, which is a significant restriction for morphologically rich languages such as Finnish, Turkish, Slavic and Czech. *FastText* is a new approach based on the skipgram model with negative sampling, where each word is represented as a bag of character n-grams [Bojanowski *et al.*, 2017]. Given a word w and a dictionary of n-grams of size G, the set of n-grams appearing in w is denoted by $\mathcal{G}_w \subset \{1, ..., G\}$. In this method, which takes into account subword information, a vector representation $\mathbf{z}_g$ is associated to each n-gram g. The word w is represented by the sum of the vector representations of its n-grams. To this end, the scoring function

$$s(w, c) = \sum_{g \in \mathcal{G}_w} \mathbf{z}_g^T \mathbf{v}_c \tag{2.13}$$

is used in (2.11), where c is a context word.

By sharing the representation across words, this simple model allows to learn reliable representation for rare words. Moreover, this model is capable of building word vectors

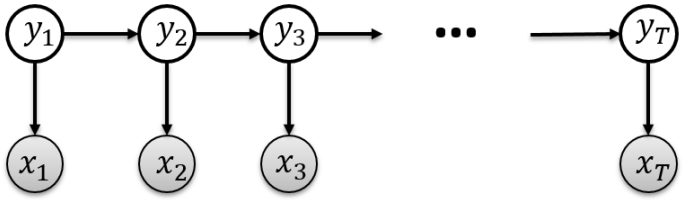

Figure 2.2: Hidden Markov Model (HMM) of order one.

for out-of-vocabulary words by simply averaging the vector representations of its n-grams.

2.1.2 Sequential Classifiers

Sequential labeling is a very common task in NLP for applications such as chunking, POS tagging, slot-filling, and NER. It predicts a sequence of class labels, $y = \{y_1, \dots y_t, \dots y_T\}$, based on a corresponding sequence of measurements, $x = \{x_1, \dots x_t, \dots x_T\}$. In this section, we cover most popular sequential classifiers such as HMMs [Zhou and Su, 2002], CRFs [Lafferty *et al.*, 2001; Finkel *et al.*, 2005], and structural SVM [Lei *et al.*, 2014]. Moreover, after introducing the loss-augmented inference under the structural SVM method in subsection 2.1.2.3, we summarize some surrogate loss functions in the following subsection 2.1.2.4.

2.1.2.1 Hidden Markov Models

A widespread model for sequential labeling is the hidden Markov model (HMM) that factorises the joint probability of the measurements and the labels, $p(x, y)$, by arranging the latter in a Markov chain (of order one or above) and conditioning the measurement at frame t on only the corresponding label, as shown in Figure 2.2. Here, the state of a system cannot be directly observed, yet inferred through measurements of other variables, or observations. This implies that the state of the system at any given time can be treated as a hidden random variable, while a set of measurements which we can acquire from the system are treated as observed variables. For an HMM of order one,

$$p(x, y) = p(y_1) \prod_{t=2}^{T} p(y_t|y_{t-1}) \prod_{t=1}^{T} p(x_t|y_t) \tag{2.14}$$

where $p(y_1)$ is the probability of the initial class, terms $p(y_t|y_{t-1})$ are the transition probabilities and terms $p(x_t|y_t)$ are the emission (also known as measurement or observation) probabilities. By restricting the emission probabilities to the exponential family, i.e., $p(x_t|y_t) \propto exp(w^T f(x_t, y_t))$, the logarithm of probability $p(x, y)$ can be expressed as the score of a *generalized linear model*:

$$\ln p(x, y) \propto w^T \phi(x, y) = w_{in} f(y_1) + \sum_{t=2}^{T} w_{tr}^T f(y_t, y_{t-1}) + \sum_{t=1}^{T} w_{em}^T f(x_t, y_t) \quad (2.15)$$

where w_{in}, w_{tr} and w_{em} are the linear models for assigning a score to the initial classes, transitions and emissions, respectively. Functions $f(y_1)$, $f(y_t, y_{t-1})$ and $f(x_t, y_t)$ are arbitrary, fixed "feature" functions of the measurements and the labels.

The generalised linear model in (2.15) is more suitable for discriminative training than the generative probabilistic model in (2.14). Notable discriminative approaches are conditional random fields (CRFs) [Lafferty *et al.*, 2001] and structural SVM [Tsochantaridis *et al.*, 2005], which will be discussed in the following subsections.

Eventually, given a measurement sequence x in input, inference of the optimal label sequence can be obtained as:

$$\bar{y} = \underset{y}{\operatorname{argmax}}\, p(x, y) = \underset{y}{\operatorname{argmax}}(w^T \phi(x, y)) \quad (2.16)$$

This problem can be efficiently solved in $O(T)$ time by the Viterbi algorithm, a dynamic programming algorithm to find the most likely sequence of hidden states, working in either the linear or logarithmic scale [Rabiner, 1989]. The Viterbi recursion can be defined formally as following, by assuming N possible values for the state and $i, j \in [1 \cdots N]$:

Initialization:

$$v_1(j) = p(y_j|y_0)p(x_1|y_j) \quad (2.17)$$

$$backtrace: \; b_1(j) = 0 \quad (2.18)$$

Recurrence:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)p(y_j|y_i)p(x_t|y_j); 1 < j \leqslant N, 1 < t \leqslant T \tag{2.19}$$

$$backtrace:\ b_t(j) = \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)p(y_j|y_i)p(x_t|y_j); 1 < j \leqslant N, 1 < t \leqslant T \tag{2.20}$$

Termination:

$$The\ best\ score:\ P^* = v_T = \max_{i=1}^{N} v_T(i) \tag{2.21}$$

$$The\ start\ of\ backtrace:\ Q^* = b_T = \operatorname*{argmax}_{i=1}^{N} v_T(i) * p(y_T|y_i) \tag{2.22}$$

2.1.2.2 Conditional Random Fields

A popular probabilistic model for structured prediction with wide applications in many domains such as NLP, computer vision and bioinformatics is the *Conditional Random Fields* (CRF) [Sutton and McCallum, 2012].

In the case of discrete measurements, by simply setting $\lambda_{ij} = \log p(y' = i|y = j)$, the HMM joint probability (2.14) can be rewritten in the following format:

$$p(x, y) = \frac{1}{Z} \exp\left\{ \sum_t \sum_{i,j \in S} \lambda_{ij} \mathbf{1}_{\{y_t = i\}} \mathbf{1}_{\{y_{t-1} = j\}} + \sum_t \sum_{i \in S} \sum_{o \in O} \mu_{oi} \mathbf{1}_{\{y_t = i\}} \mathbf{1}_{\{x_t = o\}} \right\} \tag{2.23}$$

where $\theta = \{\lambda_{ij}, \mu_{oi}\}$ are the parameters of the distribution and can be any real numbers. Moreover, by introducing the concept of *feature functions*, (2.23) can be written more compactly as:

$$p(x, y) = \frac{1}{Z} \exp\left\{ \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t) \right\} \tag{2.24}$$

where each feature function $f_k(y_t, y_{t-1}, x_t)$ takes in the current sample x_t, the label y_t of the current sample, and the label y_{t-1} of the previous sample as input. It is required to assume that there is one feature $f_{ij}(y, y', x) = \mathbf{1}_{\{y=i\}} \mathbf{1}_{\{y'=j\}}$ for each transition (i, j) and one feature $f_{io}(y, y', x) = \mathbf{1}_{\{y=i\}} \mathbf{1}_{\{x=o\}}$ for each state-observation pair (i, o). Finally, the conditional distribution $p(y|x)$ that yields from HMM (2.24) and indicates a linear-chain

CRF is written as:

$$p(y|x) = \frac{p(x,y)}{\sum_{y'} p(y',x)} = \frac{\exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\}}{\sum_{y'} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y'_t, y'_{t-1}, x_t)\right\}} \\ = \frac{1}{Z(x)} \exp\left\{\sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, x_t)\right\} \tag{2.25}$$

The parameters $\theta = \{\lambda_k\}$ is typically estimated by penalized maximum likelihood. To this end, the CRF model (2.25) is substituted into the *conditional log likelihood*, $l(\theta) = \sum_{i=1}^{N} \log p(y^{(i)}|x^{(i)})$, which results into the following expression:

$$l(\theta) = \sum_{i=1}^{N} \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, x_t^{(i)}) - \sum_{i=1}^{N} \log Z(x^{(i)}) \tag{2.26}$$

In general, numerical optimization methods, like gradient ascent, are used to maximize (2.26) based on partial derivatives $\frac{\partial l}{\partial \lambda_k}$. The feature functions can be easily extended to the continuous case.

2.1.2.3 Structural Support Vector Machines

A *support vector machine* (SVM), shown in Figure 2.3, is a binary discriminative classifier that determines a separating hyperplane located at the maximum distance from the closest points, known as support vectors, of the two classes [Vapnik, 1995]. The *soft-margin SVM* finds the hyperplane $w^T x + b = 0$, also known as the decision boundary, by solving the following optimization function:

$$\underset{w,b}{\operatorname{argmin}} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \quad s.t. \\ y_i(w^T x_i + b) \geq 1 - \xi_i \quad , \quad i = 1 \ldots N, \ , \quad y_i \in \{-1, +1\} \tag{2.27}$$

where 1) $\frac{1}{\|w\|}$ corresponds to the margin, 2) slack variable ξ^i account for a penalty for the constraints that cannot be satisfied and are proven to be an upper bound for the empirical

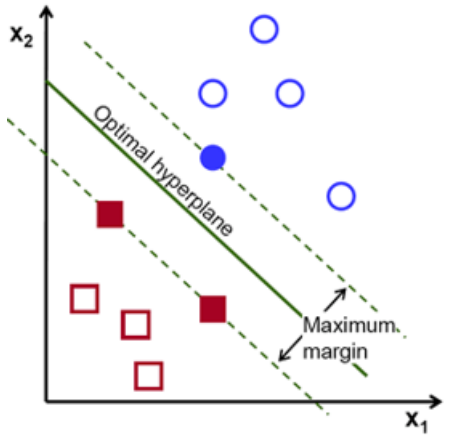

Figure 2.3: A hyperplane separating two classes.

loss, and 3) arbitrary weight C holds the balance between the two terms. In practice, the existence of noise or inseparable data from the two classes, makes it impossible to elegantly separate the two classes by a hyperplane. In this case, the slack variables are introduced as penalties for allowing data points to be sit on the wrong side of the hyperplane. The value of the slack variables are proportional to the distance from the hyperplane.

Once the model is learned, the class label for an unseen sample x is inferred by:

$$\bar{y} = \underset{y}{\operatorname{argmax}}\, y(w^T x + b) \tag{2.28}$$

Structural SVM is an extension of the support vector machine for the classification of structured outputs such as sequences and trees. Amongst other, it has built a very strong reputation for experimental accuracy in diverse NLP tasks such as chunking, NER and semantic parsing [Joachims *et al.*, 2009; Tang *et al.*, 2013; Qu *et al.*, 2014]. In the case of NER, structural SVM finds the model's parameters, w, from a supervised training set of sequences, $\{X, Y\} = \{x^i, y^i\}, i = 1 \dots N$, by minimizing the usual SVM trade-off between the $L2$ regulariser and the hinge loss [Tsochantaridis *et al.*, 2005]. Its learning objective can be expressed as:

$$\begin{gathered} \underset{w,\xi}{\operatorname{argmin}}\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi^i \quad s.t. \\ w^T\phi(x^i, y^i) - w^T\phi(x^i, y) \geq \Delta(y^i, y) - \xi^i \quad , \quad i = 1 \dots N,\ \forall y \in \mathcal{Y} \end{gathered} \tag{2.29}$$

In the objective function, the first term is the regularizer while the second term, $\sum_{i=1}^{N} \xi^i$, is the hinge loss, i.e. a convex upper bound over the total loss on the training set. Hyperparameter C is an arbitrary, positive coefficient that balances these two terms. In the constraints, $w^T \phi(x, y)$ computes the generalized linear score for a (x, y) pair. In the case of sequential labeling, such a score is given by Eq. (2.15). Eventually, $\Delta(y^i, y)$ is the loss function chosen to assess the loss over the training set. The goal is to make the model assign higher score to the correct label y^i than any other labelings, possibly with a large margin. The choice of the evaluation loss as the margin leads to keep the most undesirable predictions at a larger margin, while keeping nearly-correct predictions at a lower margin. This approach is known as *margin rescaling*.

For an NER task with M entity classes, each sequence of length T adds $(M + 1)^T$ constraints to (2.29). Due to their exponential number, exhaustive satisfaction of all constraints is infeasible. However, [Tsochantaridis *et al.*, 2005] has shown that it is possible to find ϵ-correct solutions with a subset of the constraints of polynomial size consisting of only the "most violated" constraint for each sequence, i.e. the labeling with the highest sum of score and loss:

$$\begin{aligned} \xi^i &= \max_y (-w^T \phi(x^i, y^i) + w^T \phi(x^i, y) + \Delta(y^i, y)) \\ \rightarrow \bar{y}^i &= \operatorname*{argmax}_y (w^T \phi(x^i, y) + \Delta(y^i, y)) \end{aligned} \tag{2.30}$$

Here, the score of the correct label y^i must beat that of the closest rival, $\bar{y}^i$, by a margin while the model's norm is kept as low as possible. This problem is commonly referred to as "loss-augmented inference" given its resemblance with the common inference of Eq. (2.16) and is the core of structural SVM. In the case of scores and losses that can be computed frame by frame (such as the 0-1 loss or the Hamming loss), the Viterbi algorithm with appropriate weights can still be used to compute the loss-augmented inference in $O(T)$ time.

2.1.2.4 Surrogate Loss Functions

The main aim of classifier training is to find a parametrization minimizing the expectation of a chosen loss function. However, the loss functions commonly used for evaluation such as the 0-1 loss are discontinuous in parameter space, non-convex and flat over large regions. For this reason, the common approach is to optimize an alternative function, referred to as surrogate loss, instead of the chosen loss. The most well-known surrogates, which are both upper bounds of the 0-1 loss, are the hinge loss in soft-margin SVM:

$$\bar{l}_{hinge} = \max_{y'} \left[l(y, y') - w^T \phi(x, y) + w^T \phi(x, y') \right] \tag{2.31}$$

and the logistic loss (also known as negative log-likelihood or cross entropy in the neural networks community):

$$\bar{l}_{logistic} = \log \Big(\sum_{y'} \big(w^T \phi(x, y') \big) \Big) - w^T \phi(x, y) \tag{2.32}$$

where ϕ is the scoring function. Other structured surrogate loss functions that have found use in the literature include the structured probit loss [McAllester and Keshet, 2011]:

$$\bar{l}_{probit} = \mathbb{E}_{\epsilon \sim N(0,1)} [l(y, \hat{s}_{w+\epsilon}(x))] \tag{2.33}$$

where $\hat{s}_w(x) = \operatorname{argmax}_{y'} w^T \phi(x, y')$; the structured ramp loss (also called truncated hinge loss) [McAllester and Keshet, 2011; Gimpel and Smith, 2012; Huang *et al.*, 2014]:

$$\bar{l}_{ramp}(u) = \begin{cases} 1 - b, & u < b, \\ \bar{l}_{hinge}(u), & b \leq u \leq 1, \\ 0, & u > 1. \end{cases} \tag{2.34}$$

and the structured orbit loss [Karmon and Keshet, 2015]:

$$\bar{l}_{orbit} = \mathbb{P}_{\epsilon \sim N(0,1)} [\epsilon > w \cdot \delta\phi(y, y')] l(y, y') \tag{2.35}$$

where $\delta\phi(y, y')$ is the normalized version of $\Delta\phi(y, y') = w \cdot \phi(x, y) - w \cdot \phi(x, y')$:

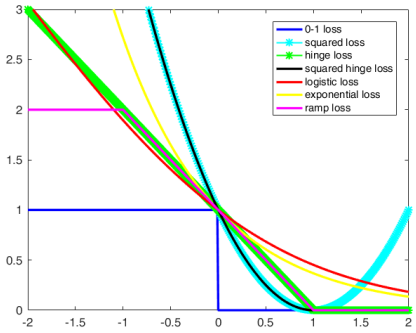

Figure 2.4: Comparison of a number of evaluation and surrogate loss functions for the binary case. The horizontal axis maps the difference in the scores of the correct and incorrect labels; the vertical axis maps the loss.

$$\delta\phi(y, y') = \begin{cases} \Delta\phi(y, y')/||\Delta\phi(y, y')||, & \text{if } y \neq y' \\ 0, & \text{if } y = y' \end{cases} \tag{2.36}$$

In this way, the orbit loss can be interpreted as the cost multiplied by the probability that the prediction score $w \cdot \phi(x, y')$ plus a small number ϵ is greater than the score of the true label $w \cdot \phi(x, y)$. Figure 2.4 illustrates a number of evaluation and surrogate loss functions for the binary case.

Other work has proposed the direct minimization of the evaluation loss by means of asymptotic equalities [McAllester *et al.*, 2010; Song *et al.*, 2016]. The aim of direct loss minimization is to minimize the expectation of the task loss:

$$w^* = \operatorname*{argmin}_w \mathbb{E}[l(y, y_w)] \tag{2.37}$$

If the scoring function F is linear in the parameters, i.e., $F(x, y, w) = w^T\phi(x, y)$, the direct loss gradient has the following form:

$$\nabla_w \mathbb{E}[l(y, y_w)] = \lim_{\epsilon \to 0} \frac{1}{\epsilon} \mathbb{E}[\phi(x, y_{direct}) - \phi(x, y_w)] \tag{2.38}$$

with

$$
\begin{aligned}
y_w &= \underset{\hat{y}}{\operatorname{argmax}}\, F(x, \hat{y}, w) \\
y_{direct} &= \underset{\hat{y}}{\operatorname{argmax}}\, F(x, \hat{y}, w) + \epsilon l(y, \hat{y}).
\end{aligned}
\tag{2.39}
$$

2.1.3 End-to-End Deep (Recurrent) Neural Networks

The approach proposed by [Collobert *et al.*, 2011] and followed by many including [Mesnil *et al.*, 2013; Mesnil *et al.*, 2015; Huang *et al.*, 2015], leverages neural networks to deliver end-to-end systems for sequence tagging tasks including NER. With this approach, there is no explicit need for preliminary word embeddings. The network is fed initially with random vectors as word representations and an implicit word embedding is automatically optimized in the network's early layers during the convergence process. Therefore, one can avoid exploiting hand-made input features which requires a priori linguistic proficiency or designing task-specific cost functions for an optimal word embedding.

Collobert et. al. [2011] proposed an architecture consisting of a multilayer neural network (with/without a convolution layer) and a CRF layer on the output to solve four NLP sequence tagging tasks. The tasks were POS, chunking, NER and semantic role labeling (SLR) which all were seen as a word label-assignment task. Figure 2.5 depicts the multilayer neural network architecture. The network takes a sentence or an n-gram of words as input. The first layer extracts features for each word in the input (the so-called *word embedding*) and stores them in a lookup table. The lookup table, that contains pertinent presentation for each word, is initialized with random vectors and trained/updated by the back-propagation. However, the architecture can take advantage of pre-trained word embeddings, by simply initializing the word lookup table with these word embeddings. The second layer extracts features from the whole sequence which guides the deep layers of the network to learn features that are relevant to the tagging task. To this end, two approaches were proposed: *i)* a window approach to tag one word at the time and *ii)* a (convolutional) sentence approach.

The window approach, shown in Figure 2.5, assumes the tag of a word depends mainly on its surrounding words. In this approach, a matrix of word features is produced by passing each word in the window through the lookup table. The produced matrix is fed

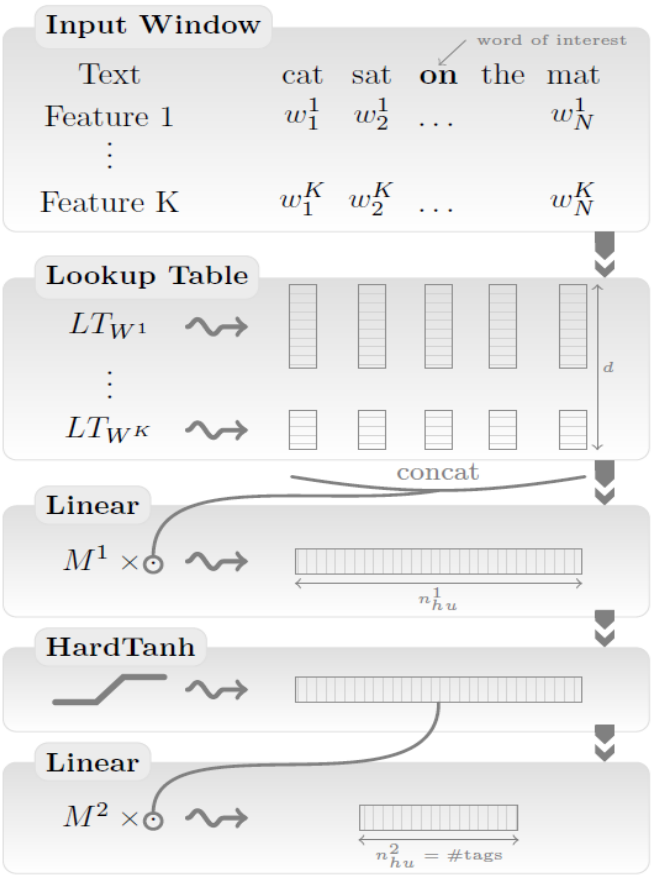

Figure 2.5: Multilayer neural network architecture for sequence tagging (reproduced from [Collobert *et al.*, 2011]).

into linear layers which perform affine transformations over their input. Several linear layers, interleaved with hyperbolic tangent as non-linearity, are often stacked to extract highly non-linear features. The last layer outputs a score for each possible tag for the central word of the input window. Accordingly, the system is trained in an end-to-end manner for any specific task.

For SRL, a word may not be predicted correctly if the *verb* of a sentence falls outside of the window. A convolutional approach is the natural choice when using neural networks to consider the whole sentence. It can be seen as a generalization of a window approach where the aforementioned matrix is produced for each window (as local features) of successive windows in the sequence. The local feature vectors are combined by a max operation to obtain a global feature vector which can be then fed to standard affine network layers.

The labels that are used for training the network are provided from supervised benchmarks. Concisely, the introduced neural network uses stochastic gradient ascent to maximize a conditional likelihood over the supervised training data:

$$\theta \rightarrow \sum_{(x,y)} \log p(y|x,\theta) \tag{2.40}$$

where θ denotes all the network parameters to be trained, x corresponds to either a training word window or sentence, and y represents the corresponding tag. The probability $p(\cdot)$ is computed from the output of the neural network.

In the word-level log-likelihood approach, each word x in a sentence is considered independently. Given network parameters θ, the conditional tag probability $p(i|x,\theta)$ is obtained by applying a softmax operation over all the tags:

$$p(i|x,\theta) = \frac{e^{[f_\theta(x)]_i}}{\sum_j e^{[f_\theta(x)]_j}} \tag{2.41}$$

where $[f_\theta(x)]_i$ is the network's output score for the i'th tag. Accordingly, the log-likelihood for training example (x, y) can be expressed as:

$$\log p(y|x,\theta) = [f_\theta(x)]_y - \log \sum_j e^{[f_\theta(x)]_j} \tag{2.42}$$

which is known as *cross-entropy* and is widely used in supervised learning. However, it simply ignores any correlation between the tag of a word in a sentence and its surrounding tags. To enforce dependencies between the predicted tags in a sentence, the sentence structure is taken into account by encouraging valid paths of tags during training and discouraging all others. This becomes possible by adding a parameter to the model for optimizing the scores for going from any tag to another. In this way, considering the network score $f_\theta([x]_1^T)$ for sentence the $[x]_1^T$ and the transition score $[A]_{i,j}$ for jumping from tag i to tag j in successive words, the sentence score along a path of tags $[i]_1^T$ is written as:

$$s([x]_1^T, [i]_1^T, \tilde{\theta}) = \sum_{t=1}^{T} \left([A]_{[i]_{t-1},[i]_t} + [f_\theta[x]]_{[i]_t}, t \right) \tag{2.43}$$

where $\tilde{\theta} = \theta \cup \{[A]_{i,j}, \forall i, j\}$. As in (2.41), this score is normalized over all possible tag paths $[j]_1^T$ by using a softmax to obtain the conditional tag path probability. Then, by taking the log, the conditional log probability of the true path $[y]_1^T$ is given by:

$$\log p([y]_1^T | [x]_1^T, \tilde{\theta}) = s([x]_1^T, [y]_1^T, \tilde{\theta}) - \log \sum_{\forall [j]_1^T} s([x]_1^T, [j]_1^T, \tilde{\theta}). \tag{2.44}$$

In CRF, the same likelihood as (2.44) is maximized by using a linear model instead of a non-linear neural network. At inference time, the Viterbi algorithm is used to find the best tag path which maximizes the sentence score (2.43). That is:

$$\underset{[j]_1^T}{\operatorname{argmax}}\, s([x]_1^T, [j]_1^T, \tilde{\theta}). \tag{2.45}$$

Following the interesting reported results from the above architecture (that achieved the state-of-the-art tagging accuracy back on those dates) further work had been investigated in applying recurrent neural networks (RNNs), including Elman-type and Jordan-type RNNs [Mesnil *et al.*, 2013; Mesnil *et al.*, 2015], BiLSTM-CRF [Huang *et al.*, 2015] and BiLSTM-CNN-CRF for sequential labeling tasks. In the following we will describe these architectures.

2.1.3.1 Elman-RNN and Jordan-RNN

Deep learning approaches based on the Elman-type and the Jordan-type recurrent neural networks (RNN) have been successfully used by [Mesnil *et al.*, 2013; Mesnil *et al.*, 2015] for the sequence discrimination task of slot filling in spoken language understanding. Similar to NLP tasks, the input is a sequence of words and the output is a sequence of slot/concept IDs, one for each word. The aim is to automatically extract semantic concepts, or to fill in a set of augments or slots embedded in a semantic frame. For instance, in example sentence *"show flight from Boston to New York today"* from [Mesnil *et al.*, 2015]:

- *"Boston*$_{\textit{B-dept}}$*"* and *"New*$_{\textit{B-arr}}$ *York*$_{\textit{I-arr}}$*"* are extracted as *departure* and *arrival* concepts. Moreover, *"today*$_{\textit{B-date}}$*"* is extracted as *date*.
- *"Boston*$_{\textit{B-city}}$*"* and *"New*$_{\textit{B-city}}$ *York*$_{\textit{I-city}}$*"* are recognized as named entities (*city*).

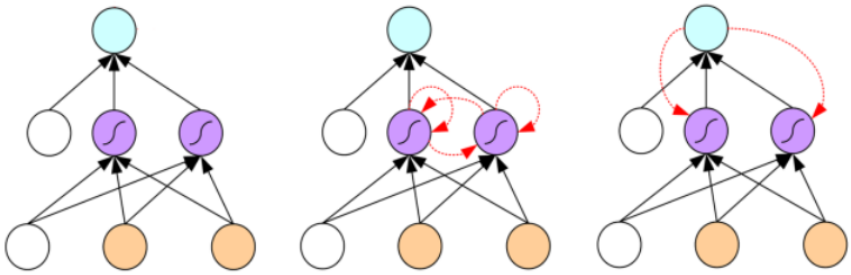

Figure 2.6: Three types neural networks. (left) Feed-forward NN; (middle) Elman-RNN; (right) Jordan-RNN (reproduced from [Mesnil *et al.*, 2015]).

- The intent of the sentence is to *find flight*.
- and, the domain of the sentence is *airline travel*.

As shown in Figure 2.6, the Elman-RNN [Elman, 1990] is similar to the feed-forward neural networks except that the output of the hidden layer at time $t-1$ is fed back into it at time t along with the input for time t. In the Jordan-RNN [Jordan, 1997] the output layer feeds into the hidden layer. In contrast with feed-forward neural networks, the RNNs keep track of the previous hidden layer states through the recurrent connections. Thus, the hidden layer serves as a state that summarizes the past inputs along with the current input. This design allows RNNs to perform well in sequence-predictions tasks.

Mathematically, the Elamn dynamics are depicted as:

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h h_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned} \tag{2.46}$$

where x_t is the input vector, y_t is the output vector and h_t is the hidden layer vector. While σ_h and σ_y are the activation functions, the parameters to be learned are the recurrent weights connection U, and feed-forward weights matrix W and bias vector b. Figure 2.7 shows the Elman-RNN (also known as simple RNN) structure, and its compact visualization, for a sample sentence and the NER task. Similarly, the Jordan dynamics are as:

$$\begin{aligned} h_t &= \sigma_h(W_h x_t + U_h y_{t-1} + b_h) \\ y_t &= \sigma_y(W_y h_t + b_y) \end{aligned} \tag{2.47}$$

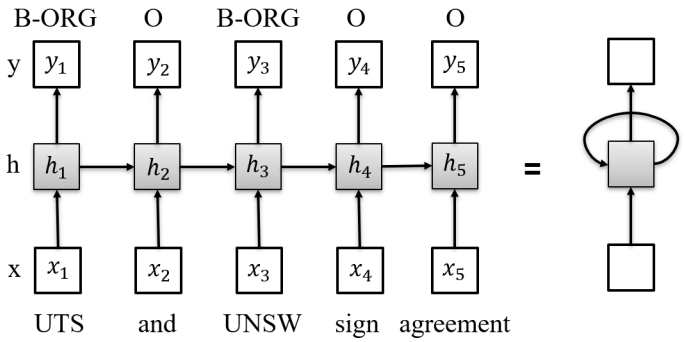

Figure 2.7: A simple RNN structure for the NER task and its compact visualization.

2.1.3.2 Bidirectional Long Short Term Memory Networks

Huang et. al. [2015] applied a variety of long short-term memory (LSTM) based models for three NLP sequence tagging tasks. These models include LSTM network, bidirectional LSTM (BiLSTM), LSTM with a CRF layer, and BiLSTM with a CRF layer which is well-known as BiLSTM-CRF. The BiLSTM-CRF is similar to the model proposed by [Collobert *et al.*, 2011] where the convolution layer is replaced with a BiLSTM network.

As explained before, simple RNNs are able to predict the current output conditioned on distance features by maintaining a memory unit. LSTM networks [Hochreiter and Schmidhuber, 1997] are the same as simple RNNs, except that the hidden layer updates are replaced by particular cells that are well designed to remember longer-term dependencies. Figure 2.8 illustrates a single LSTM memory cell. The LSTM memory cell is implemented as follows [Graves, 2013]:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \tanh(c_t)
\end{aligned}
\tag{2.48}
$$

where σ is the logistic sigmoid function. The input, forget and output gates are represented with i, f and o and c is the cell vector; all the same size as hidden vector h. The weight

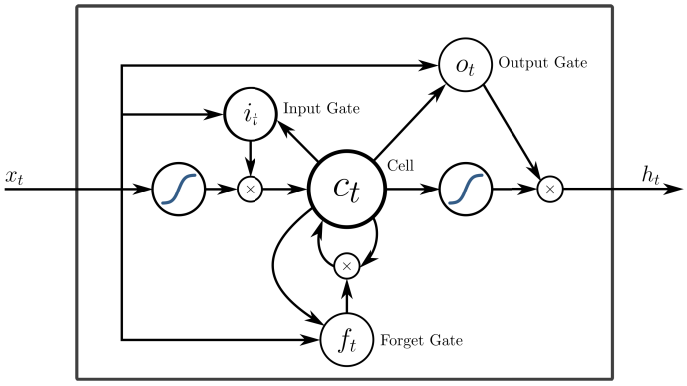

Figure 2.8: A single LSTM memory cell.

matrix subscripts implies the meaning. For instance, W_{hi} and W_{xo} are the hidden-input and input-output gate matrices, respectively. The gates, which are equipped with logistic sigmoid to push their values between 0 and -1, describe how much of each component should be let through. A 1 value represents to "thoroughly keep this" while a 0 represents "thoroughly get rid of this". The input gate i and the forget gate f decide what information are going to store in and throw away from the cell state by looking at h_{t-1}, x_t and old cell state c_{t-1}. The old cell state is updated with previous steps already decided what to do. The old state is first multiplied by f_t, forgetting the things we decided to forget earlier. Then, the new candidate values, scaled by how much we decided to update each state value, is added. The output gate o decides what information are going to output. Finally, the hidden state is updated by putting the cell state through tanh and multiplying it by o_t to only output the parts we decided to.

Simple RNNs and LSTM networks only make use of previous context. However, Bidirectional RNNs, like BiLSTM network [Graves and Schmidhuber, 2005; Graves *et al.*, 2013], process the data with two separate hidden layers in order to make use of past and future information. As shown in Figure 2.9, the forward hidden sequence $\overrightarrow{h}$ and the backward hidden sequence $\overleftarrow{h}$ are fed forwards to the same output layer, after being concatenated $h = [\overrightarrow{h}; \overleftarrow{h}]$.

2.1.3.3 BiLSTM-CRF

The BiLSTM-CRF is a recurrent neural network obtained from the combination of a BiLSTM and a CRF [Huang *et al.*, 2015; Lample *et al.*, 2016]. These two models enjoy com-

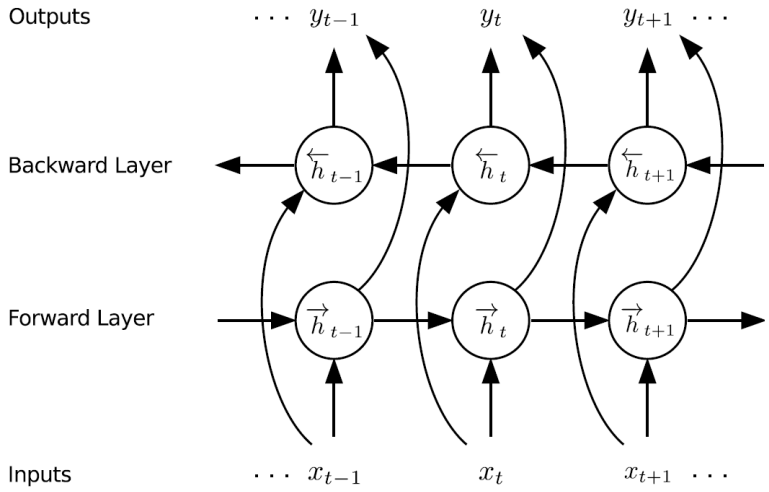

Figure 2.9: A Bidirectional RNN (reproduced from [Graves *et al.*, 2013]).

plementary features: as a complex, nonlinear model, the LSTM can effectively capture the sequential relationships amongst the input tokens. In turn, the CRF permits optimal, joint prediction of all the labels in the sentence, capturing the relationships at label level. The structure is visualized in Figure 2.10. In the BiLSTM-CRF, the posterior probability of label sequence y given input sequence x can be expressed as:

$$p(y|x) = \frac{\exp(F(x,y))}{\sum_{u \in Y} \exp(F(x,u))} \tag{2.49}$$

where F is the scoring function of the BiLSTM-CRF and Y is the set of all possible predictions. Given a training set of labelled sequences, $\{x_i, y_i\}, i = 1 \dots N$, the BiLSTM-CRF can be trained by minimizing the cross entropy (or negative conditional log-likelihood):

$$\bar{w} = \underset{w}{\operatorname{argmin}} - \sum_{i=1}^{N} \ln p(y_i|x_i, w) \tag{2.50}$$

where we have made explicit the dependence on the model's parameters, w, including the transition weights of the CRF, the weights of the main and auxiliary LSTMs, and the embeddings of all tokens and characters. Replacing (2.49) in (2.50), we obtain:

$$\bar{w} = \underset{w}{\operatorname{argmin}} - \sum_{i=1}^{N} [F(x_i, y_i; w) - \ln \sum_{u \in Y} \exp(F(x, u; w)] \tag{2.51}$$

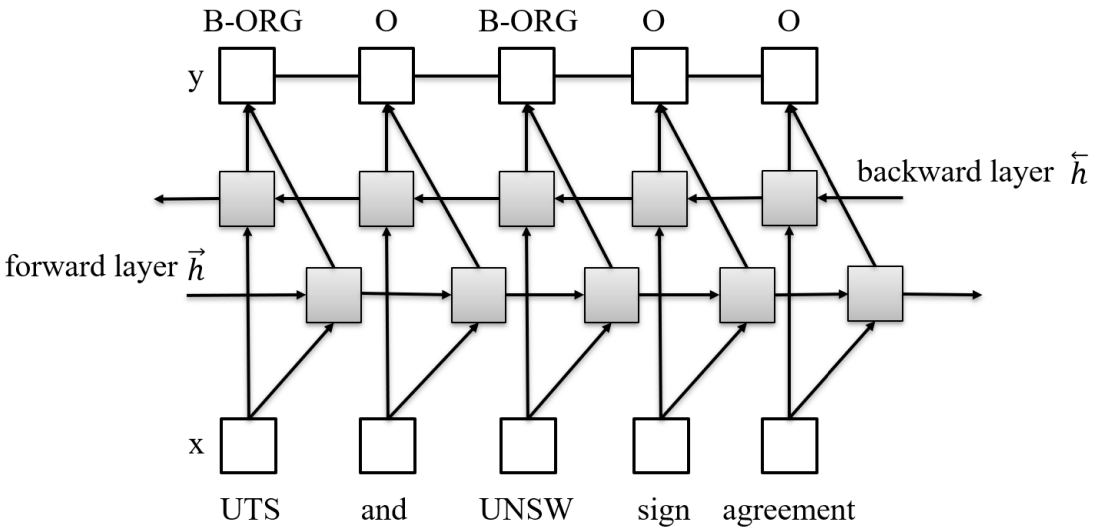

Figure 2.10: A BiLSTM-CRF model.

showing that the optimal parameters are a trade-off between the score assigned to the true labeling and the log-sum-exp of the scores of all the possible labelings. Once the model is trained, inference for a new sentence x is obtained as:

$$\bar{y} = \underset{y}{\operatorname{argmax}}\, p(y|x, \bar{w}) = \underset{y}{\operatorname{argmax}}\, F(x, y; \bar{w}) \tag{2.52}$$

by propagating x through the network and applying the Viterbi algorithm at the CRF output layer.

In the BiLSTM-CRF, the LSTM layers are used to process each sentence token-by-token and produce an intermediate representation as input for the CRF to provide the prediction of all the tokens' labels. In [Lample *et al.*, 2016], the network also includes a second, auxiliary LSTM that further encodes each token character-by-character to capture the regularities at character level. For each token in a sentence, its word and characters embeddings are concatenated and used as the input of the corresponding slot in the main LSTM. Figure 2.11 shows a complete diagram of the BiLSTM-CRF with the auxiliary character encoding layer. The word embedding of the 5-th token is noted as x_5 in the diagram. Its character-level embedding is the last output of the auxiliary LSTM and is noted as x_5^*.

BiLSTM-CNN-CRF is another variant of the BiLSTM-CRF with character encoding

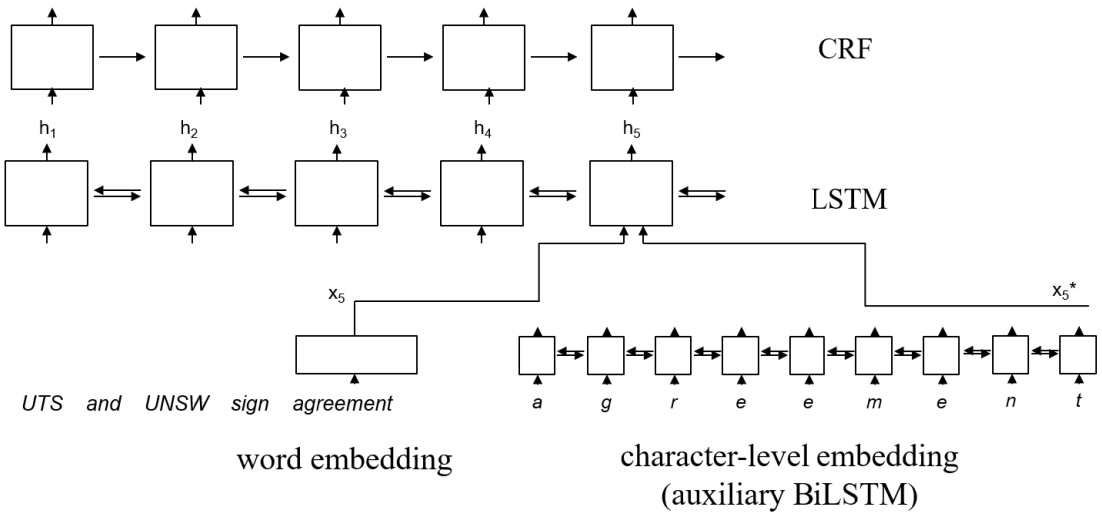

Figure 2.11: A diagram of the BiLSTM-CRF with an auxiliary LSTM layer for character encoding.

where the auxiliary LSTM is replaced by a convolutional neural network (CNN) for character encoding [Ma and Hovy, 2016]. The two variants are compared against each other in [Reimers and Gurevych, 2017].

2.1.3.4 Language Model Augmented Sequence Taggers

More recently, sequential labeling has extensively incorporated neural language models (LM) to improve the token encoding. Peters et al. [2017] have proposed TagLM, a hierarchical RNN model where pre-trained, bidirectional LM embeddings are concatenated with the output of the first bidirectional RNN layer. The evaluation of TagLM on NER and chunking tasks surpasses previous systems and achieves the state-of-the-art. In ELMo [Peters *et al.*, 2018], an improvement of TagLM, the bidirectional LM embeddings are obtained from the aggregation of all the internal layers of a deep bidirectional LM. It significantly improves the the state-of-the-art across six NLP problems, including question answering, textual entailment, semantic role labeling, coreference resolution, NER and sentiment analysis.

The tokens representation are commonly initialized with pre-trained word embeddings, which are rich in semantic and syntactic information. However, they lack contextual information. An approach to encode token sequences into a context sensitive

representation is to include a bidirectional RNN. However, while the word embeddings are pre-trained on large unsupervised corpora, the parameters of the bi-RNNs are typically learned on a relatively small-size supervised data sets. TagLM is a semi-supervised approach, which leverages a neural language model (LM) to compute the context encodings of sequence tokens in the supervised sequence tagging model. The LM embeddings, which are pre-trained on a large unlabeled corpus, encode the semantic and syntactic roles of words in context as they are used to compute the probability of future words in a neural language model.

As shown in Figure 2.12, TagLM is a hierarchical neural tagging model, following BiLSTM-CRF, with a separate module of multiple layers of LSTMs (on the right) for learning LM representations. A forward/backward language model compute the probability of a sequence as:

$$\begin{aligned} &\text{forward:} \quad p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \cdots, t_{k-1}) \\ &\text{backward:} \quad p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_{k+1}, t_{k+2}, \cdots, t_N) \end{aligned} \tag{2.53}$$

A representation of token t_k, for instance a CNN over characters, is passed through multiple layers of LSTMs to embed the history $(t_1, t_2, \ldots, t_k)$ and the future context $(t_k, t_{k+1}, \cdots, t_N)$ into fixed dimensional vectors $\overrightarrow{h}_k^{LM}$ and $\overleftarrow{h}_k^{LM}$. The two LMs, with a softmax layer over the final LSTM to predict the probabilities, are trained independently without sharing any parameters. After the pre-training, the bidirectional LM embedding h_k^{LM} is formed by concatenating the two vectors $[\overrightarrow{h}_k^{LM}; \overleftarrow{h}_k^{LM}]$.

The sequence tagger module (on the left) consists of $L = 2$ layers of bidirectional LSTMs or gated recurrent units (GRU) [Cho *et al.*, 2014] to learn a context sensitive representation. The output from the first layer of the bi-RNNs in the sequence model is concatenated with the LM embeddings to have $h_{k,1} = [\overrightarrow{h}_{k,1}; \overleftarrow{h}_{k,1}; h_k^{LM}]$. The output of the final Bi-RNN layer $h_{k,L}$ predicts a score for each possible tag by using a single dense layer. Finally, the ultimate CRF layer takes into account the dependencies between successive tags.

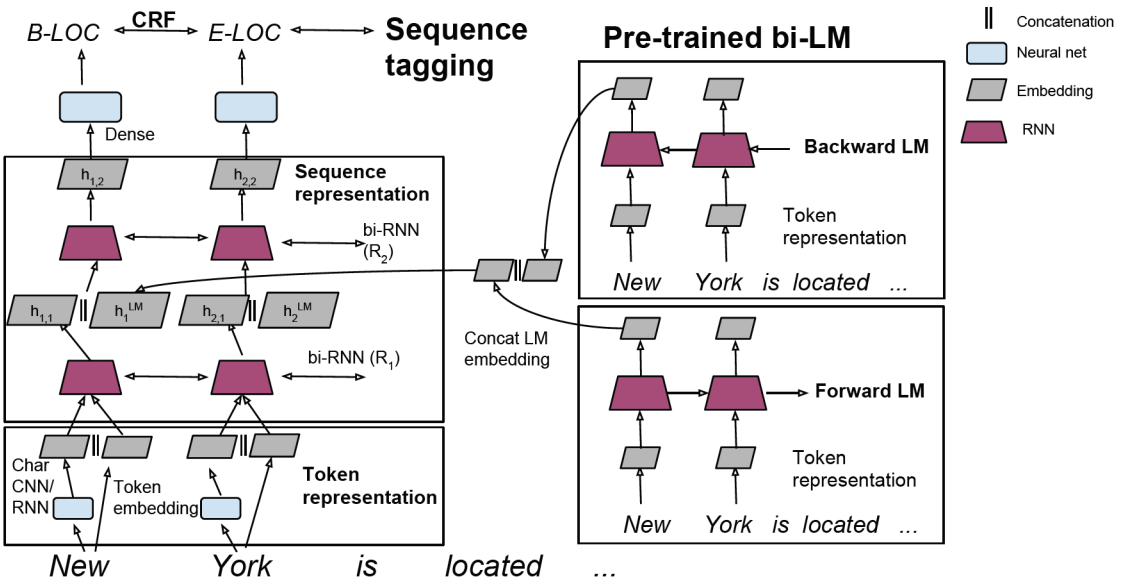

Figure 2.12: The TagLM architecture (reproduced from [Peters *et al.*, 2017]). The output of the forward and backward neural language models (on the right) are concatenated with the output from the first layer of Bi-RNNs in the sequence tagger model (on the left).

There are alternative possibilities to add the LM embeddings to the sequence model. In contrast to TagLM which only selects LM embeddings from the output of the top LSTM layer of the Bi-LM module, in Embeddings from Language Models (ELMo), the deep LM representation is a function of all of the Bi-LM's layers. That function is a weighted linear combination of the vectors stacked above each layer, with the higher-level LSTM states capturing context-dependent aspects of word meaning and the lower-level LSTM states modelling syntax aspects [Peters *et al.*, 2018]. To this end, the softmax-normalized weights are optimized specifically for each task. Moreover, in ELMo, the token representation Θ_x and softmax layer Θ_s is shared between the forward and backward LMs and are optimized jointly with the maximization of the log likelihood of the two directions:

$$\begin{aligned}\sum_{k=1}^{N}(&\log p(t_k|t_1,\cdot,t_{k-1};\Theta_x,\overrightarrow{\Theta}_{LSTM},\Theta_s)\\&+\log p(t_k|t_{k+1},\cdot,t_N;\Theta_x,\overleftarrow{\Theta}_{LSTM},\Theta_s)).\end{aligned} \tag{2.54}$$

Liu et al. [2018] have proposed the LM-LSTM-CRF, where transfer learning and multi-task learning are used to extract character-level representations. In this model, the

objective functions of the LM and the sequence labeling are minimized jointly. In a similar vein, in [Rei, 2017] the objective function includes the prediction of the previous word, the current label and the next word in the sequence.

2.1.4 NER Evaluation Metrics

The NLP community borrowed the precision and recall evaluation measures from the information retrieval community to define particular F_1 scores for the NER task [Bikel *et al.*, 1999; Nadeau and Sekine, 2007]. The **CoNLL-$\mathbb{F}_1$** score[1] is a strict measure to evaluate the performance of any named-entity recognition system. The precision and recall are defined as below:

$$precision = \frac{\#correct\ chunks}{\#guessed\ chunks} \quad , \quad recall = \frac{\#correct\ chunks}{\#true\ chunks} \tag{2.55}$$

where '$\#correct\ chunks$' indicates the number of entity chunks which labels and both boundaries are predicted correctly. The number of predicted entity chunks and the total number of entity chunks in the ground truth are indicated with '$\#guessed\ chunks$' and '$\#true\ chunks$', respectively. Then, the usual F_1 measure is computed as the uniformly weighted harmonic mean of precision and recall from (2.55):

$$F_1 = \frac{precision \times recall}{0.5 \times (precision + recall)} \tag{2.56}$$

MUC7[2] is a relaxed version of CoNLL-F_1 where the predictions are evaluated separately in terms of class and segmentation [Chinchor, 1998; Bikel *et al.*, 1999; Voorhees and Chinchor, 2001]. Under these terms, a prediction from a named-entity extractor is fully correct if the class and both entity boundaries are predicted correctly. Moreover, a prediction is partially correct if the class and only one of the boundaries are recognized correctly. To this end, after aligning the entity chunks in the ground truth with the predicted entity chunks, a number of tallies are calculated to compute the final F_1 score [Chinchor, 1998]:

[1] `http://www.cnts.ua.ac.be/conll2000/chunking/output.html`
[2] `https://catalog.ldc.upenn.edu/docs/LDC2001T02/MUC_scorer3.3/`

System	CoNLL-F_1
[Collobert *et al.*, 2011]	89.59
[Huang *et al.*, 2015]	90.10
[Lample *et al.*, 2016]	90.94
[Ma and Hovy, 2016]	91.21
[Peters *et al.*, 2017]	91.93
[Peters *et al.*, 2018]	92.22

Table 2.1: Comparison of the state-of-the-art techniques on CoNLL 2003 English NER dataset.

- COR (correct): the two entities (the ground truth versus the predicted entity) are considered identical.
- INC (incorrect): the two entities are not identical.
- PAR (partially correct): the two aligned entities are partially identical.
- MIS (missing): no predicted entity for an entity in the ground truth.
- SPU (spurious): a predicted entity does not aligned with any entity in the ground truth.
- POS (possible): POS = COR + INC + PAR + MIS.
- ACT (actual): ACT = COR + INC + PAR + SPU.

Accordingly, the final MUC7 F_1 score is calculated using (2.56), where:

$$\begin{aligned} precision &= \frac{COR + (0.5 \times PAR)}{ACT} \\ recall &= \frac{COR + (0.5 \times PAR)}{POS} \end{aligned} \tag{2.57}$$

Table 2.1 shows the comparison of CoNLL 2003 English NER [Tjong Kim Sang and De Meulder, 2003] results, in terms of CoNLL-F_1, obtained by some of the neural networks based models described in this chapter.

2.2 Automatic Cluster Naming

Document collections are often organized into clusters of either documents or words to facilitate applications such as navigation, search and classification. The organization can prove more useful if its clusters are characterized by *sets of representative labels*. The task of assigning a set of labels to each individual cluster in a document organization is known as cluster labeling [Wang *et al.*, 2014]. In Manning et al. [2008], cluster labeling approaches have been subdivided into:

i) *differential cluster labeling* which selects cluster labels by comparing the distribution of terms in one cluster with those of the other clusters, and

ii) *cluster-internal labeling* which selects labels that are solely based on each cluster individually.

Cluster-internal labeling approaches include computing the clusters' centroids and using them as labels, or using lists of terms with highest frequencies in the clusters. However, all these approaches can only select cluster labels from the terms and phrases that explicitly appear in the documents, possibly failing to provide an appropriate level of abstraction or description [Lau *et al.*, 2011]. As an example, a word cluster containing words *dog* and *wolf* should not be labeled with either word, but as *canids*.

An approach for labeling clusters with abstractive terms is to leverage hypernymy relations. A hypernymy relation represents an asymmetric relation between a class and each of its instances. A hypernym (e.g., *vertebrate*) has a broader context than its hyponyms (*bird, fishes, reptiles* etc). Conversely, the contextual properties of the hyponyms are usually a subset of those of their hypernym(s). A representative selection of the hypernyms of cluster's members makes a general description for the cluster. Hypernym-hyponym datasets are available on public taxonomies, such as WordNet [Miller, 1995], or can be built as explained in section 2.2.1.

Another approach for cluster labeling is to make use of Wikipedia's meta-data. Lau [2011] proposed a model for automatic labeling of topic models, where the label candidate sets are the title of Wikipedia articles containing the top-ranking topic terms. A post-processing approach extract sub-strings from the candidate set to represent them as the

ultimate topic labels. Following [Lau *et al.*, 2011], Shraey et. al. [2016] computed neural embeddings for document and words to select the most relevant labels for topics. They trained a *doc2vec* and a *word2vec* model on the English Wikipedia articles to generate embeddings for Wikipedia titles. With the *doc2vec* model, the embedding of a Wikipedia title is represented by its document embedding. Moreover, as *doc2vec* runs *word2vec* internally, word embeddings are also learned during the training. The topic embedding is represented by the word embeddings of the top-N topic term. The *word2vec* model, treats titles and all of the Wikipedia articles as single tokens to generate embeddings for Wikipedia titles. Given a topic T and a candidate title a, the relevance of the title embedding is measured as:

$$\begin{aligned} rel_{d2v+w2v}(a,T) &= rel_{d2v}(a,T) + rel_{w2v}(a,T) \\ rel_{d2v}(a,T) &= \frac{1}{|T|}\sum_{v\in T}\cos\left(E^{d}_{d2v}(a), E^{w}_{d2v}(v)\right) \\ rel_{w2v}(a,T) &= \frac{1}{|T|}\sum_{v\in T}\cos\left(E^{w}_{w2v}(a), E^{w}_{w2v}(v)\right) \end{aligned} \tag{2.58}$$

where $E^{d}_{d2v}(x)$ and $E^{w}_{d2v}(y)$ are the document embedding of title x and the word embedding of word y generated by *doc2vec*, and $E^{w}_{w2v}(z)$ is the word embedding of word z generated by *word2vec*. A supervised learn-to-rank model attempts to improve the quality of top-ranking candidates by re-ranking them. The supervised re-ranker, a support vector regression model, uses four features, including *LetterTrigram* (the overlap of letter trigrams between a given topic label and the topic words), *PageRank* (which uses directed links to estimate the significane of a document and prefers labels that represent more core concepts in Wikipedia), *NumWords* (the number of words in the candidate label), and *TopicOverlap* (the lexical overlap between the candidate label and the top-N topic terms).

2.2.1 Building Hypernym-Hyponym Hierarchical Structures

Taxonomy extraction refers to the automatic extraction of hierarchical relations from text and construction of the subsequent taxonomy. It typically consists of four modules: term extraction, relation discovery, taxonomy construction, and taxonomy cleaning. In term

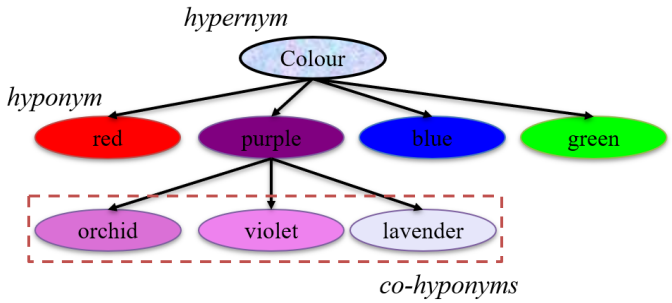

Figure 2.13: Taxonomy as a hypernym-hyponym hierarchical structure.

extraction, the aim is to identify the set of terms which are specific to the domain and will form the lexicons of the ontology. Relation discovery finds the relations between pair of terms, which is usually expressed by verbs, and taxonomy construction organizes them in a hierarchical structure. Finally, taxonomy cleaning prunes the resulting structure. Figure 2.13 shows a representative example of a taxonomy of entities, as a hypernym-hyponym hierarchical structure.

There are two main approaches in the literature on automatic taxonomy extraction:

i) Linguistics methods which employ pre-defined rules or heuristic patterns to extract terms and relations from the web, and

ii) Statistical methods that model ontology learning as a clustering or a classification task.

In [Kozareva and Hovy, 2010], predefined patterns are used as queries for hypernym-hyponym harvesting from the web. For clarification, consider having a root concept *root* and a term called *seed*. Submitting the hyponym pattern $P_i = \{concept$ such as seed and $*\}$ to any search engine retrieves a set of co-hyponyms for *seed*. The searching process continues by substituting *seed* with any term $* \in P_i$. Similarly, the set of hypernyms are retrieved by submitting the hypernym pattern $P_c = \{*$ such as $term_1$ and $term_2\}$. Then, a graph-based taxonomy induction algorithm is applied to position intermediate concepts that are located between the root concept and basic-level terms. To this end, patterns such as "*X are Y that*" and "*X including Y*" were submitted to the search engine. The term with more hits wins the higher position in the hierarchical structure.

An unsupervised ontology learning framework based on available background knowledge is proposed in [Hoxha *et al.*, 2016]. First, the term extraction is performed by looking for the longest phrases in the background knowledge base for biomedical text. Then, a set of concept similarity measures are defined to compute the similarity among the concepts. Finally, a hierarchical agglomerative clustering technique used the similarity matrix to organize the objects into a dendrogram.

A supervised approach, called ExTaSem, has been proposed in [Espinosa-Anke *et al.*, 2016]. First, a valuable set of terms is accumulated by adding in the synonyms of the initial seed terms from BabelNet. Second, the set of concepts is aggregated by looking for the Wikipedia categories that each term belongs to. Third, hypernym classification with CRF++ is engineered using the supervised WCL[3] dataset, a database of hypernym-hyponym definitions. As the feature vectors, the classifier employs word-sense embedding vectors, SENSEEMBED [Iacobacci *et al.*, 2015], which are trained on the English Wikipedia and BabelNet as a reference sense inventory. Finally, a heuristic algorithm based on hypernym decomposition constructs a set of candidate paths which are later used by the second heuristic algorithm for taxonomy creation.

Hypernymy also has been used extensively in natural language processing, including in recent works such as Yu et al. [2015] and HyperVec [Nguyen *et al.*, 2017] that have proposed neural models for learning hierarchical word embeddings that reflect the hypernymy relation. These models are generally trained on pairwise training data to move hypernym and hyponym vectors close to each other. Another field of research, including [Fu *et al.*, 2014] and [Shwartz *et al.*, 2016], focuses on identification of hypernym-hyponym relation between word pairs by using features like vector offsets for supervised classification.

[3] `http://lcl.uniroma1.it/wcl`

Chapter 3

Persian Named-Entity Recognition with Structural SVM

Named-Entity Recognition (NER) is still a challenging task for languages with low digital resources. The main difficulties arise from the scarcity of annotated corpora and the consequent problematic training of an effective NER pipeline. To abridge this gap, in this chapter we target the Persian language that is spoken by a population of over a hundred million people world-wide. We first present and provide ArmanPerosNERCorpus, the first manually-annotated Persian NER corpus. Then, we introduce PersoNER, an NER pipeline for Persian that leverages a word embedding and a sequential max-margin classifier. The experimental results show that the proposed approach is capable of achieving promising MUC7 and CoNLL scores while outperforming two alternatives based on a CRF and a recurrent neural network.

3.1 Introduction

Named-Entity Recognition (NER), introduced in the sixth Message Understanding Conference (MUC-6) [Grishman and Sundheim, 1996], concerns the recognition of Named Entities (NE) and numeric expressions in unstructured text. Since 1996, great effort has been devoted to NER as a foundational task for higher-level natural language processing tasks such as summarization, question answering and machine translation.

Shortage of gold standards has initially limited NER investigation to high-resource

languages such as English, German, Dutch and Spanish [Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003]. Gradually, publicly available encyclopediae have enabled combinations of semi-supervised and distant supervision approaches for other languages [Althobaiti *et al.*, 2015]. However, low-resource languages still face a significant scarcity of public repositories. For instance, only 8.8% of Wikipedia articles in Hindi are identified as entity-based articles in Freebase [Al-Rfou *et al.*, 2015]. In this work, we aim to enable supervised NER for a low-resource language, namely Persian, by providing the first manually-annotated Persian NE dataset. The Persian language, despite accounting for more than a hundred million speakers around the globe, has been rarely studied for NER [Khormuji and Bazrafkan, 2014] and even text processing [Shamsfard, 2011]. In addition, we present PersoNER, a Persian NER pipeline consisting of a word embedding module and a sequential classifier based on the structural support vector machine [Tsochantaridis *et al.*, 2005]. The proposed pipeline achieves reasonable MUC7 and CoNLL scores and outperforms two alternatives based on a CRF and a recurrent neural network.

3.2 Related Work

Early research on NER was mostly devoted to handcrafted rule-based systems which are intrinsically language-dependent, and thus laborious to be extended to new languages. As a consequence, recent studies are mainly focused on language-independent machine learning techniques that attempt to learn statistical models for NER from data [Nadeau and Sekine, 2007]. Moreover, replacement of manually-annotated gold standards with very large "silver standard" corpora mollifies the scarcity of supervised data. Silver standards are NE annotated corpora derived from processing Wikipedia's text and meta-information alongside entity databases such as Freebase [Nothman *et al.*, 2013; Al-Rfou *et al.*, 2015].

Existing NER approaches mainly divide over two categories: in the first, the task is decoupled into an initial step of word embedding, where words are mapped to feature vectors, followed by a step of word/sentence-level classification. The feature vector can be as simple as a binary vector of text features like '*word is all uppercased*'

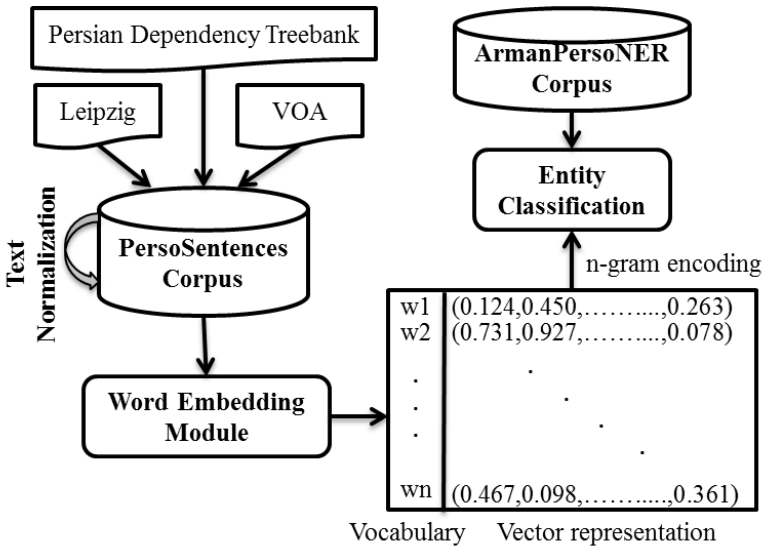

Figure 3.1: PersoNER workflow.

or a more complex, real-valued vector capturing semantic and syntactic aspects of the word. Word2vec [Mikolov *et al.*, 2013a], GloVe [Pennington *et al.*, 2014] and Hellinger-PCA [Lebret and Collobert, 2014] are well-known examples of unsupervised word embeddings applied successfully to the NER task. For classification, sequential classifiers such as HMMs [Zhou and Su, 2002], CRFs [Lafferty *et al.*, 2001; Finkel *et al.*, 2005] and deep neural networks [Al-Rfou *et al.*, 2015] have been amongst the most popular choices.

The second category, proposed by [Collobert *et al.*, 2011] and recently followed by many including [Mesnil *et al.*, 2013; Mesnil *et al.*, 2015] and others, leverages recurrent neural networks (RNNs) to deliver end-to-end systems for NER. With this approach, an implicit word embedding is automatically extracted in the network's early layers by initializing the training with random values or a preliminary embedding. In this chapter, we apply and compare approaches from both categories.

3.3 The Proposed Approach

The workflow of PersoNER is illustrated in Figure 3.1. The steps include data collection, text normalization, word embedding and entity classification. In this section, we focus on the two technical modules, word embedding and classification, while data collection and text normalization are described in Section 3.4.

3.3.1 Word Embedding

Term-frequency (tf), term-frequency inverse-document-frequency (tf-idf), bag of words (bow) and word co-occurrence are general statistics intended to characterize words in a collection of documents. Out of them, word co-occurrence statistics have the ability to represent a word by the frequencies of its surrounding words which well aligns with the requirements of NER. Recently, Lebret and Collobert [2014] have shown that a simple spectral method analogous to PCA can produce word embeddings as useful as those of neural learning algorithms such as word2vec. Given an unsupervised training corpus and a vocabulary, V, the co-occurrence matrix, $C_{|V|\times|D|}$, in [Lebret and Collobert, 2014] is computed as:

$$C(v_i, d_j) = p(d_j|v_i) = \frac{n(v_i, d_j)}{\sum_d n(v_i, d)} \tag{3.1}$$

where $v_i \in V; i = 1 \dots |V|$ and $d_j \in D \subseteq V; j = 1 \dots |D|$. $n(v_i, d_j)$ is the count of occurrences of context word d_j in the neighborhood of reference word v_i. Thus, $C(v_i, :)$ represents discrete probability distribution $p(d|v_i)$ and is used to characterize v_i. Since words are represented as discrete distributions, Lebret and Collobert [2014] argue that it is more appropriate to measure their distances in a Hellinger space. Accordingly, $H(C)$ is the transformation of C into Hellinger space where the distance between any two discrete probability distributions, P and Q, is given by:

$$dist(P, Q) = \frac{1}{\sqrt{2}} ||\sqrt{P} - \sqrt{Q}||_2. \tag{3.2}$$

Eventually, PCA is applied to reduce the dimensionality of $H(C) \in \mathbb{R}^{|V|\times|D|}$ to $h(C) \in \mathbb{R}^{|V|\times m}$, where $m \ll |D|$.

3.3.2 Classification

In this subsection, we first briefly introduce sequential labeling as a formal problem and then describe the sequential classifier based on the structural support vector machine.

3.3.2.1 Sequential Labeling

Sequential labeling predicts a sequence of class labels, $y = \{y_1, \dots y_t, \dots y_T\}$, based on a corresponding sequence of measurements, $x = \{x_1, \dots x_t, \dots x_T\}$. It is a very common task in NLP for applications such as chunking, POS tagging, slot-filling and NER. A widespread model for sequential labeling is the hidden Markov model (HMM) that factorizes the joint probability of the measurements and the labels, $p(x, y)$, by arranging the latter in a Markov chain (of order one or above) and conditioning the measurement at frame t on only the corresponding label. For an HMM of order one, $p(x, y)$ is expressed as:

$$p(x, y) = p(y_1) \prod_{t=2}^{T} p(y_t|y_{t-1}) \prod_{t=1}^{T} p(x_t|y_t) \tag{3.3}$$

where $p(y_1)$ is the probability of the initial class, terms $p(y_t|y_{t-1})$ are the transition probabilities and terms $p(x_t|y_t)$ are the emission, or measurement, probabilities. By restricting the emission probabilities to the exponential family, i.e., $p(x_t|y_t) \propto exp(w^T f(x_t, y_t))$, the logarithm of probability $p(x, y)$ can be expressed as the score of a *generalized linear model*:

$$\ln p(x, y) \propto w^T \phi(x, y) = w_{in} f(y_1) + \sum_{t=2}^{T} w_{tr}^T f(y_t, y_{t-1}) + \sum_{t=1}^{T} w_{em}^T f(x_t, y_t) \tag{3.4}$$

where w_{in}, w_{tr} and w_{em} are the linear models for assigning a score to the initial classes, transitions and emissions, respectively. Functions $f(y_1)$, $f(y_t, y_{t-1})$ and $f(x_t, y_t)$ are arbitrary, fixed "feature" functions of the measurements and the labels.

The generalized linear model in (3.4) is more suitable for discriminative training than the generative probabilistic model in (5.1). Notable discriminative approaches are conditional random fields (CRFs) [Lafferty *et al.*, 2001] and structural SVM [Tsochantaridis *et al.*, 2005]. In particular, structural SVM has built a very strong reputation for experimental accuracy in NLP tasks [Joachims *et al.*, 2009; Tang *et al.*, 2013; Qu *et al.*, 2014] and for this reason we exploit it in our NER pipeline.

Eventually, given a measurement sequence x in input, inference of the optimal label

sequence can be obtained as:

$$\bar{y} = \underset{y}{\operatorname{argmax}}\, p(x, y) = \underset{y}{\operatorname{argmax}}(w^T \phi(x, y)) \tag{3.5}$$

This problem can be efficiently solved in $O(T)$ time by the Viterbi algorithm working in either the linear or logarithmic scale [Rabiner, 1989].

3.3.2.2 Structural SVM

From a supervised training set of sequences, $\{X, Y\} = \{x^i, y^i\}, i = 1 \dots N$, *structural SVM* finds the model's parameters, w, by minimizing the usual SVM trade-off between the hinge loss and an $L2$ regularizer [Tsochantaridis *et al.*, 2005]. Its learning objective can be expressed as:

$$\begin{aligned} &\underset{w,\xi}{\operatorname{argmin}}\ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi^i \quad s.t. \\ &w^T\phi(x^i, y^i) - w^T\phi(x^i, y) \geq \Delta(y^i, y) - \xi^i, \\ &\qquad i = 1 \dots N,\ \ \forall y \in \mathcal{Y} \end{aligned} \tag{3.6}$$

In the objective function, the first term is the regularizer while the second term, $\sum_{i=1}^{N}\xi^i$, is the hinge loss, i.e. a convex upper bound over the total loss on the training set. Hyperparameter C is an arbitrary, positive coefficient that balances these two terms. In the constraints, $w^T\phi(x, y)$ computes the generalized linear score for a (x, y) pair. In the case of sequential labeling, such a score is given by Eq. (3.4). Eventually, $\Delta(y^i, y)$ is the loss function chosen to assess the loss over the training set.

For an NER task with M entity classes, each sequence of length T adds $(M + 1)^T$ constraints to (3.6). Due to their exponential number, exhaustive satisfaction of all constraints is infeasible. However, [Tsochantaridis *et al.*, 2005] has shown that it is possible to find ϵ-correct solutions with a subset of the constraints of polynomial size consisting of only the "most violated" constraint for each sequence, i.e. the labeling with the highest sum of score and loss:

$$\begin{aligned} \xi^i &= \max_y(-w^T\phi(x^i, y^i) + w^T\phi(x^i, y) + \Delta(y^i, y)) \\ \rightarrow \bar{y}^i &= \operatorname*{argmax}_y(w^T\phi(x^i, y) + \Delta(y^i, y)) \end{aligned} \tag{3.7}$$

This problem is commonly referred to as "loss-augmented inference" given its resemblance with the common inference of Eq. (5.5) and is the core of structural SVM. In the case of scores and losses that can be computed frame by frame (such as the 0-1 loss or the Hamming loss), the Viterbi algorithm with appropriate weights can still be used to compute the loss-augmented inference in $O(T)$ time.

3.4 Data Collection

The collected corpora consists of two datasets: *i*) an unsupervised corpus, called PersoSentencesCorpus, that we use for the word embedding module and *ii*) a manually named-entity annotated dataset of Persian sentences, called ArmanPersoNERCorpus, that we use for supervised classification and release as the first ever publicly-available Persian NER dataset.

3.4.1 PersoSentencesCorpus

A very large corpus of documents covering a variety of contexts is required to populate an effective co-occurrence matrix. We fulfill this requirement by accumulating the following three datasets of Persian sentences:

- The *Leipzig corpora* [1] with 1,000,000 sentences from news crawling and 300,000 from Wikipedia.
- The *VOA*[2] news dataset with 277,000 sentences.
- The *Persian Dependency Treebank*[3] with 29,982 sentences of average length of

[1] `http://corpora2.informatik.uni-leipzig.de/download.html`
[2] `http://www.ling.ohio-state.edu/~jonsafari/corpora/index.html#persian`
[3] `http://dadegan.ir/en/perdt/`

16.61 from contemporary Persian-language text [Rasooli *et al.*, 2013]. It contains 498,081 words and 37,618 distinct words.

The aggregated corpus, called **PersoSentencesCorpus**, holds more than 1.6 million sentences and seems of adequate size to train the co-occurrence matrix.

As the preprocessing phase, the PersoSentencesCorpus has been normalized and tokenized following the approach proposed in [Feely *et al.*, 2014] that suggests applying a pipeline of useful tools to deal with written Persian. The pipeline starts with PrePer [Seraji, 2013] which maps Arabic specific characters to their Persian Unicode equivalent. In addition, it replaces the full space between a word and its affix with a zero-width-non-joiner character. Then, a Farsi text normalizer [Feely, 2013] omits Arabic and Persian diacritics and unifies variant forms of some Persian characters to a single Unicode representation. Finally, tokenization is performed by using three tokenizers in a cascade: the Farsi verb tokenizer of [Manshadi, 2013], SetPer [Seraji *et al.*, 2012] and tok-tok [Dehdari, 2015].

After normalisation, we have trained the word embeddings using the provided corpus. A window of size 5 in both directions was used to calculate the co-occurrence matrix. Then, only words with a minimum frequency of 15 were selected, resulting in a dictionary of $49,902$ distinct words. The length of the embedding vectors was set to 300. All these hyper-parameters were chosen empirically during an initial evaluation.

The collated corpus (*PersoSentencesCorpus*) cannot be publicly released due to licensing restrictions on some of its parts. However, we have released the Hellinger PCA word embeddings on GitHub[4], which will allow easy replication of our experiments.

3.4.2 ArmanPersoNERCorpus

To create an NE dataset, in collaboration with ArmanSoft[5], we have decided to manually annotate NEs in a subset of the **BijanKhan**[6] [Bijankhan *et al.*, 2011] corpus which is the most-established tagged Persian corpus, yet lacking entity annotation[7]. We selected the

[4] `https://github.com/HaniehP/PersianNER`

[5] `http://armansoft.ir`

[6] `http://ece.ut.ac.ir/dbrg/bijankhan/`

[7] As an ongoing work, it was faster to complete annotating NEs in the ArmanPersoNERCorpus, in collaboration with ArmanSoft company, than to start collecting NEs from Persian web resources, like

subset from news sentences since they are the most entity-rich. Before the annotation, a comprehensive manual was designed based on the definition of Sekine's extended named entities [Sekine, 2007] adapted to the Persian Language. The annotation task was led by an experienced lead annotator who instructed the front-end annotators (two native post-graduate students) and revised their annotations. The guidelines were very clear and we expected minimal subjectivity. We have verified this hypothesis in two ways: by a sample of 500 already annotated NEs (single or multi-token) chosen randomly, and by another sample of 500 already annotated NEs (single or multi-token) from the two most semantically-close classes (location and organization). Both samples were revised by three other, independent native annotators and the percentages of corrections have been only 1.8% and 1.9%, respectively.

All NEs have been annotated in IOB2 format and are categorized into six classes: *person*, *organization* (such as banks, ministries, embassies, teams, nationalities, networks and publishers), *location* (such as cities, villages, rivers, seas, golfs, deserts and mountains), *facility* (such as schools, universities, research centers, airports, railways, bridges, roads, harbors, stations, hospitals, parks, zoos and cinemas), *product* (such as books, newspapers, TV shows, movies, airplanes, ships, cars, theories, laws, agreements and religion), and *event* (such as wars, earthquakes, national holidays, festivals and conferences); *other* are the remaining tokens. It is worth noting that annotation was not trivial since individual tokens have been categorized according to the context. For instance, *"Tokyo"* is a different type of entity in sentence *"$Tokyo_{loc}$ is a beautiful city"* versus sentence *"$London_{org}$ and $Tokyo_{org}$ sign flight agreement"*. As another example, *"Ferdowsi"* has different labels in *"$Ferdowsi_{B\text{-}ORG}$ $University_{I\text{-}ORG}$"* and *"$Ferdowsi_{B\text{-}PER}$, the great epic poet"*. Having multiple labels for the same tokens, which makes NER more challenging, seem to be more frequent in Persian than in English.

The annotated dataset, **ArmanPersoNERCorpus**, contains 7,682 sentences (considering the full-stop as the sentence terminator). The histogram of the sentences' length is shown in figure 3.2. The mode is around 24 words per sentence, but with a significant tail of longer sentences. It contains 250,015 tokens with a hit rate of 87.68% in the trained

Wikipedia articles in Persian language, by using silver standard approaches. Moreover, as demanded by the industry partner, the NE annotated dataset needed to be completely accurate.

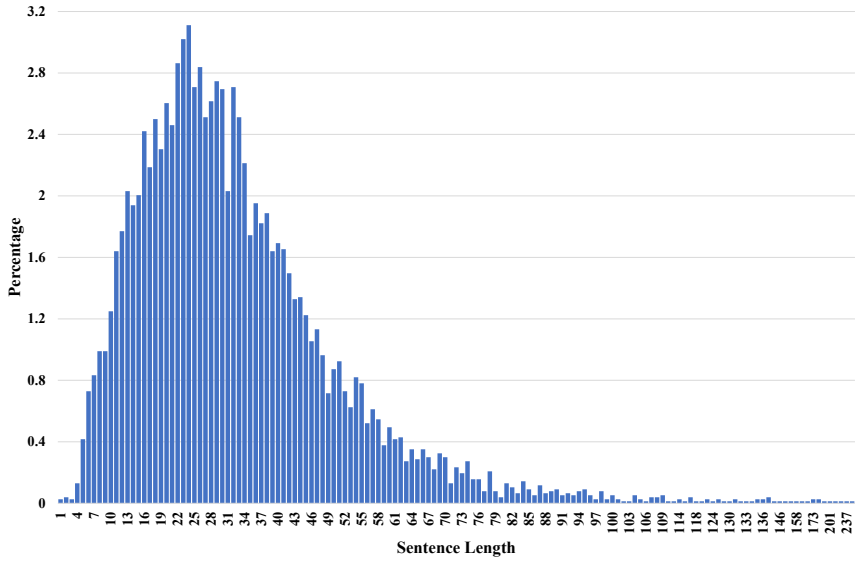

Figure 3.2: Histogram of the sentences' length in ArmanPersoNERCorpus.

Entity type	**Person**	**Organization**	**Location**	**Facility**	**Event**	**Product**	**Other**
Number of Tokens (NT)	5,215	10,036	4,309	1,486	2,519	1,463	224,987
Percentage	2.08%	4.01%	1.72%	0.59%	1.00%	0.58%	89.98%
Number of Unique-Tokens (NUT)	1,829	1,289	834	548	555	634	15,676
Percentage (NUT/NT)	35.07%	12.84%	19.35%	36.87%	22.03%	43.33%	6.96%

Table 3.1: Class percentages in ArmanPersoNERCorpus.

dictionary. Only 11.08% of the tokens are marked as part of entities. More than 60% of the sentences have at least one entity of any type. Figure 3.3 shows the percentage of sentences containing at least one entity and a maximum between 1 and 7 entities of each particular class. The most frequent NE class is *organisation* with an appearance rate of more than 33% of the sentences. This is followed by *person* and *location* with more than 25%. *Event* and *product* are far less frequent with just over 6% and *facility* has the lowest frequency with about 4%. Table 3.1 summarizes the number of tokens for each entity class in ArmanPersoNERCorpus. Moreover, Table 3.2 compares the size of ArmanPersoNERCorpus with four standard NE datasets of other languages including English and German from CoNLL-2003 [Tjong Kim Sang and De Meulder, 2003], and Spanish and Dutch from CoNLL-2002 [Tjong Kim Sang, 2002].

Figure 3.4 shows a snapshot of the dataset together with an English transliteration of

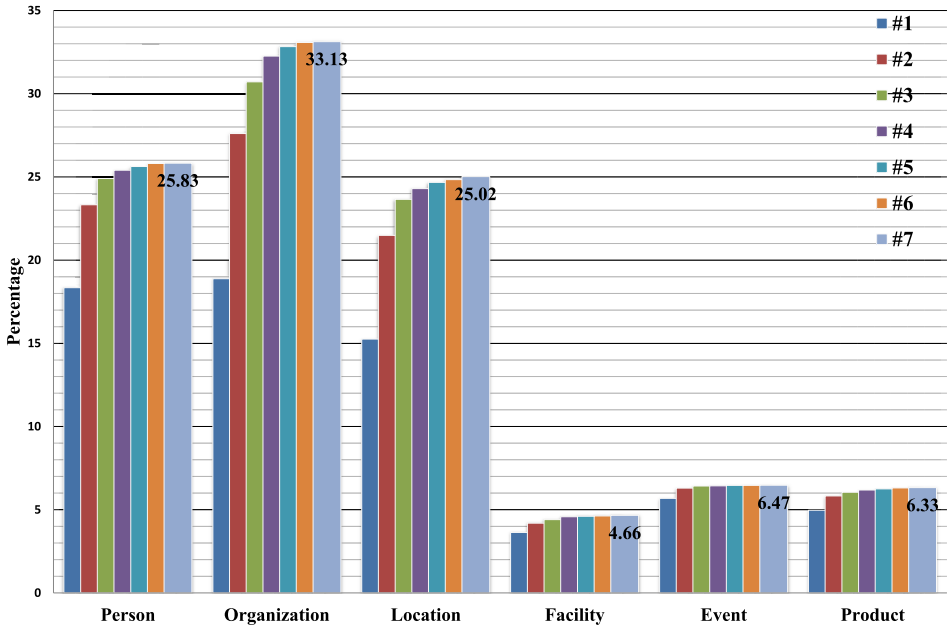

Figure 3.3: Percentage of sentences containing at least one entity and maximum $p = \{1, \ldots, 7\}$ entities (in left-to-right order in the plot) of each particular named-entity class.

	Persian	English	German	Spanish	Dutch
Sentences	7,682	21,921	18,933	11,755	23,898
Tokens	250,015	301,418	310,318	369,171	309,684

Table 3.2: Comparing the size of ArmanPersoNERCorpus as the first manually annotated Persian NER dataset versus popular NER datasets in other languages.

the tokens. Each line contains five tab-separated columns. In order from left to right, they are ez$\bar{a}$fe, POS-tag, inflexion, token and NER-tag. The first three columns are inherited from the BijanKhan corpus. Ez$\bar{a}$fe [8] is a grammatical particle in the Persian language that connects words of a phrase, usually noun-phrase, together. It is pronounced as an unstressed i vowel between the linked words, but generally not indicated in writing.

The dataset is organised in the same 3 folds that we have used for the experiments and stored on GitHub[9]. It can be used to train NER systems in future research on Persian NER, but it also offers an ideal test set for evaluation of NER systems trained on silver

[8] https://en.wikipedia.org/wiki/Ezafe
[9] https://github.com/HaniehP/PersianNER

Ezāfe	POS-tag	Inflexion	Token	NER-tag	Transliteration
O	N	N,SING,SURN	سید	B-PERS	Seyed
EZ	N	N,SING,PR,GEN	محمود	I-PERS	Mahmoud
O	N	N,SING,PR	محدث	I-PERS	Mohadess
EZ	N	N,SING,COM,GEN	مدیر	O	manager
EZ	N	N,SING,COM,GEN	اکتشاف	O	discovery
EZ	N	N,SING,COM,GEN	شرکت	B-ORG	Company
EZ	AJ	ADJ,SIM,GEN	ملی	I-ORG	National
EZ	N	N,SING,COM,GEN	نفت	I-ORG	Oil
O	N	N,SING,LOC,PR	ایران	I-ORG	Iranian
O	P	P	در	O	in
O	N	N,SING,COM	مصاحبه	O	interview
O	P	P	با	O	with
EZ	AJ	ADJ,SIM,GEN	واحد	B-ORG	Unit
EZ	AJ	ADJ,SIM,GEN	مرکزی	I-ORG	Central
O	N	N,SING,COM	خبر	I-ORG	News
O	P	P	با	O	with
EZ	N	N,SING,COM,GEN	اعلام	O	declaring
O	DET	DET	این	O	this
O	N	N,SING,COM	خبر	O	announcement
O	V	V,PA,SIM,POS,3	افزود	O	adds
O	PUNC	DELM	:	O	:
O	P	P	با	O	With
EZ	N	N,PL,COM,GEN	حفاریهای	O	diggings
O	AJ	ADJ,CMPR	بیشتر	O	more
O	P	P	در	O	in
EZ	N	N,SING,LOC,GEN	میدان	B-LOC	field
EZ	AJ	ADJ,SIM,GEN	نفتی	I-LOC	oil
O	N	N,SING,LOC,PR	چنگوله	I-LOC	Changuleh
O	N	N,SING,COM	انتظار	O	expectation
O	V	V,PRS,POS,4	داریم	O	have
EZ	N	N,PL,COM,GEN	ذخائر	O	reservoirs
O	DET	DET	این	O	In
O	N	N,SING,LOC	میدان	O	field
O	N	N,SING,COM	افزایش	O	increase
O	V	V,SUB,POS,3	یابد	O	will
O	PUNC	DELM	.	O	.

Figure 3.4: A snapshot of ArmanPersoNERCorpus.

standards (which is left as future work and is not covered in this thesis).

3.5 Experiments

In this section, we report NER results based on the PersoSentencesCorpus and Arman-PersoNERCorpus datasets. The classification task is challenging given the much lower frequencies of the entity classes versus the non-entity class (*other*), as shown in Table 3.1. For this task, we have not used any of the additional linguistic information that is available from the dataset (such as POS tag, inflexion etc).

To calculate the co-occurrence matrix, C, we have used a context window of radius 5. The size of the dictionary, V, from the PersoSentencesCorpus is $|V| = 49,902$ and that of subset D is $D = 7,099$, obtained by selecting only the words with count greater than 15. The word embedding matrix $h(C)$ has been computed by heuristically setting $m = 300$. For classification, each word has been encoded as a 3-gram that includes the previous and following feature vectors. All the models used for classification share the same word embeddings.

For classification, we have compared the proposed SVM-HMM with a CRF and a deep learning approach based on the Jordan-RNN [Mesnil *et al.*, 2013]. For the SVM-HMM we have used structural SVM from [Joachims, 2008] with a Markov chain of order 3 and learning constant $C = 0.5$. The CRF is from the HCRF library [Morency *et al.*, 2010] and is trained with an $L2$ regularizer of weight 100. The Jordan-RNN is a recurrent neural network from [Mesnil *et al.*, 2013] trained with 100 hidden states and initialized using the same features vectors. All parameters were chosen by 3-fold cross-validation over a reasonable range of values. The indices for the three folds are available in the dataset to allow for future result comparison. We have also tried continuous bag of words [Mikolov *et al.*, 2013a], skip-grams [Mikolov *et al.*, 2013a] and GloVe [Pennington *et al.*, 2014] as embeddings, and the Elman-RNN [Mesnil *et al.*, 2013] as classifier, but results have proved generally less accurate.

Table 3.3 shows the comparison of the average MUC7 and CoNLL scores from the 3-fold cross-validation for the three classifiers. The MUC7 and CoNLL scores are F_1 values adapted to the NER task, with the CoNLL score generally stricter than MUC7 [Nadeau

	Entities													
	Person		**Organization**		**Location**		**Facility**		**Event**		**Product**		**Overall**	
Methods	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL	MUC7	CoNLL
CRF	76.98	64.10	60.59	42.25	66.98	57.97	61.46	41.09	59.98	22.48	33.75	20.00	60.89	49.92
Jordan-RNN	79.13	72.13	67.31	57.28	69.90	62.70	63.49	51.92	62.30	39.79	49.50	**42.08**	68.53	60.52
SVM-HMM	**82.40**	**75.65**	**71.65**	**61.59**	**72.92**	**66.67**	**72.22**	**61.20**	**71.63**	**52.58**	**50.90**	41.37	**72.59**	**65.13**

Table 3.3: F_1 score comparison between three different classifiers based on MUC7 and CoNLL score functions for NER task on ArmanPersoNERCorpus. The F_1 score achieved by structural SVM is higher overall and for all classes but one, with the Jordan-RNN as the second best.

and Sekine, 2007]. As shown in Table 3.3, the scores achieved by the SVM-HMM are higher overall and for all classes but one, with the Jordan-RNN as the second best. To verify statistical significance, we have also run a paired t-test over the results from the six individual classes and confirmed statistical significance of the differences even at $p = 0.02$. The relative ranking between SVM-HMM and the CRF is supported by similar results in the literature, including [Nguyen and Guo, 2007; Tang *et al.*, 2013; Lei *et al.*, 2014], showing that regularized minimum-risk classifiers tend to outperform equivalent models trained under maximum conditional likelihood. The relative ranking between SVM-HMM and the RNN is instead somehow in contrast with the recent results in the literature, and a possible explanation for it is the relatively small size of the dataset compared to the number of free parameters in the models. We plan future comparative experiments with larger corpora to further probe this assumption.

Although we have achieved reasonable accuracy on Persian NER, the results are approximately 20 percentage points lower than the best results for English NER. We believe a challenge is that the size of the Persian NER corpus is $1/3$ of the sentences of the English dataset, as compared in Table3.2. Moreover, having multiple labels for the same tokens, which makes NER more challenging, seem to be more frequent in Persian than in English.

3.6 Conclusion

In this chapter, we have presented and released ArmanPersoNERCorpus, the first manually-annotated Persian NE dataset, and proposed an NER pipeline for the Persian language. The main components of the pipeline are word embedding by Hellinger PCA and classifi-

cation by a structural SVM-HMM classifier. Experiments conducted over the ArmanPersoNERCorpus dataset have achieved promising overall F_1 scores of 72.59 (MUC7) and 65.13 (CoNLL), higher than those of a CRF and a Jordan-RNN. The released dataset can be used for further development of Persian NER systems and for evaluation of systems trained on silver-standard corpora, and the achieved accuracy will provide a baseline for future comparisons.

Chapter 4

BiLSTM-CRF for Persian Named-Entity Recognition

As described in the previous chapter, named-entity recognition (NER) can still be regarded as work in progress for a number of Asian languages due to the scarcity of annotated corpora. In this chapter, we present a performing approach for Persian NER based on a deep learning architecture. To this end, we used the two datasets introduced in Section 3.4. Moreover, we release a number of word embeddings (including GloVe, skip-gram, CBOW and Hellinger PCA) trained on the sizable collation of Persian text, *PersoSentencesCorpus*, which is introduced in Section 3.4.1. The combination of the deep learning architecture (a BiLSTM-CRF) and the pre-trained word embeddings has allowed us to achieve a 77.45% CoNLL $F1$ score, a result that is more than 12 percentage points higher than the best previous result and interesting in absolute terms.

4.1 Introduction

Named-entity recognition (NER) is a natural language processing component that aims to identify all the "named entities" (NEs) such as names of people, locations, organisations and numerical expressions in an unstructured text. This information can be useful in its own right or facilitate higher-level NLP tasks such as text summarization and machine translation. To date, NER research has mostly focussed on languages with a high number of digitally annotated resources such as English and German [Tjong Kim Sang

and De Meulder, 2003] and Spanish and Dutch [Tjong Kim Sang, 2002]. The main reason why many other languages, including many from the Asian region, have received less attention is the significant scarcity of public, annotated digital resources. Amongst those, the Persian language is spoken by more than 110 million speakers world-wide and has more than 570K articles on Wikipedia. However, it has been rarely studied for NER [Khormuji and Bazrafkan, 2014] or even just text processing [Shamsfard, 2011].

Although language-agnostic NER systems such as Polyglot-NER [Al-Rfou *et al.*, 2015] exist, their performance is generally not competitive in comparison to language-specific NER. For this reason, in our previous work [Poostchi *et al.*, 2016] we developed a dedicated NER system for Persian[1]. Its development was supported by two datasets: a) a sizable unannotated dataset of Persian sentences for training word embeddings, and b) an entity-annotated dataset for training named-entity classifiers.

This chapter makes three distinct contributions: 1) it officially releases the entity-annotated dataset, *ArmanPersoNERCorpus*, with an ISLRN[2] that should make its utilisation easier; 2) it releases four different word embeddings trained on the unannotated resources, *PersoSentencesCorpus*, for a comprehensive Persian dictionary of nearly 50K unique words, also available via an ISLRN[3]. We have released all four word embeddings on GitHub[4]; 3) it proposes a deep learning Persian NER based on a state-of-the-art architecture, the BiLSTM-CRF [Huang *et al.*, 2015; Lample *et al.*, 2016]. Thanks to this architecture and the trained word embeddings, we have been able to achieve an improvement of over 12 percentage points of CoNLL $F1$ score over our previous approach based on structural SVM [Poostchi *et al.*, 2016].

4.2 Methods

Supervised NER is split into an initial step of word embedding followed by a step of token-level classification of the named entities. In this section we briefly describe the methods employed.

[1]Particularly, Western/Iranian Persian which is also known as Farsi.
[2]ISLRN: 399-379-640-828-6
[3]ISLRN: 921-509-141-609-6
[4]`https://github.com/HaniehP/PersianNER`

4.2.1 Word Embedding

A word embedding maps distinct words to high-dimensional feature vectors. GloVe [Pennington *et al.*, 2014], word2vec [Mikolov *et al.*, 2013a], and Hellinger PCA (HPCA) [Lebret and Collobert, 2014] are well-known examples of unsupervised word embeddings used successfully for NER.

GloVe is a global log-bilinear regression model with a weighted least-square objective that combines advantages of global matrix factorization and local context windows. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence.

Word2vec is a generative model for continuous representations of words that preserves the linear regularities amongst words. This model has two variants described hereafter: 1) the *skip-gram* model aims to learn word vector representations that are useful for predicting the nearby words in a sentence. A shallow neural network consisting of an input projection layer, an output layer and a softmax activation is trained to maximize the average of the log probability of a context word surrounding a given word; 2) the *continuous bag of words (CBOW)* model is similar to the skip-gram except that the roles of the input and output are reversed: in this model, the probability of the current word given the context is explicitly estimated.

HPCA is a simple spectral method analogous to PCA. First, the co-occurrence matrix is normalised row-by-row to represent the words by proper discrete probability distributions. Then, the resulting matrix is transformed into Hellinger space before applying PCA to reduce its dimensionality[5].

4.2.2 The BiLSTM-CRF for Sequential Labelling

The BiLSTM-CRF is a recurrent neural network obtained from the combination of a long short-term memory (LSTM) and a conditional random field (CRF) [Huang *et al.*, 2015; Lample *et al.*, 2016]. The LSTM is used first to process each sentence token-by-token and produce an intermediate representation. Then, this intermediate representation is used as input for the CRF to provide the prediction of all the tokens' labels. The two models enjoy

[5]We use the same implementation as in Chapter 3.

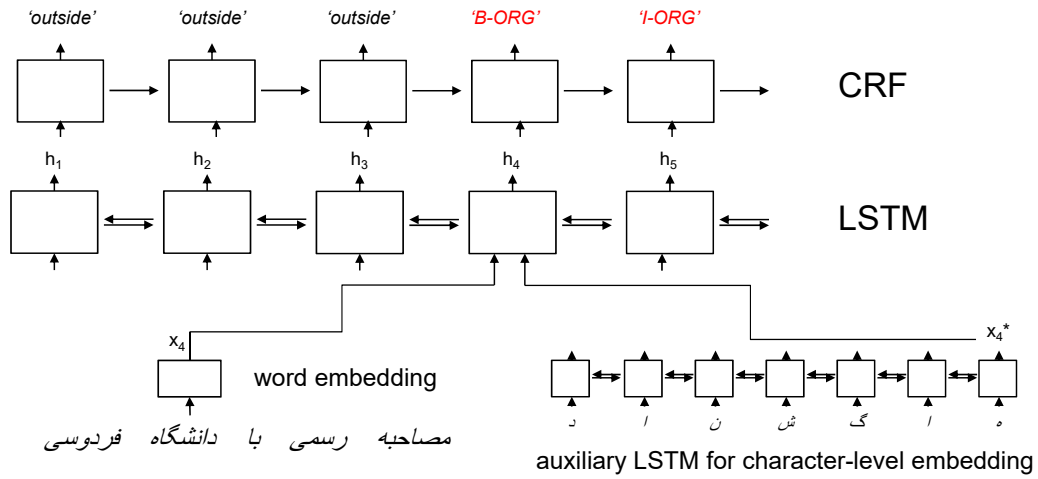

Figure 4.1: A diagram of the BiLSTM-CRF with an example of prediction. The input is a Persian sentence that consists of 5 tokens and translates into English as "an official interview with Ferdowsi University". The sentence is displayed in right-to-left order since this is how it would appear in Persian writing. However, this is not important for processing since both the tokens and their characters are processed in both directions. Token "University" is the 4-th token and its word embedding is noted as x_4 in the diagram. Its character-level embedding is the last output of the auxiliary LSTM and is noted as x_4^*. These embeddings are concatenated and used as the input of the corresponding slot in the main LSTM. In turn, the output of the LSTM slot is noted as h_4 and used as input for the CRF. Eventually, the CRF slot emits prediction "B-ORG". Token "Ferdowsi" is the 5-th token in the sentence and is predicted as "I-ORG".

complementary features: as a complex, nonlinear model, the LSTM is able to effectively capture the sequential relationships amongst the input tokens; at its turn, the CRF permits optimal, joint prediction of all the labels in the sentence, capturing the relationships at label level. The "bi" in the name stands for "bidirectional" and alludes to the fact that the LSTM processes each sentence in both left-to-right and right-to-left order to embed the sequential dependencies in both directions. Before being processed, each token needs to be converted to a high-dimensional numerical vector, and this embedding is learned automatically alongside all the other parameters as part of the training stage. Eventually, the network also includes a second, auxiliary LSTM that further encodes each token character-by-character to capture the regularities at character level. Prior to being processed, also the individual characters need to be mapped to numerical embeddings. Figure 4.1 shows a complete diagram of the BiLSTM-CRF with an ample caption describing all the main variables and components (the character embeddings have been omitted to

avoid cluttering).

Given a training set of labelled sequences, $\{x_i, y_i\}, i = 1 \dots N$, where x denotes a sequence of tokens and y the sequence of their labels, the BiLSTM-CRF is trained by maximizing the conditional log-likelihood:

$$\bar{w} = \underset{w}{\operatorname{argmax}} \sum_{i=1}^{N} \ln p(y_i|x_i, w) \tag{4.1}$$

where w denotes all the model's parameters including the transition weights of the CRF, the weights of the main and auxiliary LSTMs, and the token and character embeddings. Once the model is trained, inference for a new sentence x is obtained as:

$$\bar{y} = \underset{y}{\operatorname{argmax}}\, p(y|x, \bar{w}) \tag{4.2}$$

by propagating x through the network and applying the Viterbi algorithm at the CRF output layer.

4.3 Experimental Results

In this section, we present the NER results obtained with the BiLSTM-CRF and the different word embedding and we compare them with those reported in [Poostchi *et al.*, 2016]. For the experiments, we have used a TensorFlow implementation of the BiLSTM-CRF [6] [Dernoncourt *et al.*, 2017], running each training session for 80 epochs (a value where the validation accuracy always seemed to have stabilised). For processing, all digits have been replaced with 0s. All hyper-parameters have been left to their default values.

Table 4.1 shows a comparison of the CoNLL $F1$ scores (by class and as overall micro-average) over the NER task for the various classifiers. The CoNLL $F1$ score is a strict version of the standard $F1$ score where a true positive is scored only if all the tokens of a given named entity are classified correctly (including their B- and I- tags). Conversely, every incorrect B- prediction is counted as a false positive. All the experiments have been performed with three-fold cross validation, using each of the three folds in turn as the test set and the other two for training. Moreover, each experiment has been repeated three

[6] `https://github.com/Franck-Dernoncourt/NeuroNER`

	Entities						
Methods	**Person**	**Organization**	**Location**	**Facility**	**Event**	**Product**	**Overall**
CRF (HPCA)	64.10	42.25	57.97	41.09	22.48	20.00	49.92
Jordan-RNN (HPCA)	72.13	57.28	62.70	51.92	39.79	42.08	60.52
SVM-HMM (HPCA)	75.65	61.59	66.67	61.20	52.58	41.37	65.13
BiLSTM-CRF (HPCA)	77.69	69.70	69.67	57.33	52.69	49.24	69.43
BiLSTM-CRF (CBOW)	87.32	74.84	76.06	66.38	56.93	55.06	76.19
BiLSTM-CRF (GloVe)	86.97	75.73	76.62	67.41	55.58	55.08	76.58
BiLSTM-CRF (Skip-Gram)	**88.18**	**76.03**	**76.94**	**70.47**	**60.12**	**55.69**	**77.45**

Table 4.1: Comparison of Persian NER results with different classifiers and word embeddings (by class and as overall micro-average). The results above the double horizontal line are from (Poostchi et al., 2016) and are based on the same data and splits.

times to mollify the effects of the random initialisation of the network's weights. This means that the values reported in Table 4.1 for our system are the average of $3 \times 3 = 9$ runs.

As shown in Table 4.1, the scores achieved by the BiLSTM-CRF have been higher than any of the results previously presented in [Poostchi *et al.*, 2016]. The results with the different word embeddings have ranged from a minimum average of 69.43% $F1$ score with HPCA to a maximum of 77.45% with the skip-gram. This relative ranking seems in good accordance with other NER results from the literature [Huang *et al.*, 2015; Ma and Hovy, 2016]. Moreover, the RNN that was used in chapter 3 (Jordan-RNN) is much simpler than the BiLSTM-CRF and therefore its performance can be expected to be limited. Amongst the classes, *person* is clearly the easiest and *product* the most challenging. This could be explained by the fact that the latter has much fewer samples (6% vs 25% of sentences) or that its patterns are possibly more diverse and harder to learn. In all cases, the proposed system has managed to outperform the best previous results for all classes and by a remarkable 12.32 $F1$ score percentage points on average.

4.4 Conclusion

In this chapter, we have presented an approach for Persian NER based on a deep learning architecture and released a Persian annotated corpus alongside four different Persian word embeddings based on GloVe, CBOW, skip-gram and HPCA. The proposed approach has achieved an average $F1$ score of 77.45% which, to the best of our knowledge, is the highest Persian NER $F1$ score reported in the literature by 12.32 percentage points over

the previous best result. Moreover, in addition to NER, the released word embeddings could find future use in other Persian NLP tasks including translation, question answering and summarisation.

Chapter 5

BiLSTM-SSVM: Training the BiLSTM to Minimize the CoNLL Loss

Building on the achievements of the BiLSTM-CRF in named-entity recognition (NER), this chapter introduces the BiLSTM-SSVM, an equivalent neural model where training is performed using a structured hinge loss. The typical loss functions used for evaluating NER are entity-level variants of the F_1 score such as the CoNLL and MUC losses. Unfortunately, the common loss function used for training NER - the cross entropy - is only loosely related to the evaluation losses. For this reason, in this chapter we propose a training approach for the BiLSTM-CRF that leverages a hinge loss bounding the CoNLL loss from above. In addition, we present a mixed hinge loss that bounds either the CoNLL loss or the Hamming loss based on the density of entity tokens in each sentence. The experimental results over four benchmark languages (English, German, Spanish and Dutch) show that training with the mixed hinge loss has led to small but consistent improvements over the cross entropy across all languages and four different evaluation measures.

5.1 Introduction

The main aim of classifier training is to find a parametrization minimizing the expectation of a chosen loss function. However, the loss functions commonly used for evaluation such as the 0-1 loss are discontinuous in parameter space, non-convex and flat over large regions. For this reason, the common approach is to optimize an alternative function,

referred to as *surrogate loss*, instead of the chosen loss. The most well-known surrogates are the logistic loss and the hinge loss which are both upper bounds of the 0-1 loss. In the deep learning community, the logistic loss is also known as *cross entropy* (or negative log-likelihood) and is the de facto objective for training.

The typical functions used for evaluating named-entity recognition (NER) and other sequential labeling tasks are derivatives of the F_1 score and include the CoNLL and MUC scores [Nadeau and Sekine, 2007]. The relationship of the corresponding losses with the cross entropy is loose. However, the hinge loss can still be defined as a formal upper bound for all of them. For this reason, in this chapter we propose an approach for training NER using a hinge loss that bounds the CoNLL loss from above.

Given its inherently sequential nature, NER is often tackled by sequential classifiers such as linear-chain conditional random fields and recurrent neural networks. In particular, the BiLSTM-CRF [Huang *et al.*, 2015; Lample *et al.*, 2016] is a high-performing deep learning architecture obtained from the combination of a long short-term memory (LSTM) and a conditional random field (CRF) as the output layer. The addition of the CRF permits efficient, structured prediction of the entire sequence of labels and tends to increase the classification accuracy [Huang *et al.*, 2015; Lample *et al.*, 2016; Ma and Hovy, 2016]. Following the achievements of the BiLSTM-CRF, in this chapter we propose the *BiLSTM-SSVM*, an equivalent neural model where training is performed using the structural support vector machine (SSVM) [Joachims, 2005]. The crux of this method is the solution of a special inference problem known as the "loss-augmented inference". This inference returns the sequence of labels maximizing the sum of the score and the chosen loss, and it is needed in order to ensure that the training objective acts as an upper bound on the loss. The loss-augmented inference is straightforward if both the scoring and loss functions decompose over the individual labels in the sequence: however, this is not the case for the CoNLL loss and any other entity-based loss. Therefore, this chapter presents a novel dynamic programming algorithm to address this case. Overall, our main contributions are:

- A training approach for the BiLSTM-CRF based on the minimization of a structured hinge loss (BiLSTM-SSVM). This loss can usefully bound the evaluation losses commonly employed for NER such as CoNLL and MUC;

- A dynamic programming algorithm for the loss-augmented inference with the CoNLL loss. This algorithm is presented in Section 5.4 and a proof of optimality is given in Section 5.5;

- Experimental results on NER over four benchmark languages (English, German, Dutch and Spanish) showing that the proposed approach has led to slight yet consistent improvements over the conventional cross-entropy training, across all the tested languages and four different evaluation measures (CoNLL, MUC, entity segmentation F_1 score and entity classification F_1 score).

The rest of this chapter is organized as follows: Section 5.2 describes the main related work. Section 5.3 recaps the BiLSTM-CRF and introduces the BiLSTM-SSVM. Section 5.4 presents the proposed algorithm for the loss-augmented inference under the CoNLL loss and introduces the mixed hinge. Section 5.5 contains the proof of optimality of the proposed loss-augmented inference algorithm. Section 5.6 describes the experiments and results. Eventually, Section 5.7 concludes the chapter.

5.2 Related Work

Most of the NER approaches proposed in recent years leverage recurrent neural networks (RNNs), and a selection is briefly reviewed in the following. One of the main initial works in this area is [Collobert *et al.*, 2011] that proposed an architecture made of a convolutional network and a CRF output layer to be used for chunking, POS tagging and NER. The model is trained by using stochastic gradient ascent to maximize the cross entropy, and an implicit word embedding is automatically learned in the network's early layers from a random or external initialization. Recurrent neural networks such as the Elman and Jordan RNNs have also been used for sequential labeling [Mesnil *et al.*, 2013; Mesnil *et al.*, 2015]. The Elman RNN is similar to the feed-forward neural networks, except that the output of the hidden layer at slot $t-1$ is fed back into the input at slot t. Conversely, in the Jordan RNN it is the output layer that feeds back into the input.

To capture the properties of both RNNs and CRFs for sequential tagging, Huang et al. [2015] have combined the bidirectional LSTM with a CRF output layer. In this model,

the LSTM is used first to process each sentence token-by-token and produce an intermediate representation. Then, the intermediate representation is used as input for the CRF to provide the joint prediction of all the labels. Lample et al. [2016] have extended this model with a second, auxiliary LSTM encoding each token character-by-character to capture the regularities at character level. This extended model, that we refer to simply as BiLSTM-CRF in the following, had reported state-of-the-art NER accuracy for several benchmark languages at the time. Since then, it has been adopted widely as a strong baseline for comparison. Variants have also been proposed such as the BiLSTM-CNN-CRF [Ma and Hovy, 2016] where the auxiliary LSTM is replaced by a CNN.

More recently, sequential labeling has extensively incorporated neural language models (LM) to improve the token encoding. Peters et al. [2017] have proposed TagLM, a hierarchical RNN model where pre-trained, bidirectional LM embeddings are concatenated with the output of the first bidirectional RNN layer. In ELMo [Peters *et al.*, 2018], the bidirectional LM embeddings are obtained from the aggregation of all the internal layers of a deep bidirectional LM. Liu et al. [2018] have proposed the LM-LSTM-CRF, where transfer learning and multi-task learning are used to extract character-level representations. In this model, the objective functions of the LM and the sequence labeling are minimized jointly. In a similar vein, in [Rei, 2017] the objective function includes the prediction of the previous word, the current label and the next word in the sequence.

As an alternative to the cross entropy, the hinge loss has been used as the training objective in a number of works. The Recurrent SVM is an LSTM trained using an SVM objective [Zhang *et al.*, 2016], where the parameters of the LSTM and the SVM are learned jointly using a combination of sequence-level and frame-level regularized hinge losses. The model has been evaluated on the Windows phone task for speech recognition and has yielded improvement over a standard LSTM. The RSVM, a combination of RNN and structured SVM, has been proposed in [Shi *et al.*, 2016] for slot tagging in spoken language understanding. This network improves the discriminative capability of an RNN by optimizing a sequence-level max-margin training criterion [Tsochantaridis *et al.*, 2005]. The training times take advantage of the fact that the hinge loss can be an identical zero for some training samples, and therefore such samples do not need to be involved in the parameter updates. Similarly, [Adi *et al.*, 2017] has proposed the DeepSegmentor,

a neural model for speech segmentation composed of an RNN and a structured prediction output layer that are trained jointly using a structured loss function (the combined duration).

The loss-augmented inference, too, has received significant attention in the literature. In an early work, Joachims et al. [2005] proposed SVM^{perf}, a support vector method for loss functions such as the Hamming loss, the F_1 loss, the precision-recall break-even point, the precision and recall at *k*, and the ROC area. However, this method only addresses independent and identically distributed (i.i.d.) data, and the extension to structured prediction is challenging because of the dependencies within the scoring function. For the structured prediction case, Suzuki et al. [2006]) have proposed a training algorithm for the F_1 loss that approximates the decision loss with a softmax; Rosenfeld et al. [2014] have proposed an approximate inference algorithm for the AUC loss based on a linear programming relaxation; Haffari et al. [2015] have proposed an approximate algorithm for the F_1 loss leveraging a dual decomposition and alternate optimizations; and Zhang et al. [2017] have proposed an exact, exhaustive algorithm for the loss-augmented inference under the F_1 loss, yet not for the multi-class case required by NER. Closely inspired by this algorithm, in this chapter we instead propose an algorithm for multi-class loss-augmented inference under the CoNLL loss.

Other structured surrogate loss functions that have found use in the literature include the structured probit loss [McAllester and Keshet, 2011], the structured ramp loss (also called truncated hinge loss) [McAllester and Keshet, 2011; Gimpel and Smith, 2012; Huang *et al.*, 2014] and the structured orbit loss [Karmon and Keshet, 2015]. Other work has proposed the direct minimization of the evaluation loss by means of asymptotic equalities [McAllester *et al.*, 2010; Song *et al.*, 2016]. However, their adoption to date has been more limited.

5.3 Sequential Labeling

Sequential labeling aims to predict a sequence of class labels, $y = \{y_1 \dots y_t \dots y_T\}$, from a corresponding sequence of measurements, $x = \{x_1 \dots x_t \dots x_T\}$. It is a very common task in NLP for applications such as chunking, POS tagging, supertagging and NER.

A widespread model for sequential labeling is the hidden Markov model (HMM) that factorizes the joint probability of the measurements and the labels, $p(x, y)$, by arranging the latter in a Markov chain (of order one or above) and conditioning the measurement at frame t on only the corresponding label. For an HMM of order one, $p(x, y)$ is expressed as:

$$p(x, y) = p(y_1) \prod_{t=2}^{T} p(y_t|y_{t-1}) \prod_{t=1}^{T} p(x_t|y_t) \tag{5.1}$$

where $p(y_1)$ is the probability of the initial class, terms $p(y_t|y_{t-1})$ are the transition probabilities and terms $p(x_t|y_t)$ are the emission, or measurement, probabilities. With limited modifications (exponential family for the emission probabilities, denormalized factors), this model becomes the widely-used CRF, allowing for discriminative training [Sutton and McCallum, 2012].

5.3.1 BiLSTM-CRF (Cross Entropy)

The BiLSTM-CRF is a recurrent neural network obtained from the combination of an LSTM and a CRF [Huang *et al.*, 2015; Lample *et al.*, 2016]. These two models enjoy complementary features: as a complex, nonlinear model, the LSTM can effectively capture the sequential relationships amongst the input tokens. In turn, the CRF permits optimal, joint prediction of all the labels in the sentence, capturing the relationships at label level. In the BiLSTM-CRF, the posterior probability of label sequence y given input sequence x can be expressed as:

$$p(y|x) = \frac{\exp(F(x, y))}{\sum_{u \in Y} \exp(F(x, u))} \tag{5.2}$$

where F is the scoring function of the BiLSTM-CRF and Y is the set of all possible predictions. Given a training set of labelled sequences, $\{x_i, y_i\}, i = 1 \dots N$, the BiLSTM-CRF can be trained by minimizing the cross entropy (or negative conditional log-likelihood):

$$\bar{w} = \underset{w}{\operatorname{argmin}} - \sum_{i=1}^{N} \ln p(y_i|x_i, w) \tag{5.3}$$

where we have made explicit the dependence on the model's parameters, w, including the transition weights of the CRF, the weights of the main and auxiliary LSTMs, and the

embeddings of all tokens and characters. Replacing (5.2) in (5.3), we obtain:

$$\bar{w} = \underset{w}{\operatorname{argmin}} - \sum_{i=1}^{N} [F(x_i, y_i; w) - \ln \sum_{u \in Y} \exp(F(x, u; w)] \tag{5.4}$$

showing that the optimal parameters are a trade-off between the score assigned to the true labeling and the log-sum-exp of the scores of all the possible labelings. Once the model is trained, inference for a new sentence x is obtained as:

$$\bar{y} = \underset{y}{\operatorname{argmax}}\, p(y|x, \bar{w}) = \underset{y}{\operatorname{argmax}}\, F(x, y; \bar{w}) \tag{5.5}$$

by propagating x through the network and applying the Viterbi algorithm at the CRF output layer.

5.3.2 BiLSTM-SSVM (Hinge Loss)

Most NLP tasks rely on dedicated performance measures for evaluation. Examples include the BLEU score for machine translation, the ROUGE score for summarization, and the MUC and CoNLL scores for NER [Adi and Keshet, 2016]. It is therefore tempting to train the classifier to explicitly minimize such evaluation losses in the hope of obtaining higher predictive accuracy [Suzuki *et al.*, 2006; Haffari *et al.*, 2015; Zhang *et al.*, 2017]. However, a risk of bias toward certain predictions exists and the proof is ultimately empirical. Hereafter, we consider NER and adopt the surrogate loss of SSVM [Joachims, 2005], namely the *structured hinge loss*, which can be made a convex upper bound for any chosen decision loss [Vapnik, 1995]. Given a training set, $\{x_i, y_i\}$, $i = 1 \dots N$, and a decision loss, $\Delta(y_i, y)$, the hinge loss for the i-th sample is defined as:

$$\begin{aligned} l_i &= \max_{y \in Y} \big[- F(x_i, y_i) + F(x_i, y) + \Delta(y_i, y) \big]_+ \\ &= \big[- F(x_i, y_i) + F(x_i, y_i^*) + \Delta(y_i, y_i^*) \big]_+ \end{aligned} \tag{5.6}$$

where $[\pi]_+ = \max\{\pi, 0\}$ and

$$y_i^* = \underset{y \in Y}{\operatorname{argmax}}\, F(x_i, y) + \Delta(y_i, y). \tag{5.7}$$

Equation (5.6) shows that the hinge loss is different from zero if the score assigned to the ground-truth labeling does not surpass that of any other labeling by an amount equal to the decision loss itself. The value of the hinge loss is therefore set by a 'most violating' labeling that becomes the explicit target for the margin. Tsochantaridis *et al* [2005] have proven that this is a sufficient condition for the hinge loss to bound the decision loss from above. The inference in (5.7) is commonly referred to as "loss-augmented inference" and is the core of structural SVM. In the case of scores and decision losses that can be computed token-by-token (such as the 0-1 loss or the Hamming loss), a Viterbi algorithm with appropriate weights can still be used to compute the loss-augmented inference [Tsochantaridis *et al.*, 2005]. However, there are two standing problems in using the CoNLL loss for the loss-augmented inference: 1) similarly to the F_1 loss, the CoNLL loss is based on precision and recall and is therefore not decomposable over single tokens; and 2) the evaluation of correct and incorrect predictions inherently spans multiple tokens. Our main contribution, presented in the following section, is a dynamic programming algorithm that solves the loss-augmented inference in the case of the CoNLL loss.

5.4 Loss-Augmented Inference Under the CoNLL Loss

The CoNLL score is an F_1 score specialized for the NER task. In this score, a prediction is counted as a true positive only if the entire named entity mention is segmented and classified correctly. The tokens are commonly annotated in the IOB2 format (which stands for *inside*, *outside*, *beginning* of an entity) and with their true class label. Since the CoNLL score is normalized between zero and one, the CoNLL loss can be naturally defined as its complement to one. In the following, we present a dynamic programming algorithm for multi-class sequential labeling that solves the loss-augmented inference of (5.7) under this loss.

The CoNLL loss is a decision loss and, as such, it only depends on the classification contingency table, i.e., the values of TP, FN, TN, and FP. Given a ground-truth labeling y^g, the number of true entities in a sentence is fixed and known, and it can be

expressed as $P = TP + FN$. Accordingly, the CoNLL loss can be written as:

$$\Delta_{CoNLL}(y^g, y) = 1 - \frac{2\,pre \times rec}{pre + rec}, \quad pre = \frac{TP}{TP + FP}, \;\; rec = \frac{TP}{P} \tag{5.8}$$

showing that (5.8) depends on the prediction only via the value of TP and $(TP + FP)$. The loss-augmented inference aims to find the y^* labeling with the highest sum of score and loss. Following [Zhang *et al.*, 2017], we can approach this problem in two steps: 1) finding the labelings maximizing the score alone for fixed values of TP and $(TP + FP)$, and 2) finding the best of all these labelings. Given that the loss is solely a function of TP and $(TP + FP)$, this ensures finding the required maximum.

The modified Viterbi algorithm of [Zhang *et al.*, 2017] extends the notion of Viterbi state at slot t with the TP and $(TP + FP)$ counts up to that slot. If these counts can be incremented token-by-token as in [Zhang *et al.*, 2017], the update rules of the algorithm are relatively simple. However, with the CoNLL loss a true positive prediction can only be resolved at the end of a chunk, which can span multiple tokens, and therefore the required update rules are substantially more complicated. We present the proposed dynamic programming algorithm in Algorithms 1 and 2, with a description hereafter. A proof of optimality is provided in Section 5.5.

In the proposed algorithm, prediction y is developed in left-to-right order along the sequence. Thus, the TP and $(TP+FP)$ counts can only increment or remain unchanged. The state of the partial solution at slot t is indicated with $(TP, (TP + FP), y_t)$ and it consists of the predicted label for the current token, y_t, and the TP and $(TP+FP)$ counts up to the current token. The sequence itself is noted as $seq(TP, (TP + FP), y_t)$ and the scoring function for a sequence is noted as $F(seq)$. The induction step is as follows: at slot t, the partial solution is obtained by extending a number of the partial solutions at slot $t - 1$ with the current prediction, y_t and the corresponding increment of TP and $(TP + FP)$. The domain for y_t is $L = \{$O, B-PER, ... B-LOC, I-PER, ... I-LOC$\}$, even in the case where this gives place to an invalid IOB annotation (for instance, an I-x label following an O one) since the run-time model predicts in a similarly unconstrained way. We use shorthand notations B-x and I-x to mean one of the B and I labels, and notations

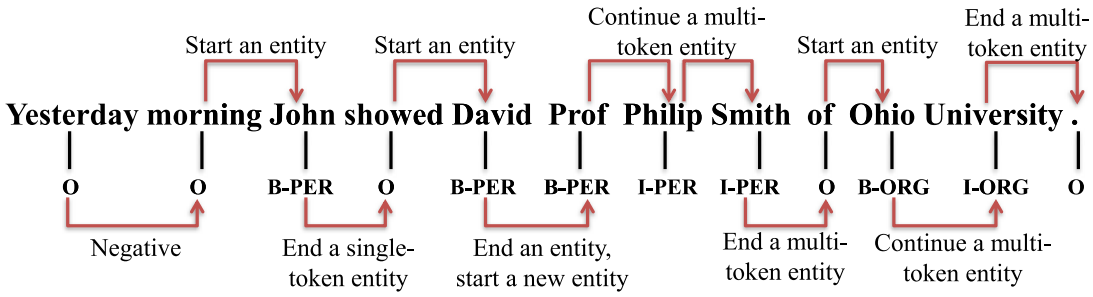

Figure 5.1: A pseudo-sentence illustrating all the cases of label transitions.

B-* and I-* to mean any of them.

All the update rules in Algorithm 1 depend on the values of the current true label, y_t^g, and the immediately previous, y_{t-1}^g. Figure 5.1 shows examples of possible label transitions. To facilitate an understanding of the update rules, we have explicitly named the label transitions as:

- 'Negative': transition from O to O;
- 'Start an entity': transition from O to B-*;
- 'Continue a multi-token entity': transition from B-x or I-x to I-x;
- 'End an entity, start a new entity': transition from B-* or I-* to B-*;
- 'End a multi-token entity: transition from I-* to O;
- 'End a single-token entity': transition from B-* to O.

For more clarification, here we describe the update rules in the 'Negative' case:

- predicting y_t as O is a true negative and increments neither TP nor $(TP + FP)$;
- predicting y_t as B-* starts a false positive and, as a consequence, it increases the $(TP + FP)$ count by one;
- predicting y_t as I-x with $y_{t-1} \in$ {B-x,I-x} means that the prediction of a multi-token entity is continuing and neither TP nor $(TP + FP)$ is incremented;
- instead, predicting y_t as I-x with $y_{t-1} \in L \setminus$ {B-x,I-x} starts a new false positive and, therefore, $(TP + FP)$ is increased by one.

Algorithm 1 Finding the argmax of equation (7) for every possible value of TP and $(TP + FP)$.

procedure FINDBESTSEQUENCES
$T =$ length (y^g);
y^g,$y \in$ L = {O, B-PER, ..., B-LOC, I-PER, ..., I-LOC}
$TP_{max} = 0$;
$(TP + FP)_{max} = 0$;
for $t = 1 : T$ **do**
$(TP + FP)_{max} + +$;
if ENDOFENTITY(y^g_{t-1}, y^g_t) **then**
$TP_{max} + +$;
end if
for each $y_t \in L$ **do**
switch Transition(y^g_{t-1}, y^g_t) **do**
case (O→O) // **Negative**
for $TP = 0 : TP_{max}$ **do**
for $(TP + FP) = 0 : (TP + FP)_{max} - 1$ **do**
if $(y_t ==$ O**) then**
$seq(TP, (TP + FP), y_t) = \text{argmax}_{y_{t-1} \in L} F\Big(seq(TP, (TP + FP), y_{t-1})$,O$\Big)$;
else if $(y_t ==$ B-*$)$ **then**
$seq(TP, (TP + FP) + 1, y_t) = \text{argmax}_{y_{t-1} \in L} F\Big(seq(TP, (TP + FP), y_{t-1})$,B-*$\Big)$;
else if $(y_t ==$ I-*$)$ **then**
$seq(TP, (TP + FP), y_t) = \text{argmax}_{y_{t-1} \in \{\text{B-*,I-*}\}} F\Big(seq(TP, (TP + FP), y_{t-1})$,I-*$\Big)$;
$seq(TP, (TP + FP) + 1, y_t) = \text{argmax}_{y_{t-1} \in L \setminus \{\text{B-*,I-*}\}} F\Big(seq(TP, (TP + FP), y_{t-1})$,I-*$\Big)$;
end if
end for \\ loop over $(TP + FP)$
end for \\ loop over TP

Algorithm 1 (Continued)

case (O→B-x) //**Start an entity**

for $TP = 0 : TP_{max}$ **do**

for $(TP + FP) = 0 : (TP + FP)_{max} - 1$ **do**

if $(y_t ==$ O**)** **then**

$$seq(TP, (TP + FP), y_t) = \text{argmax}_{y_{t-1} \in L} F\Big(seq(TP, (TP + FP), y_{t-1}), \text{O}\Big);$$

else if $(y_t ==$ B-*$)$ **then**

$$seq(TP, (TP + FP) + 1, y_t) = \text{argmax}_{y_{t-1} \in L} F\Big(seq(TP, (TP + FP), y_{t-1}), \text{B-*}\Big);$$

else if $(y_t ==$ I-*$)$ **then**

$$seq(TP, (TP + FP), y_t) = \text{argmax}_{y_{t-1} \in \{\text{B-*,I-*}\}} F\Big(seq(TP, (TP + FP), y_{t-1}), \text{I-*}\Big);$$

$$seq(TP, (TP + FP) + 1, y_t) = \text{argmax}_{y_{t-1} \in L \setminus \{\text{B-*,I-*}\}} F\Big(seq(TP, (TP + FP), y_{t-1}), \text{I-*}\Big);$$

end if

if $(y_t ==$ B-x$)||(y_t ==$ I-x**)** **then**

$$inseq(TP, (TP+FP)+1) = \text{argmax} \begin{bmatrix} F\Big(\text{argmax}_{y_{t-1} \in L} F\big(seq(TP, (TP + FP), y_{t-1}), \text{B-x}\big)\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{\text{B-x,I-x}\}} F\big(seq(TP, (TP + FP), y_{t-1}), \text{I-x}\big)\Big) \end{bmatrix};$$

end if

end for \\ loop over $(TP + FP)$

end for \\ loop over TP

case (B-x→I-x) || (I-x→I-x) //**Continue a multi-token entity**

for $TP = 0 : TP_{max}$ **do**

for $(TP + FP) = 0 : (TP + FP)_{max} - 1$ **do**

if $(y_t ==$ O**)** **then**

$$seq(TP, (TP + FP), y_t) = \text{argmax}_{y_{t-1} \in L} F\Big(seq(TP, (TP + FP), y_{t-1}), \text{O}\Big);$$

else if $(y_t ==$ B-*$)$ **then**

$$seq(TP, (TP + FP) + 1, y_t) = \text{argmax}_{y_{t-1} \in L} F\Big(seq(TP, (TP + FP), y_{t-1}), \text{B-*}\Big);$$

Algorithm 1 (Continued)

else if ($y_t ==$ I-*) **then**

$seq(TP, (TP+FP), y_t) = \text{argmax}_{y_{t-1} \in \{\text{B-*,I-*}\}} F\Big(seq(TP, (TP+FP), y), y_{t-1}\Big)$;

$seq(TP, (TP+FP)+1, y_t) = \text{argmax}_{y_{t-1} \in L \setminus \{\text{B-*,I-*}\}} F\Big(seq(TP, (TP+FP), y), y_{t-1}\Big)$;

if ($y_t ==$ I-x) **then**

$inseq(TP, (TP+FP)) = (inseq(TP, (TP+FP))$,I-x);

end if

end if

end for \\ loop over $(TP+FP)$

end for \\ loop over TP

case (B-*→B-x) || (I-*→B-x) //**End an entity, start a new entity**

for $TP = 0 : TP_{max}$ **do**

for $(TP+FP) = 0 : (TP+FP)_{max} - 1$ **do**

if ($y_t ==$ O) **then**

$$seq(TP, (TP+FP), y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP-1, (TP+FP))\text{,O}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1}\}} F\big(seq(TP, (TP+FP), y_{t-1})\text{,O}\big)\Big) \end{bmatrix};$$

$$seq(TP{+}1, (TP{+}FP), y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP, (TP+FP))\text{,O}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1}\}} F\big(seq(TP+1, (TP+FP), y_{t-1})\text{,O}\big)\Big) \end{bmatrix};$$

else if ($y_t ==$ B-*) **then**

$$seq(TP, (TP{+}FP){+}1, y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP-1, (TP+FP))\text{,B-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1}\}} F\big(seq(TP, (TP+FP), y_{t-1})\text{,B-*}\big)\Big) \end{bmatrix};$$

$$seq(TP{+}1, (TP{+}FP){+}1, y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP, (TP+FP))\text{,B-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1}\}} F\big(seq(TP+1, (TP+FP), y_{t-1})\text{,B-*}\big)\Big) \end{bmatrix};$$

Algorithm 1 (Continued)

else if ($y_t ==$ I-*) **then**

$seq(TP,(TP+FP),y_t) = \text{argmax}_{y_{t-1}\in\{\text{B-*,I-*}\}} F\Big(seq(TP,(TP+FP),y_{t-1}),\text{I-*}\Big)$;

$seq(TP,(TP+FP)+1,y_t) =$

$$\text{argmax}\begin{bmatrix} F\Big(inseq(TP-1,(TP+FP)),\text{I-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1},\text{B-*,I-*}\}} F\big(seq(TP,(TP+FP),y_{t-1}),\text{I-*}\big)\Big) \end{bmatrix};$$

$seq(TP+1,(TP+FP)+1,y_t) =$

$$\text{argmax}\begin{bmatrix} F\Big(inseq(TP,(TP+FP)),\text{I-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1},\text{B-*,I-*}\}} F\big(seq(TP+1,(TP+FP),y_{t-1}),\text{I-*}\big)\Big) \end{bmatrix};$$

end if

if ($y_t ==$ B-x)||($y_t ==$ I-x) **then**

$$inseq(TP,(TP+FP)+1) = \text{argmax}\begin{bmatrix} F\Big(inseq(TP-1,(TP+FP)),\text{B-x}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L} F\big(seq(TP,(TP+FP),y_{t-1}),\text{B-x}\big)\Big) \\ F\Big(inseq(TP-1,(TP+FP)),\text{I-x}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{\text{B-x,I-x}\}} F\big(seq(TP,(TP+FP),y_{t-1}),\text{I-x}\big)\Big) \end{bmatrix};$$

end if

end for \\ loop over $(TP+FP)$

end for \\ loop over TP

case (I-x→O) //**End a multi-token entity**

for $TP = 0 : TP_{max}-1$ **do**

for $(TP+FP) = 0 : (TP+FP)_{max}-1$ **do**

if ($y_t ==$ O) **then**

$$seq(TP,(TP+FP),y_t) = \text{argmax}\begin{bmatrix} F\Big(inseq(TP-1,(TP+FP)),\text{O}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1}\}} F\big(seq(TP,(TP+FP),y_{t-1}),\text{O}\big)\Big) \end{bmatrix};$$

Algorithm 1 (Continued)

$$seq(TP{+}1,(TP{+}FP),y_t) = \text{argmax}\begin{bmatrix} F\Big(inseq(TP,(TP+FP)),\text{O}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1}\}} F\big(seq(TP+1,(TP+FP),y_{t-1}),\text{O}\big)\Big) \end{bmatrix};$$

else if ($y_t ==$ B-*) **then**

$$seq(TP,(TP{+}FP){+}1,y_t) = \text{argmax}\begin{bmatrix} F\Big(inseq(TP-1,(TP+FP)),\text{B-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1}\}} F\big(seq(TP,(TP+FP),y_{t-1}),\text{B-*}\big)\Big) \end{bmatrix};$$

$$seq(TP+1,(TP+FP)+1,y_t) = \text{argmax}\begin{bmatrix} F\Big(inseq(TP,(TP+FP)),\text{B-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1}\}} F\big(seq(TP+1,(TP+FP),y_{t-1}),\text{B-*}\big)\Big) \end{bmatrix};$$

else if ($y_t ==$ I-*) **then**

$$seq(TP,(TP+FP)+1,y_t) = \text{argmax}\begin{bmatrix} F\Big(inseq(TP-1,(TP+FP)),\text{I-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1},\text{B-*},\text{I-*}\}} F\big(seq(TP,(TP+FP),y_{t-1}),\text{I-*}\big)\Big) \end{bmatrix};$$

$$seq(TP+1,(TP+FP)+1,y_t) = \text{argmax}\begin{bmatrix} F\Big(inseq(TP,(TP+FP)),\text{I-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1},\text{B-*},\text{I-*}\}} F\big(seq(TP+1,(TP+FP),y_{t-1}),\text{I-*}\big)\Big) \end{bmatrix};$$

if ($y_t ==$ I-x) **then**

$$seq(TP,(TP+FP),y_t) = \text{argmax}\begin{bmatrix} F\Big(inseq(TP,(TP+FP)),\text{I-x}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in\{\text{B-x,I-x}\}} F\big(seq(TP,(TP+FP),y_{t-1}),\text{I-x}\big)\Big) \end{bmatrix};$$

else if ($y_t ==$ I-y) **then**

$$seq(TP,(TP+FP),y_t) = \text{argmax}_{y_{t-1}\in\{\text{B-y,I-y}\}} F\Big(seq(TP,(TP+FP),y_{t-1}),\text{I-y}\Big);$$

end if

end if

Algorithm 1 (Continued)

end for \\ loop over $(TP + FP)$

end for \\ loop over TP

case (B-x→O) //**End a single-token entity**

for $TP = 0 : TP_{max} - 1$ **do**

for $(TP + FP) = 0 : (TP + FP)_{max} - 1$ **do**

if $(y_t ==$ O**)** **then**

$$seq(TP, (TP + FP), y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP - 1, (TP + FP)),\text{O}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1}\}} F\big(seq(TP, (TP + FP), y_{t-1}),\text{O}\big)\Big) \end{bmatrix};$$

$$seq(TP{+}1, (TP{+}FP), y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP, (TP + FP)),\text{O}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1}\}} F\big(seq(TP + 1, (TP + FP), y_{t-1}),\text{O}\big)\Big) \end{bmatrix};$$

else if $(y_t ==$ B-*) **then**

$$seq(TP, (TP{+}FP){+}1, y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP - 1, (TP + FP)),\text{B-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1}\}} F\big(seq(TP, (TP + FP), y_{t-1}),\text{B-*}\big)\Big) \end{bmatrix};$$

$$seq(TP + 1, (TP + FP) + 1, y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP, (TP + FP)),\text{B-*}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1}\}} F\big(seq(TP + 1, (TP + FP), y_{t-1}),\text{B-*}\big)\Big) \end{bmatrix};$$

else if $(y_t ==$ I-*) **then**

$$seq(TP, (TP + FP), y_t) = \text{argmax}_{y_{t-1} \in \{\text{B-*,I-*}\}} F\Big(seq(TP, (TP + FP), y_{t-1}),\text{I-*}\Big);$$

if $(y_t ==$ I-y**)** **then**

$$seq(TP, (TP + FP) + 1, y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP - 1, (TP + FP)),\text{I-y}\Big) \\ F\Big(\text{argmax}_{y_{t-1} \in L \setminus \{y^g_{t-1},\text{B-y,I-y}\}} F\big(seq(TP, (TP + FP), y_{t-1}),\text{I-y}\big)\Big) \end{bmatrix};$$

Algorithm 1 (Continued)

$$seq(TP+1,(TP+FP)+1,y_t) = \text{argmax} \begin{bmatrix} F\Big(inseq(TP,(TP+FP)),\text{I-y}\Big) \\ F\Big(\text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1},\text{B-y,I-y}\})}\, F\big(seq(TP+1,(TP+FP),y_{t-1}),\text{I-y}\big)\Big) \end{bmatrix};$$

else if ($y_t ==$ I-x) **then**

$seq(TP,(TP+FP)+1,y_t) = \text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1},\text{B-x,I-x}\}}\, F\Big(seq(TP,(TP+FP),y_{t-1}),\text{I-x}\Big)$;

$seq(TP+1,(TP+FP)+1,y_t) = \text{argmax}_{y_{t-1}\in L\setminus\{y^g_{t-1},\text{B-x,I-x}\}}\, F\Big(seq(TP+1,(TP+FP),y_{t-1}),\text{I-x}\Big)$;

end if

end if

end for \\ loop over $(TP+FP)$

end for \\ loop over TP

end for \\ loop over y_t

end for \\ loop over t

end procedure

Algorithm 2 Finding the argmax in equation (5.7) - final loop.

procedure FINALLOOP
  $best = -\infty$
  $best_{sequence} = [\,]$
  for $TP = 0 : P$ **do**
    for $(TP + FP) = 0 : |y^g|$ **do**
      for each $y_T \in L$ **do**
        $value = F\Big(seq(TP, (TP + FP), y_T)\Big) + \Delta_{CoNLL}\Big(y^g, seq(TP, (TP + FP), y_T)\Big)$
        if $value > best$ **then**
          $best = value$
          $best_{sequence} = seq(TP, (TP + FP), y_T)$
        end if
      end for
    end for
  end for
end procedure

The update rules are fully determined by the values of y_t^g, y_{t-1}^g, y_t, and y_{t-1} and they are completely independent of the specific class set (in number and type). Therefore, the proposed algorithm can be used for any NER task. The algorithm is highly articulated since its cases stem from the combination of multiple variables: however, its computational complexity is only approximately quadratic in the length of the sentence and therefore manageable even for sentences of a few hundred tokens.

Algorithm 1 returns all the sequences of highest score for $TP = 0 \dots P, (TP+FP) = 0 \dots |y^g|$. After that, Algorithm 2 simply searches exhaustively over such best sequences for the one maximizing the sum of the score and the loss, y^*.

5.4.1 Mixed Hinge

Despite our emphasis, using the CoNLL loss as a training objective may not prove optimal for test-time performance. The CoNLL score is of the F_1 family and, as such, it is a meaningful performance measure when the positive samples (i.e., the named-entity mentions) are relatively few [Joachims, 2005]. Conversely, the Hamming loss is more meaningful in more balanced cases. This suggests exploring a *mixed hinge* loss, where the CoNLL loss is minimized for sentences where the entity density is below a chosen threshold, and the Hamming loss is minimized otherwise. The entity density is simply computed as the percentage of entity tokens in the sentence, and the threshold is tuned using cross-validation.

5.5 Proof of Optimality for the Loss-Augmented Inference Algorithm

The algorithm described in Algorithms 1, 2 returns the labeling maximizing (5.7) under the CoNLL loss. Since a proof was not given in [Zhang *et al.*, 2017], we provide an original proof hereafter.

Proposition: Algorithms 1, 2 return the labeling maximizing loss-augmented inference (5.7).

Proof: Let us consider a function, $f(y)$, of labeling y, and another function, $g(s(y))$,

that is a function of y only through a set of sufficient statistics, s (e.g., a decision loss). The sufficient statistics take value over a finite integer interval, $s \in [0 \dots N]$ for reference, and, as such, g only takes at most $N + 1$ distinct values.

Let us now consider:

$$\begin{aligned} \bar{y}_k &= \underset{y \in Y}{\operatorname{argmax}} f(y) \\ s.t.\ & s(y) = k,\ \forall k \in [0 \dots N] \end{aligned} \tag{5.9}$$

This maximization returns a set of $N + 1$ arguments of constrained maxima, one for each value of the sufficient statistics. Therefore,

$$\bar{y} = \underset{y \in \{\bar{y}_1, \dots, \bar{y}_N\}}{\operatorname{argmax}} f(y) + g(s(y)) \tag{5.10}$$

returns the desired maximum. For ease of reference, in (5.7) $f(y)$ is scoring function $F(x_i, y)$, and $g(s(y))$ is decision loss $\Delta(y_i, y)$, which is a function of y only through sufficient statistics TP and $(TP + FP)$. Algorithm 1 implements (5.9) while Algorithm 2 implements (5.10). This algorithm is viable for reasonably small values of N. □

5.6 Experiments and Results

We have run experiments over four well-known NER datasets: English and German from CoNLL-2003 [Tjong Kim Sang and De Meulder, 2003], and Spanish and Dutch from CoNLL-2002 [Tjong Kim Sang, 2002][1].

For the experiments, we have used a publicly-available TensorFlow implementation of the BiLSTM-CRF [Dernoncourt *et al.*, 2017] [2] instead of Lample et al. [2016]' Theano implementation since we needed features of this environment (the inference of y_i^* in (5.7) is performed by an external, compiled function and stored in the computational graph at run time). Each training session was run until convergence of the evaluation loss function (CoNLL) over the development set or a maximum of 120 epochs. All hyper-parameters

[1] Three anomalously long sentences containing only numerical values in the Spanish training set, of respective length 1238, 314, and 261 tokens, and three long sentences containing only mathematical equations in the Dutch training set, of respective length 859, 708, and 454, have been split into shorter sentences of length 155 ± 25.

[2] `https://github.com/Franck-Dernoncourt/NeuroNER`

Cross Entropy	$l_{CrossEntropy} = -\sum_{i=1}^{N} \log p(y_i \vert x_i)$
Hinge-Hamming	$l_{Hinge-Hamming} = -\sum_{i=1}^{N}[-F(x_i, y_i) + F(x_i, y_i^*) + \Delta_{Hamming}(y_i, y_i^*)]_+$ $y_i^* = \text{argmax}_y F(x_i, y) + \Delta_{Hamming}(y_i, y)$
Hinge-CoNLL	$l_{Hinge-CoNLL} = -\sum_{i=1}^{N}[-F(x_i, y_i) + F(x_i, y_i^*) + \Delta_{CoNLL}(y_i, y_i^*)]_+$ $y_i^* = \text{argmax}_y F(x_i, y) + \Delta_{CoNLL}(y_i, y)$
Mixed Hinge	$l_{MixedHinge} = \begin{cases} l_{Hinge-CoNLL}, & \text{if } EntityDensity(y_i) \leq th \\ l_{Hinge-Hamming}, & \text{otherwise} \end{cases}$

Table 5.1: The compared training objectives.

are as in [Dernoncourt *et al.*, 2017], and all the digits have been replaced with zeros as pre-processing. The same pre-trained word embeddings used in [Lample *et al.*, 2016] (which were trained by [Lample *et al.*, 2016] using the Spanish Gigaword version 3 [Graff, 2011] for Spanish, the Leipzig corpora collection [Biemann *et al.*, 2007] for Dutch, the German monolingual training data from the 2010 Machine Translation Workshop [Callison-Burch *et al.*, 2010] for German and the English Gigaword version 4 [Parker *et al.*, 2009] for English) have been used for training initialization. In addition to the main set of experiments, we have carried out a comparative experiment with the high-performing ELMo embeddings and the DeLFT implementation of the BiLSTM-CRF [3].

As approaches, we have compared: 1) the conventional cross-entropy training of the BiLSTM-CRF; 2) training with a hinge loss bounding the Hamming loss from above (a straightforward and well-known loss-augmented inference); 3) training with a hinge loss bounding the CoNLL loss based on the proposed algorithm; and 4) training with a mixed hinge loss bounding the CoNLL loss for sentences with $\leq 1\%$ of entity tokens, and the Hamming loss otherwise. The corresponding training objectives are given in Table 5.1.

Performance evaluation has been carried out using four different performance measures: the CoNLL score, the MUC score, an entity segmentation F_1 score, and an entity classification F_1 score. The CoNLL score has been computed using the standard scoring script[4]. The MUC score, too, has been computed using the official scorer[5]. This score is generally higher than the CoNLL score since the predictions are evaluated separately in terms of class and segmentation [Chinchor, 1998]. To compute the segmentation F_1 score,

[3] https://github.com/kermitt2/delft

[4] publicly available at https://www.clips.uantwerpen.be/conll2002/ner/bin/conlleval.txt

[5] publicly available at https://catalog.ldc.upenn.edu/docs/LDC2001T02/MUC_scorer3.3/

Experiment	English	German	Spanish	Dutch
Cross Entropy [Lample *et al.*, 2016]	90.94	78.76	85.75	81.74
Cross Entropy	90.412 ± 0.245	82.156 ± 0.331	84.312 ± 0.637	77.967 ± 0.564
Hinge-Hamming	90.406 ± 0.196	82.322 ± 0.471	83.815 ± 0.357	77.835 ± 0.542
Hinge-CoNLL	90.294 ± 0.303	81.289 ± 0.355	83.414 ± 0.425	77.387 ± 0.416
Mixed Hinge	$\mathbf{90.497 \pm 0.149}$	$\mathbf{82.349 \pm 0.318}$	$\mathbf{84.525 \pm 0.276}$	$\mathbf{78.126 \pm 0.417}$

Table 5.2: Comparison of the CoNLL scores with the different loss functions.

we have taken all the entity tokens (both ground truth and predicted) and changed their class to a single, notional class (say, X). In this way, the classes are treated only as B-X, I-X and O, and running the CoNLL scorer assesses the segmentation quality alone. The classification F_1 score has been computed by instead ignoring all the B- and I- prefixes, using the CoNLL scorer with the provided '-r' option. The separate evaluation of the segmentation and classification performance is likely to shed more light on the compared models.

Table 5.2 shows a comparison of the CoNLL scores obtained with the different loss functions. These results are computed over the test set at the epoch with the best dev-set score. To marginalize the effects of the random initialization and the dropout, we have repeated each experiment 10 times and reported the average and the standard deviation. Moreover, since the loss-augmented inference is sensitive to the scale of the loss [Joachims, 2005], we have carried out an initial sensitivity analysis over each dataset to find out an appropriate scale for the Hamming and CoNLL losses.

The results above the double horizontal line are from [Lample *et al.*, 2016] that performed training with the cross entropy. As can be seen, the scores we achieved with the cross entropy differ from those reported in [Lample *et al.*, 2016] due to the different software implementation of the BiLSTM-CRF [Dernoncourt *et al.*, 2017]. However, we believe that such differences are not relevant since the goal of this chapter is only to adopt a strong baseline architecture for comparison of the training losses.

Table 5.2 shows that the scores achieved by the mixed hinge loss have been slightly higher than those of the cross entropy across all four languages (from 0.08 percentage points for English to 0.21 for Spanish). The scores achieved by the hinge-Hamming loss alone have been generally worse than those of the cross entropy (mildly higher for

Experiment	English	German	Spanish	Dutch
Cross Entropy	93.375 ± 0.157	85.363 ± 0.304	90.808 ± 0.354	86.201 ± 0.404
Hinge-Hamming	93.393 ± 0.182	85.481 ± 0.465	90.571 ± 0.210	86.276 ± 0.332
Hinge-CoNLL	93.327 ± 0.179	84.475 ± 0.275	90.277 ± 0.248	85.745 ± 0.373
Mixed Hinge	$\mathbf{93.452 \pm 0.094}$	$\mathbf{85.489 \pm 0.299}$	$\mathbf{90.944 \pm 0.187}$	$\mathbf{86.537 \pm 0.252}$

Table 5.3: Comparison of the MUC scores with the different loss functions.

Experiment	English	German	Spanish	Dutch
Cross Entropy	94.837 ± 0.142	85.807 ± 0.328	94.105 ± 0.471	$\mathbf{91.991 \pm 0.410}$
Hinge-Hamming	94.806 ± 0.225	85.925 ± 0.597	94.149 ± 0.290	91.835 ± 0.387
Hinge-CoNLL	94.575 ± 0.152	84.609 ± 0.317	93.813 ± 0.383	91.238 ± 0.508
Mixed Hinge	$\mathbf{94.893 \pm 0.112}$	$\mathbf{86.085 \pm 0.336}$	$\mathbf{94.237 \pm 0.486}$	91.989 ± 0.349

Table 5.4: Comparison of the segmentation F_1 scores with the different loss functions.

German, but worse for English, Spanish and Dutch). In turn, hinge-Hamming has outperformed hinge-CoNLL on all four languages. This result is surprising since the hinge-CoNLL objective is closer to the evaluation measure. Aside from possible overfitting, a plausible explanation is that the Hamming loss is a "finer" loss than CoNLL and can provide a greater range of violating labelings in (5.7). As explained in Section 5.4.1, the mixed hinge aims to capture the best of both losses, by using the CoNLL loss as the training objective for sentences with sparser entities, and the Hamming loss for all others. Its results are a clear improvement over either separate loss, with an average increase of 0.28 points over Hamming and of 0.78 over CoNLL, with one-tailed p-value < 0.01 for both German and Spanish.

Table 5.3 shows the MUC scores obtained with the different loss functions. Again, the highest scores have been achieved by the mixed hinge, with an improvement over the cross entropy ranging from 0.08 percentage points for English to 0.33 for Dutch. As expected, the absolute scores for MUC are higher than the CoNLL scores, but the relative rankings of all the training losses are comparable. In particular, the MUC scores for Dutch are much higher than the corresponding CoNLL scores of Table 5.2, showing that for this language it is harder to attain both entity segmentation and classification accuracy.

To focus on segmentation alone, Table 5.4 reports the segmentation F_1 scores obtained with the different loss functions. On this measure, the mixed hinge loss has achieved higher scores for three languages while the cross entropy has achieved a notionally higher score for Dutch. For English and German, the segmentation scores are similar to the

Experiment	English	German	Spanish	Dutch
Cross Entropy	91.487 ± 0.280	84.011 ± 0.344	86.981 ± 0.589	79.500 ± 0.494
Hinge-Hamming	91.699 ± 0.214	$\mathbf{84.493 \pm 0.532}$	87.001 ± 0.395	80.012 ± 0.428
Hinge-CoNLL	91.618 ± 0.359	83.468 ± 0.389	86.190 ± 0.475	79.297 ± 0.500
Mixed Hinge	$\mathbf{91.748 \pm 0.212}$	84.184 ± 0.264	$\mathbf{87.296 \pm 0.295}$	$\mathbf{80.121 \pm 0.439}$

Table 5.5: Comparison of the classification F_1 scores with the different loss functions.

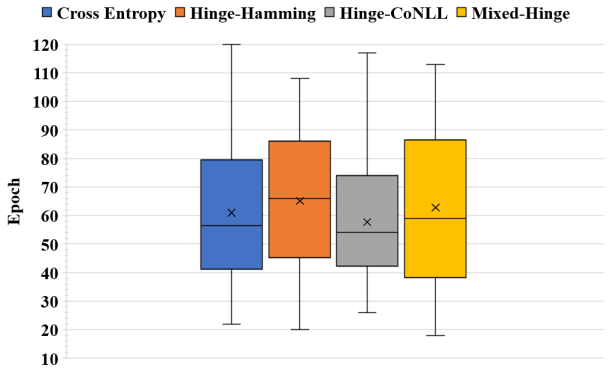

Figure 5.2: Training epoch of maximum validation accuracy for the different loss functions.

MUC scores of Table 5.3. However, for Spanish and Dutch the segmentation scores are much higher than the corresponding MUC scores, showing that classification is relatively harder than segmentation for these two languages.

Eventually, Table 5.5 shows the classification F_1 scores obtained with the different loss functions. Also on this measure, the highest scores have been achieved by the mixed hinge in most cases, with an improvement over the cross entropy ranging from 0.18 percentage points for English to 0.62 for Dutch (one-tailed p-value < 0.05), with an average of 0.32 percentage points across all languages. The relative rankings for the other losses are again similar, but the cross entropy has achieved a more limited performance over Dutch.

As computational times are concerned, training with the cross entropy has proved the fastest, followed by the hinge-Hamming and the hinge-CoNLL, respectively. For instance, training one epoch of the English dataset on a 3.4 GHz Intel Xeon with 32 GB of RAM has taken 177 s with the cross entropy, 279 s with the hinge-Hamming, 464 s with the hinge-CoNLL, and 297 s with the mixed hinge. Figure 5.2 shows the epoch at which each training loss has reached the maximum validation accuracy over all languages

Experiment	English
Cross Entropy [Peters *et al.*, 2018]	92.22 ± 0.10
Cross Entropy	92.076 ± 0.087
Hinge-Hamming	92.026 ± 0.220
Hinge-CoNLL	90.873 ± 0.381
Mixed Hinge	$\mathbf{92.206 \pm 0.136}$

Table 5.6: Comparison of the CoNLL scores over the English dataset with the different loss functions and ELMo word embeddings.

and runs. This figure shows that the rate of convergence is comparable for all training losses. Overall, the total training times have proved manageable in all cases.

In addition to the above experiments, we have performed a comparative experiment over the English dataset using the high-performing ELMo contextualized embeddings [Peters *et al.*, 2018]. For this experiment, we have used the DeLFT BiLSTM-CRF with its default word embeddings (GLoVe-300d [6] and ELMo-1024d [7]). Table 5.6 shows the CoNLL scores obtained with the different loss functions as average of 3 independent runs. These results show that the proposed mixed-hinge loss can achieve slightly higher scores than the cross entropy also with state-of-the-art word embeddings, and higher than those of the Hamming and CoNLL losses in isolation. The results above the double horizontal line are from [Peters *et al.*, 2018] that performed training with the cross entropy and differ from those we reported due to the different software implementation.

5.7 Conclusion

In this chapter, we have presented the BiLSTM-SSVM, a training approach for the BiLSTM-CRF based on a hinge loss minimization. In the approach, the hinge loss is used as an upper bound for three evaluation losses, namely Hamming, CoNLL and a combination of the two. The required loss-augmented inference is challenging in the case of a non-decomposable loss such as CoNLL, and, for this reason, in this chapter we have proposed an articulated dynamic programming algorithm that can perform the loss-augmented inference for the CoNLL loss and any other loss similarly based on entity-level error count-

[6] `http://nlp.stanford.edu/data/glove.840B.300d.zip`
[7] `https://allennlp.org/elmo`

ing. Since the CoNLL loss is of the F_1 type, we have also argued that it may be a promising training objective for sentences with relatively sparse entities. For this reason, we have proposed a training objective that bounds the CoNLL loss for sentences with low entity density, and the Hamming loss otherwise ("mixed hinge"). Experiments conducted over four benchmark languages (English, German, Spanish and Dutch) have shown that training with the mixed hinge loss has achieved slightly higher accuracies than with the cross entropy for all languages. These results suggest that training with objectives closer to the evaluation measures can be an effective strategy, and that using different losses for sentences with different sufficient statistics should be explored further. As such, we plan to soon extend our investigation to a variety of other tasks, losses and architectures.

Chapter 6

Cluster Naming Using Word Embeddings and WordNet's Hypernymy

Cluster labeling is the assignment of representative labels to clusters of documents or words. Once assigned, the labels can play an important role in applications such as navigation, search and document classification. However, finding appropriately descriptive labels is still a challenging task. In this chapter, we propose various approaches for assigning labels to word clusters by leveraging word embeddings and the synonymy and hypernymy relations in the WordNet lexical ontology. Experiments carried out using the WebAP document dataset have shown that one of the approaches stand out in the comparison and is capable of selecting labels that are reasonably aligned with those chosen by a pool of four human annotators. To the best of our knowledge, this work is the first to leverage hypernyms for labeling word clusters with abstract terms.

6.1 Introduction and Related Work

Document collections are often organized into clusters of either documents or words to facilitate applications such as navigation, search and classification. The organization can prove more useful if its clusters are characterized by *sets of representative labels*. The task of assigning a set of labels to each individual cluster in a document organization is

known as cluster labeling [Wang *et al.*, 2014] and it can provide a useful description of the collection in addition to fundamental support for navigation and search.

In Manning et al. [2008], cluster labeling approaches have been subdivided into $i)$ differential cluster labeling and $ii)$ cluster-internal labeling. The former selects cluster labels by comparing the distribution of terms in one cluster with those of the other clusters while the latter selects labels that are solely based on each cluster individually. Cluster-internal labeling approaches include computing the clusters' centroids and using them as labels, or using lists of terms with highest frequencies in the clusters. However, all these approaches can only select cluster labels from the terms and phrases that explicitly appear in the documents, possibly failing to provide an appropriate level of abstraction or description [Lau *et al.*, 2011]. As an example, a word cluster containing words *dog* and *wolf* should not be labeled with either word, but as *canids*. For this reason, in this chapter we explore several approaches for labeling word clusters obtained from a document collection by leveraging the synonymy and hypernymy relations in the WordNet taxonomy [Miller, 1995], together with word embeddings [Mikolov *et al.*, 2013a; Pennington *et al.*, 2014].

A hypernymy relation represents an asymmetric relation between a class and each of its instances. A hypernym (e.g., *vertebrate*) has a broader context than its hyponyms (*bird, fishes, reptiles* etc). Conversely, the contextual properties of the hyponyms are usually a subset of those of their hypernym(s). Hypernymy has been used extensively in natural language processing, including in recent works such as Yu et al. [2015] and HyperVec [Nguyen *et al.*, 2017] that have proposed learning word embeddings that reflect the hypernymy relation. Based on this, we have decided to make use of available hypernym-hyponym data to propose an approach for labeling clusters of keywords by a representative selection of their hypernyms.

In the proposed approach, we first extract a set of keywords from the original document collection. We then apply a step of hierarchical clustering on the keywords to partition them into a hierarchy of clusters. To this aim, we represent each keyword as a real-valued vector using pre-trained word embeddings [Pennington *et al.*, 2014] and repeatedly apply a standard clustering algorithm. For labeling the clusters, we first look up all the synonyms of the keywords and, in turn, their hypernyms in the WordNet hierar-

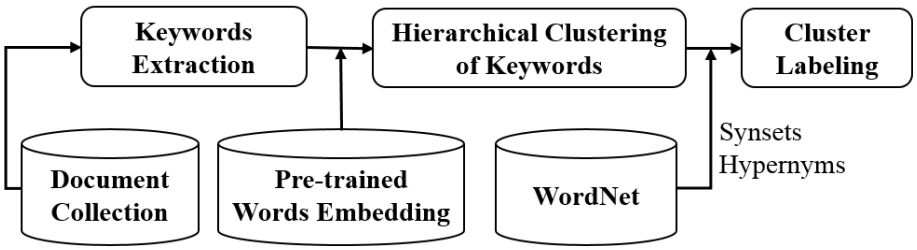

Figure 6.1: The proposed cluster labeling pipeline.

chy. We then encode the hypernyms as word embeddings and use various approaches to select them based on their distance from the clusters' centers. The experimental results over a benchmark document collection have shown that such a distance-based selection is reasonably aligned with the hypernyms selected by four, independent human annotators. As a side result, we show that the employed word embeddings spontaneously contain the hypernymy relation, offering a plausible justification for the effectiveness of the proposed method.

The rest of the chapter is organized as follows. Section 6.2 presents the proposed pipeline of processing steps. Section 6.3 discusses the experimental results and our findings. Finally, Section 6.4 concludes the chapter.

6.2 The Proposed Pipeline

The proposed pipeline of processing steps is shown in Figure 6.1. First, keywords are extracted from each document in turn and accumulated in an overall set of unique keywords. After mapping such keywords to pre-trained word embeddings, hierarchical clustering is applied in a top-down manner. The leaves of the constructed tree are considered as the clusters to be labeled. Finally, each cluster is labeled automatically by leveraging a combination of WordNet's hypernyms and synsets and word embeddings. The following subsections present each step in greater detail.

6.2.1 Keyword Extraction

For the keyword extraction, we have used the rapid automatic keyword extraction (RAKE) of Rose et al. [2010]. This method extracts keywords (i.e., single words or very short word

sequences[1]) from a given document collection and its main steps can be summarized as:

1. Split a document into sentences using a pre-defined set of sentence delimiters.
2. Split sentences into sequences of contiguous words at phrase delimiters to build the candidate set.
3. Collect the set of unique words (W) that appear in the candidate set.
4. Compute the word co-occurrence matrix $X_{|W|\times|W|}$ for W.
5. Calculate word score

$$score(w) = deg(w)/freq(w) \tag{6.1}$$

where,

$$deg(w) = \sum_{i\in\{1,\ldots,|W|\}} X[w,i] \tag{6.2}$$

and

$$freq(w) = \sum_{i\in\{1,\ldots,|W|\}} (X[w,i] \neq 0). \tag{6.3}$$

6. Score each candidate keyword as the sum of its member word scores.
7. Select the top T scoring candidates as keywords for the document.

Alternatively, RAKE can use other combinations of $deg(w)$ and $freq(w)$ as the word scoring function. The keywords extracted from all the documents are accumulated into a set, C, ensuring uniqueness.

6.2.2 Hierarchical Clustering of Keywords

A top-down approach is used to hierarchically cluster the keywords in C. First, each component word of each keyword is mapped onto a numerical vector using pre-trained GloVe50d[2] word embeddings [Pennington *et al.*, 2014]; missing words are mapped to zero vectors. Then, each keyword k is represented with the average vector $\overrightarrow{k}$ of its component

[1]Key-phrase is more appropriate term. However, for the sake of simplicity, we have used the term "keyword" instead of "key-phrase", in this chapter.

[2]`http://nlp.stanford.edu/data/wordvecs/glove.6B.zip`

words. Then, we start from set C as the root of the tree and follow a branch-and-bound approach, where each tree node is clustered into c clusters using the k-means algorithm [Hartigan and Wong, 1979]. A node is marked as a leaf if it contains less than n keywords or it belongs to level d, the tree's depth limit. The leaf nodes are the clusters to be named with a set of verbal terms.

6.2.3 Cluster Labeling

As discussed in Section 6.1, we aim to label each cluster with descriptive terms. The labels should be more general than the cluster's members to abstract the nature of the cluster. To this end, we leverage the hypernym-hyponym correspondences in the lexical ontology. First, for each cluster, we create a large set, L, of candidate labels by including the hypernyms[3] of the component words, expanded by their synonyms, of all the keywords. The synonyms are retrieved from the WordNet's sets of synonyms, called *synsets*. Then, we apply the four following approaches to select l labels from set L:

- *FreqKey*: Choose the l most frequent hypernyms of the l most frequent keywords.
- *CentKey*: Choose the l most central hypernyms of the l most central keywords.
- *FreqHyp*: Choose the l most frequent hypernyms.
- *CentHyp*: Choose the l most central hypernyms.

Approaches *FreqKey* and *FreqHyp* are based on frequencies in the collection. For performance evaluation, we sort their selected labels in descending frequency order. In *CentKey* and *CentHyp*, the centrality is computed with respect to the cluster's center in the embedding space as the average vector of all its keywords $\overrightarrow{K} = \frac{1}{|K|}\sum_{k\in K}\overrightarrow{k}$. The distance between hypernym h and the cluster's center is $d(\overrightarrow{h}, \overrightarrow{K}) = ||\overrightarrow{h} - \overrightarrow{K}||$, where $\overrightarrow{h}$ is the average vector of the hypernym's component words. The labels selected by these two approaches are sorted in ascending distance order.

[3]Nouns only (not verbs).

6.3 Experiments and Results

For the experiments, we have used the WebAP dataset[4] [Keikha *et al.*, 2014] as the document collection. This dataset contains $6,399$ documents of diverse nature with a total of $1,959,777$ sentences and is derived from the 2004 TREC Terabyte Track Gov2 collection [Clarke *et al.*, 2004]. This dataset is well-known for the task of retrieving topically-relevant text passages. In passage retrieving, the search engine looks for a particular passage in a relevant document which explicitly answers a desired query. There are intrinsic existing clusters in this corpus because the passages were retrieved according to 82 different queries. Thus, we select this dataset to ease the evaluation process. Moreover, our industry partner was interested in this dataset for the current and some other tasks.

For the RAKE software[5], the hyper-parameters are the minimum number of characters of each keyword, the maximum number of words of each keyword, and the minimum number of times each keyword appears in the text, and they have been left to their default values of 5, 3, and 4, respectively. Likewise, parameter T has been set to its default value of one third of the words in the co-occurrence matrix. For the hierarchical clustering, we have chosen the specific number of clusters ($c = 8$) heuristically as a reasonable trade-off between navigation ease and information detail and have used $n = 100$ and $d = 4$ based on our own subjective assessment.

6.3.1 Human Annotation and Evaluation

For the evaluation, eight clusters (one from each sub-tree) were chosen to be labeled manually by four, independent human annotators. For this purpose, for each cluster, we provided the list of its keywords, K, and the candidate labels, L, to the annotators, and asked them to select the best $l = 10$ terms from L to describe the cluster. Initially, we had considered asking the annotators to also select representative labels from K, but a preliminary analysis showed that they were unsuitable to describe the cluster as a whole (Table 6.1 shows an example). Although the annotators were asked to provide their selection as a ranked list, we did not make use of their ranking order in the evaluation.

[4] `https://ciir.cs.umass.edu/downloads/WebAP/`
[5] `https://github.com/aneesha/RAKE`

As discussed in sections 2.2 and 6.2.3, the existing cluster labeling approaches either use a subset of cluster members [Manning *et al.*, 2008] or title of Wikipedia articles [Lau *et al.*, 2011; Bhatia *et al.*, 2016] to label clusters. As aforementioned, our preliminary analysis showed that the label set selected from the cluster members fails to provide high-level description. Moreover, the continuing work presented in [Lau *et al.*, 2011] and [Bhatia *et al.*, 2016] provided an automatic labeling for topics, and not words. Accordingly, we were unable to compare our labeling outcomes with the outcome of existing labeling approaches.

To evaluate the prediction accuracy, for each cluster we have considered the union of the lists provided by the human annotators as the ground truth (since $|L|$ was typically in the order of $150 - 200$, the intersection of the lists was often empty or minimal). As performance figure, we have decided to report the well-known precision at k ($P@k$) for values of k between one and ten. We have not used the recall since the ground truth had size 40 in most cases while the prediction's size was kept to $l = 10$ in all cases, resulting in a highest possible recall of 0.25. Figure 6.2 compares the average $P@k$ for $k = 1, \ldots, 10$ for the four proposed approaches. The two approaches based on minimum distance to the cluster center (*CentKey* and *CentHyp*) have outperformed the other two approaches based on frequencies (*FreqKey* and *FreqHyp*) for all values of k. This shows that the word embedding space is in good correspondence with the human judgement. Moreover, approach *CentHyp* has outperformed all other approaches for all values of k, showing that the hypernyms' centrality in the cluster is the key property for their effective selection.

6.3.2 Visualization of Keywords and Hypernyms

Hypernyms are more general terms than the corresponding keywords, thus we expect them to be in larger mutual distance in the word embedding space. To explore their distribution, we have used two-dimensional multidimensional scaling (MDS) visualizations [Borg and Groenen, 2005] of selected clusters. For each cluster, the keywords set K, the hypernyms set L, and the cluster's center have all been aggregated as a single set before applying MDS. An examples is shown in Figure 6.3. As can be seen, the hypernyms (blue dots) nicely distribute as a circular crown, external and concentric to the keywords (black dots), showing that the hypernymy relation corresponds empirically to

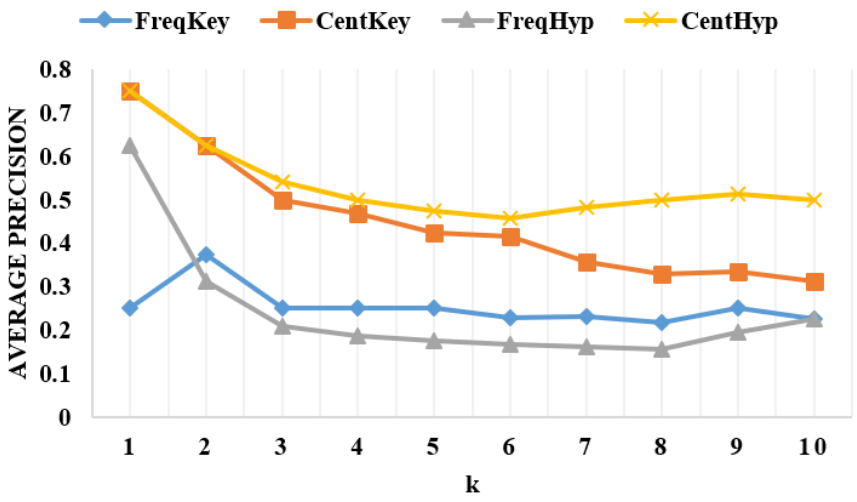

Figure 6.2: Precision at k ($P@k$) for $k = 1, \ldots, 10$ averaged over the eight chosen clusters for the compared approaches.

a radial expansion away from the cluster's center. This likely stems from the embedding space's requirement to simultaneously enforce meaningful distances between the different keywords, the keywords and the corresponding hypernyms, and between the hypernyms themselves. The hypernyms selected by the annotators (green and magenta dots) are among the closest to the cluster's center, and thus those selected by *CentHyp* (red and magenta dots) have the best correspondence (magenta dots alone) among the explored approaches.

6.3.3 A Detailed Example

As a detailed example, Table 6.1 lists all the keywords of a sample cluster and the hypernyms selected by the four human annotators and *CentHyp*. Some of the hypernyms selected by more than one annotator (e.g., "electronic communication", "web page" and "computer file") have also been successfully identified by *CentHyp*. On the other hand, *CentHyp* has selected at least two terms ("commercial enterprise" and "reference book') that are unrelated to the cluster. Qualitatively, we deem the automated annotation as noticeably inferior to the human annotations, yet usable wherever manual annotation is infeasible or impractical.

Keywords	website www, clearinghouse, nih website, bulletin, websites, hotline, kbr publications, pfm file, syst publication, gov web site, npl publication, beta site, lexis nexis document, private http, national register bulletin, daily routines, data custodian, information, serc newsletter, certified mail, informational guide, dot complaint database, coverage edit followup, local update, mass mailing, ahrq web site, homepage, journal messenger, dhhs site, htm centers, org website, web site address, service invocation, telephone directory, service records, page layout program, pdf private, newsletter, card reader, advisory workgroup, library boards, ccir papers, usg publication, webpage, bulletin boards, fbis online, teleconference info, ptd web site, insert libraries, headquarters files, volunteer website http, bibliographic records, vch publishers, ecommerce, traveler, tsbp newsletter, electronic bulletin boards, email addresses, journal url, ers web site, intranet, website http, newsletter nps files, , electronic mail notes subscribe, nna program, npci website, bulletin board, fais information, archiving, page attachment, nondriver id, mail etiquette, ip address, national directory, web page, pdq editorial boards, aml sites, dhs site, ptd website, directory, forums, digest, beta site management, api service, directories, full text online, ieee press, fips publication, org web site, clearinghouse database, monterey database, hotlines, dslip description info, danish desk files, sos web site, bna program, newsletters, app ropri, letterhead, inspections portal page, image file directory, website, reader, web site http, customized template page, mail addresses, web pages, internet questionnaire assistance, electronic bulletin board, health http, templates directory, beta site testers, informational, records service, coverage edit, quarterly newsletter, distributed, eos directly addresses, dataplot auxiliary directory, mail advertisement transmitted.
Annotator 1	**electronic_communication**, computer_network, **web_page**, **web_site**, mail, **text_file**, **computer_file**, protocol, software, electronic_equipment

Annotator 2	computer_network, telecommunication, computer, mail, **web_page**, information, news, press, code, software
Annotator 3	news, **informing**, medium, **web_page**, **computer_file**, written_record, document, press, article, essay
Annotator 4	communication, **electronic_communication**, **informing**, press, medium, document, electronic_equipment, computer_network, transmission, record
FreqKey	report, computer, trap, communication, critic, educator, fabric, holy_order, computer_network, clergyman
CentKey	web_site, computer_network, telegraphic_signal, messenger_boy, information, geographic_point, manner_of_speaking, street_sign, code, speech_act
FreqHyp	collection, object, position, system, electronic_equipment, attendant, report, computer_network, tract, computer
CentHyp	**electronic_communication**, information_measure, **text_file**, **web_page**, **informing**, print_media, **web_site**, **computer_file**, commercial_enterprise, reference_book

Table 6.1: An example cluster. The hypernyms selected by *CentHyp* and by at least one annotator are shown in boldface.

6.4 Conclusion

This chapter has explored various approaches for labeling keyword clusters based on the hypernyms from the WordNet lexical ontology. The proposed approaches map both the keywords and their hypernyms to a word embedding space and leverage the notion of centrality in the cluster. Experiments carried out using the WebAP dataset have shown that one of the approaches (*CentHyp*) has outperformed all the others in terms of precision at k for all values of k, and it has provided labels which are reasonably aligned with those of a pool of annotators.

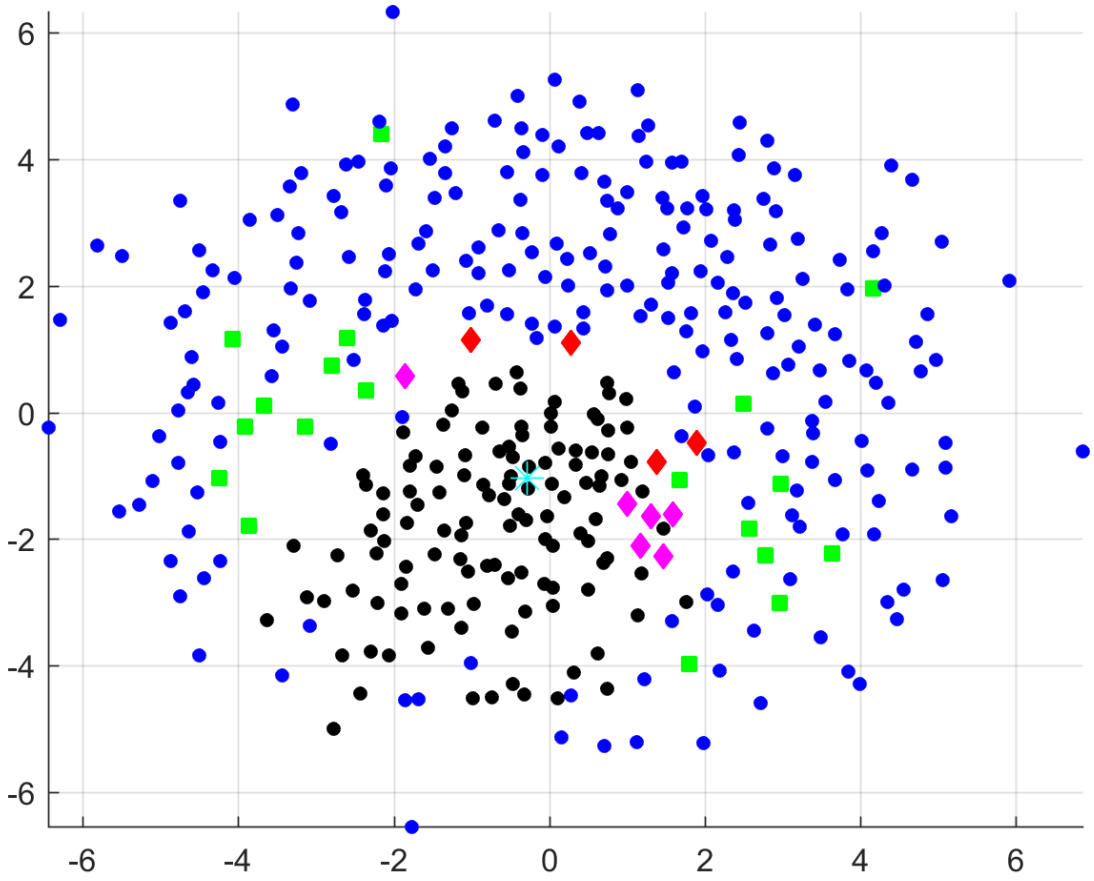

Figure 6.3: Two-dimensional visualization of an example cluster (this figure should be viewed in color). The black and blue dots are the cluster's keywords and the keywords' hypernyms, respectively. The green dots are the hypernyms selected by the human annotators, the red dots are the hypernyms selected by *CentHyp*, and their intersection is recolored in magenta. The cluster's center is the turquoise star.

The usefulness of the proposed labeling approach can be tested for tasks of search expansion. The labels can be used to expand queries or added to documents in forms of expressive tags. For both tasks, first, the query or the document is to be associated with relevant cluster/s. Then, for the former, the query can be expanded with the associated cluster's labels. For the latter, the associated clusters' labels can be augmented by the document as meta-tags.

Chapter 7

Conclusion

In this thesis, we have presented a series of work units on named entity recognition (NER) and cluster naming, two major components of text analysis and NLP. The ultimate goal of this research was to conceive and develop an improved structural prediction model for the NER task, while at the same time delivering useful software tools and digital resources for industrial application and community use.

Our first aim has been focused on the development of a supervised NER system for languages with low digital resources. To this end, we have provided and released ArmanPersoNERCorpus, the first manually-annotated Persian NE dataset, alongside four different Persian word embeddings. The released annotated dataset enables further development of Persian NER systems and can be used as test set for evaluation of systems trained on silver-standard corpora. Moreover, in addition to NER, the released word embeddings could find future use in other Persian NLP tasks including translation, question answering and summarization.

Our initial NER pipeline was composed of a sequential classifier learned over unsupervised word embeddings as feature vectors. We have investigated the combination of a number of word embeddings (including HPCA, GloVe, CBOW and skip-gram) and sequential prediction models (including CRF, structural SVM, Jordan-RNN and BiLSTM-CRF). Experiments conducted over the ArmanPersoNERCorpus have shown that the combination of skip-gram and BiLSTM-CRF has achieved the highest average F_1 score (77.45%) which provides a baseline for future comparisons on Persian NER.

Inspired by the achievements of the BiLSTM-CRF in the NLP literature, we have then

introduced the BiLSTM-SSVM, a novel training approach for the BiLSTM-CRF based on a hinge loss minimization. In this approach, the hinge loss is used as an upper bound for three evaluation losses, namely Hamming, CoNLL and a combination of the two. The required loss-augmented inference is challenging in the case of a non-decomposable loss such as CoNLL, and, for this reason, in this thesis, we have proposed an articulated dynamic programming algorithm that can perform the loss-augmented inference for the CoNLL loss and any other loss similarly based on entity-level error counting. Since the CoNLL loss is of the F_1 type, we have also argued that it may be a promising training objective for sentences with relatively sparse entities. For this reason, we have proposed a training objective that bounds the CoNLL loss for sentences with low entity density, and the Hamming loss otherwise ("mixed hinge"). Experiments conducted over four benchmark languages (English, German, Spanish and Dutch) have shown that training with the mixed hinge loss has achieved slightly higher accuracies than with the cross entropy for all languages. These results suggest that training with objectives closer to the evaluation measures can be an effective strategy, and that using different losses for sentences with different sufficient statistics should be explored further. As such, a future work is to extend our investigation to a variety of other tasks, losses and architectures.

As an extra project recommended by our sponsoring industry partner, we have also explored various approaches for labeling keyword clusters based on the hypernyms from the WordNet lexical ontology. The proposed approaches map both the keywords and their hypernyms to a word embedding space and leverage the notion of centrality in the cluster. Experiments carried out using the WebAP dataset have shown that one of the approaches (*CentHyp*) has outperformed all the others in terms of precision at k, and it has provided labels which are reasonably aligned with those of a pool of human annotators. As future work, we plan to test the usefulness of the predicted labels in tasks of search expansion and document collection organization.

References

[Adi and Keshet, 2016] Yossi Adi and Joseph Keshet. StructED: Risk minimization in structured prediction. *Journal of Machine Learning Research*, 17(64):1–5, 2016.

[Adi *et al.*, 2017] Yossi Adi, Joseph Keshet, Emily Cibelli, and Matthew Goldrick. Sequence segmentation using joint RNN and structured prediction models. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2422–2426, 2017.

[Al-Rfou *et al.*, 2015] Rami Al-Rfou, Vivek Kulkarni, Bryan Perozzi, and Steven Skiena. Polyglot-NER: Massive multilingual named entity recognition. In *Proceedings of 2015 SIAM International Conference on Data Mining*, pages 586–594, 2015.

[Althobaiti *et al.*, 2015] Maha Althobaiti, Udo Kruschwitz, and Massimo Poesio. Combining minimally-supervised methods for arabic named entity recognition. *Transactions of the Association for Computational Linguistics*, 3:243–255, 2015.

[Bhatia *et al.*, 2016] Shraey Bhatia, Jey Han Lau, and Timothy Baldwin. Automatic labelling of topics with neural embeddings. In *Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers*, pages 953–963, 2016.

[Biemann *et al.*, 2007] Chris Biemann, Gerhard Heyer, Uwe Quasthoff, and Matthias Richter. The leipzig corpora collection-monolingual corpora of standard size. In *Proceedings of Corpus Linguistic*, 2007.

[Bijankhan *et al.*, 2011] Mahmood Bijankhan, Javad Sheykhzadegan, Mohammad Bahrani, and Masood Ghayoomi. Lessons from building a persian written corpus: Peykare. *Language Resources and Evaluation*, 45(2):143–164, 2011.

[Bikel *et al.*, 1999] Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

[Bojanowski *et al.*, 2017] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[Borg and Groenen, 2005] I. Borg and P. J. F. Groenen. *Modern Multidimensional Scaling: Theory and Applications*. Springer Series in Statistics. Springer New York, 2005.

[Callison-Burch *et al.*, 2010] Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson ad Mark Przybocki, and Omar F Zaidan. Findings of the 2010 joint

workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, 2010.

[Chinchor, 1998] Nancy Chinchor. MUC-7 named entity task definition dry run version, version 3.5, 17 september 1997. In *Proceedings of the Seventh Message Understanding Conference*, 1998.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of the 8th Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, 2014.

[Clarke *et al.*, 2004] Charles Clarke, Nick Craswell, and Ian Soboroff. Overview of the trec 2004 terabyte track. In *Proceedings of the Thirteenth Text REtrieval Conference*, pages 103–111, 2004.

[Collobert *et al.*, 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*, 12:2493–2537, 2011.

[Dehdari, 2015] Jon Dehdari. A fast, simple, multilingual tokenizer. `https://github.com/jonsafari/tok-tok/`, 2015.

[Dernoncourt *et al.*, 2017] Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. NeuroNER: An easy-to-use program for named-entity recognition based on neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 97–102, 2017.

[Elman, 1990] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

[Espinosa-Anke *et al.*, 2016] Luis Espinosa-Anke, Horacio Saggion, Francesco Ronzano, and Roberto Navigli. Extasem! extending, taxonomizing and semantifying domain terminologies. In *Proceedings of the 30th Conference on Artificial Intelligence*, pages 2594–2600, 2016.

[Feely *et al.*, 2014] Weston Feely, Mehdi Manshadi, Robert E Frederking, and Lori S Levin. The CMU METAL Farsi NLP approach. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4052–4055, 2014.

[Feely, 2013] Weston Feely. Open-source dependency parser, part-of-speech-tagger, and text normalizer for farsi (persian). `https://github.com/wfeely/farsiNLPTools/`, 2013.

[Finkel *et al.*, 2005] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics*, pages 363–370, 2005.

[Fu *et al.*, 2014] Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209, 2014.

[Gimpel and Smith, 2012] Kevin Gimpel and Noah A. Smith. Structured ramp loss minimization for machine translation. In *Proceedings of the 2012 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231, 2012.

[Graff, 2011] David Graff. Spanish gigaword third edition. In *Linuistic Data Consortium*, 2011.

[Graves and Schmidhuber, 2005] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[Graves *et al.*, 2013] Alex Graves, Abdel-rahman Mohamed, and Geoffrey E. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649, 2013.

[Graves, 2013] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

[Grishman and Sundheim, 1996] Ralph Grishman and Beth Sundheim. Message understanding conference: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 466–471, 1996.

[Haffari *et al.*, 2015] Gholamreza Haffari, Ajay Nagesh, and Ganesh Ramakrishnan. Optimizing multivariate performance measures for learning relation extraction models. In *Proceedings of the 2015 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 892–900, 2015.

[Hartigan and Wong, 1979] J A. Hartigan and M. A. Wong. A K-Means clustering algorithm. *Journal of the Royal Statistical Society: Applied Statistics*, 28(1):100–108, 1979.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[Hoxha *et al.*, 2016] Julia Hoxha, Guoqian Jiang, and Chunhua Weng. Automated learning of domain taxonomies from text using background knowledge. *Journal of biomedical informatics*, 63:295–306, 2016.

[Huang *et al.*, 2014] Xiaolin Huang, Lei Shi, and Johan A.K. Suykens. Ramp loss linear programming support vector machine. *Journal of Machine Learning Research*, 15:2185–2211, 2014.

[Huang *et al.*, 2015] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

[Iacobacci *et al.*, 2015] I. Iacobacci, M. T. Pilehvar, and R. Navigli. Sensembed: Learning sense embeddings for word and relational similarity. In *Proceedings of the Association for Computational Linguistics*, pages 295–306, 2015.

[Joachims *et al.*, 2009] Thorsten Joachims, Thomas Hofmann, Yisong Yue, and Chun-Nam Yu. Predicting structured objects with support vector machines. *Communications of the ACM*, 52(11):97–104, 2009.

[Joachims, 2005] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 377–384, 2005.

[Joachims, 2008] Thorsten Joachims. SVM$^{\text{hmm}}$: Sequence tagging with structural support vector machines. `https://www.cs.cornell.edu/people/tj/svm_light/svm_hmm.html/`, 2008.

[Jordan, 1997] Michael I. Jordan. Serial order: A parallel distributed processing approach. *Advances in Psychology*, 121:471–495, 1997.

[Karmon and Keshet, 2015] Danny Karmon and Joseph Keshet. Risk minimization in structured prediction using orbit loss. *CoRR*, abs/1512.02033, 2015.

[Keikha *et al.*, 2014] M. Keikha, J. H. Park, W. B. Croft, and M. Sanderson. Retrieving Passages and Finding Answers. In *Proceedings of the 2014 Australasian Document Computing Symposium*, pages 81–84, 2014.

[Khormuji and Bazrafkan, 2014] Morteza Kolali Khormuji and Mehrnoosh Bazrafkan. Persian named entity recognition based with local filters. *International Journal of Computer Applications*, 100(4):1–6, 2014.

[Kozareva and Hovy, 2010] Zornitsa Kozareva and Eduard Hovy. A semi-supervised method to learn and construct taxonomies using the web. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1110–1118, 2010.

[Lafferty *et al.*, 2001] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

[Lample *et al.*, 2016] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris DyerRemi Lebret, and Ronan Collobert. Neural architectures for named entity recognition. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, 2016.

[Lau *et al.*, 2011] J. H. Lau, K. Grieser, D. Newman, and T. Baldwin. Automatic Labelling of Topic Models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1, pages 1536–1545, 2011.

[Lebret and Collobert, 2014] Remi Lebret and Ronan Collobert. Word embedding through Hellinger PCA. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 482–490, 2014.

[Lei *et al.*, 2014] Jianbo Lei, Buzhou Tang, Xueqin Lu, Kaihua Gao, Min Jiang, and Hua Xu. A comprehensive study of named entity recognition in chinese clinical text. *Journal of the American Medical Informatics Association*, 21:808–814, 2014.

[Liu *et al.*, 2018] Liyuan Liu, Jingbo Shang, Frank F. Xu, Xiang Ren, Huan Gui, Jian Peng, and Jiawei Han. Empower sequence labeling with task-aware neural language model. In *the 32nd AAAI Conference on Artificial Intelligence*, pages 5253–5260, 2018.

[Ma and Hovy, 2016] 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*, pages 1064–1074, 2016.

[Manning *et al.*, 2008] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[Manshadi, 2013] Mehdi Manshadi. Farsi verb tokenizer. `https://github.com/mehdi-manshadi/Farsi-Verb-Tokenizer/`, 2013.

[McAllester and Keshet, 2011] David McAllester and Joseph Keshet. Generalization bounds and consistency for latent structural probit and ramp loss. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pages 2205–2212, 2011.

[McAllester *et al.*, 2010] David McAllester, Tamir Hazan, and Joseph Keshet. Direct loss minimization for structured prediction. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems*, pages 1594–1602, 2010.

[Mesnil *et al.*, 2013] Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of the 14th Annual Conference of the International Speech Communication Association*, pages 3771–3775, 2013.

[Mesnil *et al.*, 2015] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, and Geoffrey Zweig. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539, 2015.

[Mikolov *et al.*, 2013a] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, volume 2, pages 3111–3119. 2013.

[Mikolov *et al.*, 2013b] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Workshop Proceedings of International Conference on Learning Representations*, 2013.

[Mikolov *et al.*, 2013c] Tomas Mikolov, Scott Wen tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, 2013.

[Miller, 1995] G. A. Miller. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[Mnih and Teh, 2012] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the International Conference on Machine Learning*, 2012.

[Morency *et al.*, 2010] Louis-Philippe Morency, C. Mario Christoudias, Ariadna Quattoni, Hugues Salamin, Giota Stratou, and Sybor Wang. Hidden-state conditional random field library. `http://multicomp.ict.usc.edu/?p=790`, 2010.

[Nadeau and Sekine, 2007] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, 2007.

[Nguyen and Guo, 2007] Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *Proceedings of the 24th International Conference on Machine Learning*, pages 681–688, 2007.

[Nguyen *et al.*, 2017] Kim Anh Nguyen, Maximilian Köper, Sabine Schulte im Walde, and Ngoc Thang Vu. Hierarchical embeddings for hypernymy detection and directionality. In *Proceedings of the 2017 Empirical Methods in Natural Language Processing*, pages 233–243, 2017.

[Nothman *et al.*, 2013] Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. Learning multilingual named entity recognition from Wikipedia. *Artificial Intelligence*, 194:151–175, 2013.

[Parker *et al.*, 2009] Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. English gigaword fourth edition. In *Linuistic Data Consortium*, 2009.

[Pennington *et al.*, 2014] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[Peters *et al.*, 2017] Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1756–1765, 2017.

[Peters *et al.*, 2018] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2227–2237, 2018.

[Poostchi *et al.*, 2016] Hanieh Poostchi, Ehsan Zare Borzeshi, Mohammad Abdous, and Massimo Piccardi. Personer: Persian named-entity recognition. In *The 26'th International Conference on Computational Linguistics*, pages 3381–3389, 2016.

[Poostchi *et al.*, 2018a] Hanieh Poostchi, , and Massimo Piccardi. Cluster labeling using word embeddings and WordNet's hypernymy. In *The 16'th Annual Workshop of The Australian Language Technology Association*, pages 66–70, 2018.

[Poostchi *et al.*, 2018b] Hanieh Poostchi, Ehsan Zare Borzeshi, and Massimo Piccardi. BiLSTM-CRF for Persian named-entity recognition ArmanPersoNERCorpus: the first entity-annotated Persian dataset. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, 2018.

[Qu *et al.*, 2014] Lizhen Qu, Yi Zhang, Rui Wang, Lili Jiang, Rainer Gemulla, and Gerhard Weikum. Senti-LSSVM: Sentiment-oriented multi-relation extraction with latent structural SVM. *Transactions of the Association for Computational Linguistics*, 2:155–168, 2014.

[Rabiner, 1989] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:257–286, 1989.

[Ramshaw and Marcus, 1995] Lance Ramshaw and Mitch Marcus. Text chunking using transformation-based learning. In *the third Workshop on Very Large Corpora*, 1995.

[Rasooli *et al.*, 2013] Mohammad Sadegh Rasooli, Manouchehr Kouhestani, and Amirsaeid Moloodi. Development of a persian syntactic dependency treebank. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 306–314, 2013.

[Rei, 2017] Marek Rei. Semi-supervised multitask learning for sequence labeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 2121–2130, 2017.

[Reimers and Gurevych, 2017] Nils Reimers and Iryna Gurevych. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 338–348, 2017.

[Rose *et al.*, 2010] S. Rose, D. Engel, N. Cramer, and W. Cowley. Automatic Keyword Extraction from Individual Documents. In *Text Mining: Applications and Theory*, chapter 1, pages 1–20. 2010.

[Rosenfeld *et al.*, 2014] Nir Rosenfeld, Ofer Meshi, Daniel Tarlow, and Amir Globerson. Learning structured models with the AUC loss and its generalizations. In *the 17th International Conference on Artificial Intelligence and Statistics*, pages 841–849, 2014.

[Sekine, 2007] Satoshi Sekine. The definition of sekine's extended named entities. `http://nlp.cs.nyu.edu/ene/version7_1_0Beng.html`, 2007. New York University.

[Seraji *et al.*, 2012] Mojgan Seraji, Beáta Megyesi, and Joakim Nivre. A basic language resource kit for persian. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 2245–2252, 2012.

[Seraji, 2013] Mojgan Seraji. Preper: A pre-processor for persian. In *Proceedings of Fifth International Conference on Iranian Linguistics*, 2013.

[Shamsfard, 2011] Mehrnoush Shamsfard. Challenges and open problems in persian text processing. *Proceedings of Language Technology Conference*, 11:65–69, 2011.

[Shi *et al.*, 2016] Yangyang Shi, Kaisheng Yao, Hu Chen, Dong Yu, Yi-Cheng Pan, and Mei-Yuh Hwang. Recurrent support vector machines for slot tagging in spoken language understanding. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 393–399, 2016.

[Shwartz *et al.*, 2016] Vered Shwartz, Yoav Goldberg, and Ido Dagan. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398, 2016.

[Song *et al.*, 2016] Yang Song, Alexander G. Schwing, Richard S. Zemel, and Raquel Urtasun. Training deep neural networks via direct loss minimization. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2169–2177, 2016.

[Sutton and McCallum, 2012] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2012.

[Suzuki *et al.*, 2006] Jun Suzuki, Erik McDermott, and Hideki Isozaki. Training conditional random fields with multivariate evaluation measures. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 217–224, 2006.

[Tang *et al.*, 2013] Buzhou Tang, Hongxin Cao, Yonghui Wu, Min Jiang, and Hua Xu. Recognizing clinical entities in hospital discharge summaries using structural support vector machines with word representation features. *BMC Medical Informatics and Decision Making*, 13(SUPPL1), 2013.

[Tjong Kim Sang and De Meulder, 2003] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning*, volume 4, pages 142–147, 2003.

[Tjong Kim Sang, 2002] Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning*, volume 20, pages 1–4, 2002.

[Tsochantaridis *et al.*, 2005] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[Vapnik, 1995] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[Voorhees and Chinchor, 2001] Ellen Voorhees and Nancy Chinchor. The message understanding conference scoring software user's manual. `http://www.itl.nist.gov/iaui/894.02/related_projects/muc/muc_sw/muc_sw_manual.html`, 2001.

[Wang *et al.*, 2014] J. Wang, C. Kang, Y. Chang, and J. Han. A hierarchical dirichlet model for taxonomy expansion for search engines. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 961–970, 2014.

[Yu *et al.*, 2015] Z. Yu, H. Wang, X. Lin, and M. Wang. Learning term embeddings for hypernymy identification. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1390–1397, 2015.

[Zhang *et al.*, 2016] Shi-Xiong Zhang, Rui Zhao, Chaojun Liu, Jinyu Li, and Yifan Gong. Recurrent support vector machines for speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5885–5889, 2016.

[Zhang *et al.*, 2017] Guopeng Zhang, Massimo Piccardi, and Ehsan Zare Borzeshi. Sequential labeling with structural SVM under non-decomposable losses. *IEEE Transactions on Neural Networks and Learning Systems*, 29:4177–4188, 2017.

[Zhou and Su, 2002] GuoDong Zhou and Jian Su. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of the 40'th Annual Meeting on Association for Computational Linguistics*, pages 473–480, 2002.