Towards a Universal Analyzer of Natural Languages

Waleed Ammar

CMU-LTI-16-010

June 2016

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Chris Dyer (co-chair), Carnegie Mellon University
Noah A. Smith (co-chair), University of Washington
Tom Mitchell, Carnegie Mellon University
Kuzman Ganchev, Google Research

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: multilingual NLP, dependency parsing, word alignment, part-of-speech tagging, low-resource languages, recurrent neural networks, word embeddings, linguistic typology, code switching, language identification, MALOPA, multiCluster, multiCCA, word embeddings evaluation, CRF autoencoders, language-universal.

For my family.

Acknowledgments

I had the pleasure and honor of interacting with great individuals who influenced my work throughout the five precious years of my Ph.D.

First and foremost, I would like to express my deep gratitude to Noah Smith and Chris Dyer for their enlightning guidance and tremendous support in every step of the road. Thank you for placing your confidence in me at the times I needed it the most. I will be indebted to you for the rest of my career.

I also would like to acknowledge the following groups of talented individuals:

- my collaborators: Chu-Cheng Lin, Lori Levin, Yulia Tsvetkov, D. Sculley, Kristina Toutanova, Shomir Wilson, Norman Sadeh, Miguel Ballesteros, George Mulcaire, Guillaume Lample, Pradeep Dasigi, Eduard Hovy, Anders Søgaard, Zeljko Agic, Hiroaki Hayashi, Austin Matthews and Manaal Faruqui for making work much more enjoyable;
- my (ex-)colleagues: Brendan O'Connor, Swabha Swayamdipta, Lingpeng Kong, Chu-Cheng Lin, Dani Yogatama, Ankur Parikh, Ahmed Hefny, Prasanna Kumar, Manaal Faruqui, Shay Cohen, Kevin Gimpel, Andre Martins, Avneesh Saluja, David Bamman, Sam Thomson, Austin Matthews, Dipanjan Das, Jesse Dodge, Matt Gardner, Mohammad Gowayyed, Adona Iosif, Jesse Dunietz, Subhodeep Moitra, Adhi Kuncoro, Yan Chuan Sim, Leah Nicolich-Henkin, Eva Schlinger and Victor Chahuneau for creating a diverse and enriching environment, and I forgive you for igniting my imposter syndrom more often than I would like to admit;
- my Google fellowship mentors: Ryan McDonald and George Foster for many stimulating discussions;
- my committee members Kuzman Ganchev and Tom Mitchell for their time and valuable feedback; and
- other members of the CMU family: Stacey Young, Laura Alford, Mallory Deptola, Roni Rosenfeld, Mohamed Darwish, Mahmoud Bishr, Mazen Soliman who made school feel more of a home than a workplace.

I am especially grateful for my family. Thank you Abdul-mottaleb and Mabroka Ammar for giving me everything and expecting nothing in return! Thank you Amany Hasan for sacrificing your best interests so that I can pursue my passion! Thank you Salma Ammar for creating so much joy in my life! Thank you Khaled Ammar for being a great role model to follow! Thank you Samia and Walaa Ammar for your tremendous love!

Abstract

Developing NLP tools for many languages involves unique challenges not typically encountered in English NLP work (e.g., limited annotations, unscalable architectures, code switching). Although each language is unique, different languages often exhibit similar characteristics (e.g., phonetic, morphological, lexical, syntactic) which can be exploited to synergistically train analyzers for multiple languages. In this thesis, we advocate for a novel language-universal approach to multilingual NLP in which one statistical model trained on multilingual, homogenuous annotations is used to process natural language input in multiple languages.

To empirically show the merits of the proposed approach, we develop **MALOPA** , a language-universal dependency parser which outperforms monolingually-trained parsers in several low-resource and high-resource scenarios. MALOPA is a greedy transition-based parser which uses multilingual word embeddings and other language-universal features as a homogeneous representation of the input across all languages. To address the syntactic differences between languages, MALOPA makes use of token-level language information as well as language-specific representations such as fine-grained part-of-speech tags. MALOPA uses a recurrent neural network architecture and multi-task learning to jointly predict POS tags and labeled dependency parses.

Focusing on homogeneous input representations, we propose novel methods for estimating multilingual word embeddings and for predicting word alignments. We develop two methods for estimating multilingual word embeddings from bilingual dictionaries and monolingual corpora. The first estimation method, **multiCluster**, learns embeddings of word clusters which may contain words from different languages, and learns distributional similarities by pooling the contexts of all words in the same cluster in multiple monolingual corpora. The second estimation method, **multiCCA**, learns a linear projection of monolingually trained embeddings in each language to one vector space, extending the work of Faruqui and Dyer (2014) to more than two languages. To show the scalability of our methods, we train multilingual embeddings in 59 languages. We also develop an extensible, easy-to-use web-based **evaluation portal** for evaluating arbitrary multilingual word embeddings on several intrinsic and extrinsic tasks. We develop the conditional random field autoencoder (**CRF autoencoder**) model for unsupervised learning of structured predictors, and use it to predict word alignments in parallel corpora. We use a feature-rich CRF model to predict the latent representation conditional on the observed input, then reconstruct the input conditional on the latent representation using a generative model which factorizes similarly to the CRF model. To reconstruct the observations, we experiment with a categorical distribution over word types (or word clusters), and a multivariate Gaussian distribution that generates pretrained word embeddings.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Multilingual NLP is the scientific and engineering discipline concerned with automatically analyzing written or spoken input in multiple human languages. The desired analysis depends on the application, but is often represented as a set of discrete variables (e.g., part-of-speech tags) with a presumed dependency structure (e.g., first-order Markov dependencies). A (partially) correct analysis can be used to enable natural computer interfaces such as Apple's Siri. Other applications include summarizing long articles in a few sentences (e.g., Salton et al., 1997), and discovering subtle trends in large amounts of user-generated text (e.g., O'Connor et al. 2010). The ability to process human languages has always been one of the primary goals of artificial intelligence since its conception by McCarthy et al. (1955).

1.1 Challenges in Multilingual NLP

Although some English NLP problems are far from solved, we can make a number of simplifying assumptions when developing monolingual NLP models for a high-resource language such as English. We can often assume that large annotated corpora are available. Even when they are not, it is reasonable to assume we can find qualified annotators either locally or via crowd-sourcing. It is easy to iteratively design and evaluate meaningful language-specific features (e.g., "[city] *is the capital of* [country]", "[POS] *ends with –ing*"). It is also often assumed the input language matches that of the training data.

When we develop models to analyze many languages, the first challenge we often find is the lack of annotations and linguistic resources. Language-specific feature engineering and error analysis becomes tedious at best, assuming we are lucky enough to collaborate with researchers or linguists who know all languages of interest. More often than not, however, it is infeasible to design meaningful language-specific features because the team has insufficient collective knowledge of some languages. Configuring, training, tuning, monitoring and occasionally updating the models for each language of interest is logistically difficult and requires more human and computational resources. Low-resource languages require a different pipeline than high-resource languages and are often ignored in an industrial setup.[1] The input can be in one of many languages, and often uses multiple languages in the same discourse. These challenges motivate the work in this dissertation.

Other multilingual challenges not addressed in this thesis include:

- identifying sentence boundaries in languages which do not use unique punctuation to end a sentence (e.g., Thai, Arabic),
- tokenization in languages which do not use spaces to separate words (e.g., Japanese),
- finding digitized texts of some languages (e.g., Yoruba, Hieroglyphic).

1.2 Thesis Statement

In this thesis, we show the feasibility of a language-universal approach to multilingual NLP in which one statistical model trained on multilingual, homogenuous annotations is used to process

[1] Although the performance on low-resource languages tend to be much worse than that of high-resource languages, even inaccurate predictions can be very useful. For example, a machine translation system trained on a small parallel corpus will produce inaccurate translations, but an inaccurate translation is often sufficient to learn what a foreign document is about.

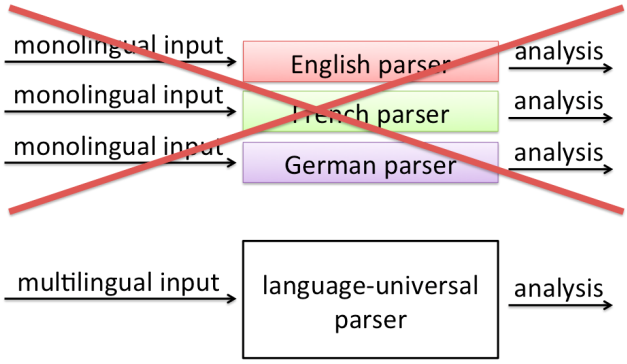


Figure 1.1: Instead of training one model per language, we develop one language-universal model to analyze text in multiple languages.

natural language input in multiple languages. This approach addresses several practical difficulties in multilingual NLP such as processing code-switched input and developing/maintaining a large number of models to cover languages of interest, especially low-resource ones. Although each language is unique, different languages often exhibit similar characteristics (e.g., phonetic, morphological, lexical, syntactic) which can be exploited to synergistically train universal language analyzers. The proposed language-universal models outperforms monolingually-trained models in several low-resource and high-resource scenarios.

Building on a rich literature in multilingual NLP, this dissertation enables the language-universal approach by developing statistical models for: i) a language-universal syntactic analyzer to exemplify the proposed approach, ii) estimating massively multilingual word embeddings to serve as a shared representation of natural language input in multiple languages, and iii) inducing word-alignments from unlabeled examples in a parallel corpus.

The models proposed in each of the three components are designed, implemented, and empirically contrasted to competitive baselines.

1.3 Summary of Contributions

In chapter 3, we describe the language-universal approach to training multilingual NLP models. Instead of training one model per language, we simultaneously train one language-universal model on annotations in multiple languages (see Fig. 1.1). We show that this approach outperforms comparable language-specific models on average, and also outperforms state-of-the-art models in low-resource scenarios where no or few annotations are available in the target language.

In chapter 4, we focus on the problem of estimating distributed, multilingual word representations. Previous work on this problem assumes the availability of sizable parallel corpora which connect all languages of interest in a fully connected graph. However, in practice, publicly available parallel corpora of high quality are only available for a relatively small number of languages. In order to scale up training of multilingual word embeddings to more languages, we propose two estimation methods which use bilingual dictionaries instead of parallel corpora.

In chapter 5, we describe a feature-rich model for structured prediction problems which learns

from unlabeled examples. We instantiate the model for POS induction, word alignments, and token-level language identification.

Chapter 2

Background

2.1 Natural Languages

While open-source NLP tools (e.g., TurboParser[1] and Stanford Parser[2]) are readily available in several languages, one can hardly find any tools for most living languages. The language-universal approach we describe in this thesis provides a practical solution for processing an arbitrary language, given a monolingual corpus and a bilingual dictionary (or a parallel corpus) to induce lexical features in that language. This section discusses some aspects of the diversity of natural languages to help the reader appreciate the full potential of our approach.

2.1.1 It is not sufficient to develop NLP tools for the few most popular languages

Ethnologue (Lewis et al., 2016), an extensive catalog of the world's languages, estimates that the world population of approximately 7.4 billion people natively speaks 6,879 languages as of 2016. Many of these languages are spoken by a large population. For instance, Fig. 2.1 shows that 306 languages have a population between one million and ten million native speakers.

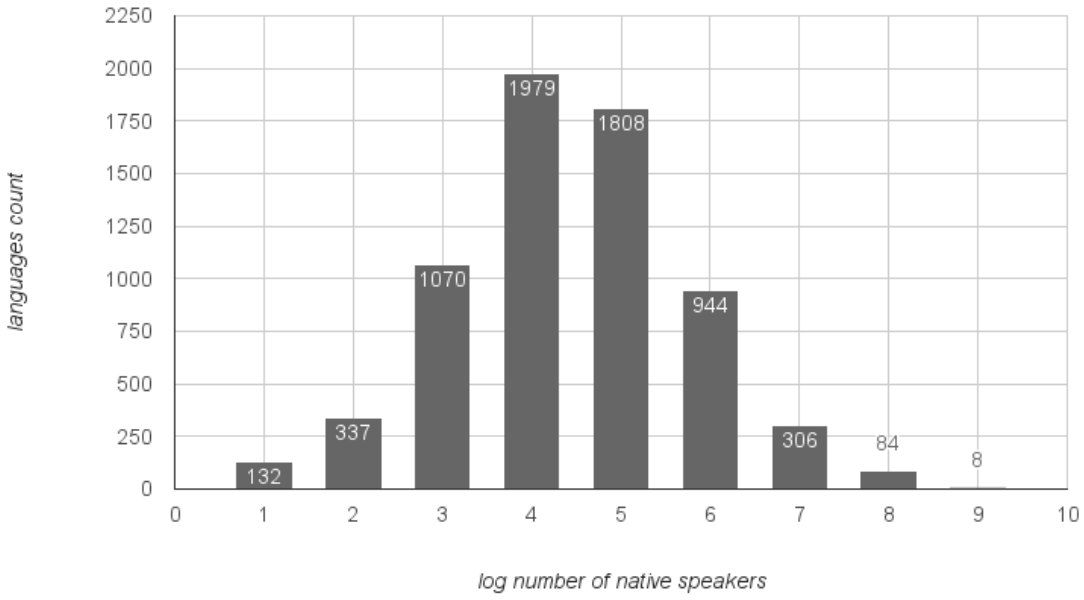


Figure 2.1: A histogram that shows the number of languages by population of native speakers in log scale. For instance, the figure shows that 306 languages have a population between one million and ten million native speakers. Numbers are based on Lewis et al. (2016).

[1] http://www.cs.cmu.edu/~ark/TurboParser/
[2] http://nlp.stanford.edu/software/lex-parser.shtml

Written languages used on the Internet also follow a similar pattern, although the language ranking is different (e.g., Chinese has the largest number of native speakers but English has the largest number of Internet users.) Fig. 2.2 gives the number of Internet users per language (for the top ten languages),[3] and shows that the long tail of languages (ranked 11 or more) account for 21.8% of all Internet users.

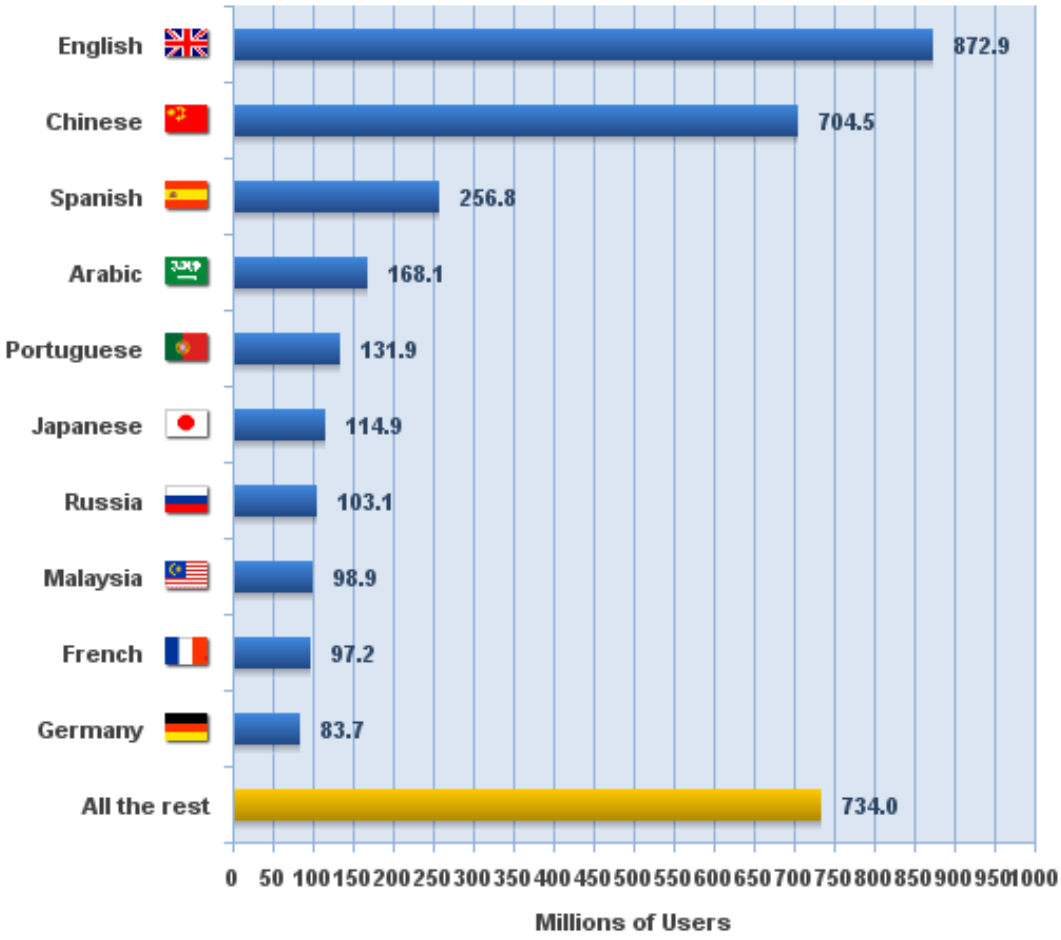


Figure 2.2: Number of Internet users (in millions) per language follows a power law distribution, with a long tail (not shown) which accounts for 21.8% of Internet users. The figure was retrieved on May 10, 2016 from `http://www.internetworldstats.com/stats7.htm`

Beyond languages with large population of native speakers, even endangered languages[4] and extinct languages[5] may be important to build NLP tools for. For example, The Endangered Languages Project[6] aims to document, preserve and teach endangered languages in order to reduce the loss of cultural knowledge when those languages fall out of use. NLP tools have

[3]Estimated by Miniwatts Marketing Group at `http://www.internetworldstats.com/stats7.htm`
[4]An endangered language is one at risk of reaching a native speaking population of zero as it falls out of use.
[5]Extinct languages have no living speaking population.
[6]http://www.endangeredlanguages.com/

also been used to help with historical discoveries by analyzing preserved text written in ancient languages (Bamman et al., 2013).

2.1.2 Languages often studied in NLP research

It is no surprise that some languages receive more attention than others in NLP research. Fig. 2.3, reproduced from Bender (2011), shows that 63% of papers in the ACL 2008 conference studied English, while only 0.7% (i.e., one paper) studied Danish, Swedish, Bulgarian, Slovene, Ukranian, Aramaic, Turkish or Wambaya.

Language	Studies		Genus	Studies		Family	Studies	
	N	%		N	%		N	%
English	81	63.28	Germanic	91	71.09	Indo-European	109	85.16
German	5	3.91						
Dutch	3	2.34						
Danish	1	0.78						
Swedish	1	0.78						
Czech	3	2.34	Slavic	8	6.25			
Russian	2	1.56						
Bulgarian	1	0.78						
Slovene	1	0.78						
Ukranian	1	0.78						
Portuguese	3	2.34	Romance	8	6.25			
Spanish	3	2.34						
French	2	1.56						
Hindi	2	1.56	Indic	2	1.56			
Arabic	4	3.13	Semitic	9	7.03	Afro-Asiatic	9	7.03
Hebrew	4	3.13						
Aramaic	1	0.78						
Chinese	5	3.91	Chinese	5	3.91	Sino-Tibetan	5	3.91
Japanese	3	2.34	Japanese	3	3.24	Japanese	3	3.24
Turkish	1	0.78	Turkic	1	0.78	Altaic	1	0.78
Wambaya	1	0.78	W. Barkly	1	0.78	Australian	1	0.78
Total	128	100.00		128	100.00		128	100.00

Figure 2.3: Reproduced from Bender (2011): Languages studied in ACL 2008 papers, by language genus and family.

The number of speakers of a language might explain the attention it receives, but only partially; e.g., Bengali, the native language of 2.5% of the world's population, is not studied by any papers in ACL 2008. Other factors which contribute to the attention a language receives in NLP research may include:

Availability of annotated datasets: Most NLP research is empirical, requiring the availability of annotated datasets. (Even unsupervised methods require an annotated dataset for evaluation.) It is customary to call a language with no or little annotations a "low-resource language", but the term is loosely defined. Some languages may have plenty of resources for one NLP problem, but no resources for another. Even for a particular NLP task, there is no clear threshold for the magnitude of annotated data below which a language is considered to be a low-resource language. Table 2.1 provides statistics about the size of datasets in

highly-multilingual resources: the Leipzig monolingual corpora,[7] OPUS parallel corpora,[8] and the universal dependency treebanks.[9]

Economic significance: The industrial arm of NLP research (e.g., Bell Labs, IBM, BBN, Microsoft, Google) has made important contributions to the field. Short of addressing all languages at the same time, more economically significant languages are often given a higher priority.

Research funding: Many NLP research studies are funded by national or regional agencies such as National Science Foundation (NSF), European Research Council (ERC), Defense Advanced Research Projects Agency (DARPA), Intelligence Advanced Research Projects Activity (IARPA) and the European Commission. Research goals of the funding agency often partially determines which languages will be studied in a funded project. For example, EuroMatrix was a three-year research project funded by the European Commission (2009–2012) and aimed to develop machine translation systems for all pairs of languages in the European Union.[10] Another example is TransTac, a five-year research project funded by DARPA which aimed to develop speech-to-speech translation systems, primarily for English–Iraqi and Iraqi–English. In both examples, the choice of languages studied was driven by strategic goals of the funding agencies.

2.1.3 Characterizing similarities and dissimilarities across languages

Linguistic typology (Comrie, 1989) is a field of linguistics which aims to organize languages into different types (i.e., to typologize languages). Typologists often use reference grammars[11] to contrast linguistic properties across different languages. An extensive list of typological properties can be found for 2,679 languages in the World Atlas of Languages Structures (WALS; Dryer and Haspelmath, 2013).[12] Studied properties include:

- syntactic patterns (e.g., order of subject, verb and object),
- morphological properties (e.g., reduplication, position of case affixes on nouns), and
- phonological properties (e.g., consonant-vowel ratio, uvular consonants).

It is also useful to consider genealogical classification of languages, emphasizing that languages which descended from a common ancestor tend to be linguistically similar. For example, Semitic languages (e.g., Hebrew, Arabic, Amharic) share a distinct morphological system that combines a triliteral root with a pattern of vowels and consonants to construct a word. Romance languages (e.g., Spanish, Italian, French) share morphological inflections that mark person (first, second, third), number (singular, plural), tense (e.g., imperfect, future), and mood (e.g., indicative, imperative).

[7] `http://corpora2.informatik.uni-leipzig.de/download.html`

[8] `http://opus.lingfil.uu.se/`

[9] `http://universaldependencies.org`

[10] Note the connection to economic significance.

[11] A reference grammar gives a technical description of the major linguistic features of a language with a few examples; e.g., Martin (2004) and Ryding (2005).

[12] WALS typological properties can be downloaded at `http://wals.info/`. Syntactic Structures of the World's Languages (SSWL) is another catalog of typological properties but it only has information about 385 languages by May 2016. SSWL typological properties can be downloaded at `http://sswl.railsplayground.net/`.

language	monolingual corpora (millions of tokens)	parallel corpora (millions of tokens)	universal dependency treebanks (thousands of tokens)
Ancient Greek		<1	244
Arabic	41	702	242
Basque	4	<1	121
Bulgarian	104	<1	156
Catalan	120	<1	530
Chinese	517	158	123
Croatian	42	407	87
Czech	487	866	1,503
Danish	154	691	100
Dutch	337	1,000	209
English	926	N/A	254
Estonian	38	262	234
Finnish	55	445	181
French	1,468	1,800	390
Galician	4	7	138
German	425	916	293
Gothic			56
Greek	73	1,000	59
Hebrew	47	356	115
Hindi	45	13	351
Hungarian	176	622	42
Indonesian	1,206	54	121
Irish	1	6	23
Italian	399	943	252
Japanese	58	17	267
Kazakh	1	<1	4
Latin	<1	<1	291
Latvian	1	134	20
Norwegian	84	44	311
Old Church Slavonic			57
Persian	194	79	151
Polish	96	600	83
Portuguese	53	1,000	226
Portuguese (Brazilian)	486	878	298
Romanian	125	859	145
Russian	1,800	619	1,032
Slovenian	54	378	140
Spanish	391	1,500	547
Swedish	107	464	96
Tamil	14	13	8
Turkish	13	520	56

Table 2.1: Current number of tokens in Leipzig monolingual corpora (in millions), word pairs in printed bilingual dictionaries (in thousands), tokens in the target side of en-xx OPUS parallel corpora (in millions), and the universal dependency treebanks (in thousands) for 41 languages. More recent statistics can be found at `http://opus.lingfil.uu.se/`, `http://www.bilingualdictionaries.com/`, `http://corpora2.informatik.uni-leipzig.de/download.html` and `http://universaldependencies.org/`, respectively.

2.2 Word Embeddings

Word embeddings (also known as vector space representations or distributed representations) provide an effective method for semi-supervised learning in monolingual NLP (Turian et al., 2010). This section gives a brief overview on monolingual word embeddings, before we discuss multilingual word embeddings in the following chapters.

2.2.1 Distributed word representations

There are several choices for representing lexical items in a computational model. The most basic representation, a sequence of characters (e.g., t–h–e–s–i–s), helps distinguish between different words. An isomorphic representation commonly referred to as "one-hot representation" assigns an arbitrary unique integer to each unique character sequence in a vocabulary of size V. The one-hot representation is often preferred to character sequences because modern computer architectures can manipulate integers more efficiently. This lexical representation is problematic for two reasons:

1. The number of unique n-gram lexical features is $\mathrm{O}\left(V^n\right)$. Since the vocabulary size V is often large,[13] the increased number of parameters makes the model more prone to overfitting. This can be problematic even for $n = 1$ (unigram features), especially when the training data is small.
2. We miss an opportunity to share statistical strength between similar words. For example, if "dissertation" is an out-of-vocabulary word (i.e., does not appears in the training set), the model cannot relate it to a similar word seen in training such as "thesis."

Word embeddings are an alternative representation which maps a word to a vector of real numbers; i.e. "embeds" the word in a vector space. This representation addresses the first problem, provided that the dimensionality of word embeddings is significantly lower than the vocabulary size, which is typical (dimensionality of word embeddings often used are in the range of 50–500). In order to address the second problem, two approaches are commonly used to estimate the embedding of a word:

- Estimate word embeddings as extra parameters in the model trained for the downstream task. Note that this approach reintroduces a large number of parameters in the model, and also misses the opportunity to learn from unlabeled examples.
- Use the distributional hypothesis of Harris (1954) to estimate word embeddings using a corpus of raw text, before training a model for the downstream task.

In §2.2.2, we describe a popular model for estimating word embetraining using an unlabeled corpus of raw text. We use this method as basis for multilingual embeddings in other chapters of the thesis.

2.2.2 Skipgram model for estimating monolingual word embeddings

The distributional hypothesis states that the semantics of a word can be determined by the words that occur in its context (Harris, 1954). The skipgram model (Mikolov et al., 2013a) is one of

[13]The vocabulary size depends on the language, genre and dataset size. An English monolingual corpus of 1 billion word tokens has a vocabulary size of 4,785,862 unique word types.

several methods which implement this hypothesis. The skipgram model generates a word u that occurs in the context (of window size K) of another word v as follows:

$$p(u \mid v) = \frac{\exp E_{\text{skipgram}}(v)^\top E_{\text{context}}(u)}{\sum_{u' \in \text{vocabulary}} \exp E_{\text{skipgram}}(v)^\top E_{\text{context}}(u')}$$

where $E_{\text{skipgram}}(u) \in \mathbb{R}^d$ is the vector word embedding of a word u with dimensionality d. $E_{\text{context}}(u)$ also embeds the word u, but the original implementation of the skipgram model in the word2vec package[14] only uses it as extra model parameters.

Note that this is a deficient model since the same word token appears (and hence generated) in more than one context (e.g., context of the word immediately before, and context of the word immediately after). The model is trained to maximize the log-likelihood as follows:

$$\sum_{i \in \text{indexes}} \sum_{k \in \{-K,\ldots,-1,1,\ldots,K\}} \log p(u_{i+k} \mid u_i)$$

where i indexes all word tokens in a monolingual corpus. To avoid the expensive summation in the partition function, the distribution $p(u \mid v)$ is approximated using noise contrastive estimation (Gutmann and Hyvärinen, 2012). We use the skipgram model in chapters 3, 4 and 5.

[14] `https://code.google.com/archive/p/word2vec/`

Chapter 3

Language-universal Dependency Parsing

3.1 Overview

In high-resource scenarios, the mainstream approach for multilingual NLP is to develop language-specific models. For each language of interest, the resources necessary for training the model are obtained (or created), and model parameters are optimized for each language separately. This approach is simple, effective and grants the flexibility of customizing the model or features to the needs of each language independently, but it is suboptimal for theoretical as well as practical reasons. Theoretically, the study of linguistic typology reveals that many languages share morphological, phonological, and syntactic phenomena (Bender, 2011). On the practical side, it is inconvenient to deploy or distribute NLP tools that are customized for many different languages because, for each language of interest, we need to configure, train, tune, monitor and occasionally update the model. Furthermore, code-switching or code-mixing (mixing more than one language in the same discourse), which is pervasive in some genres, in particular social media, presents a challenge for monolingually-trained NLP models (Barman et al., 2014).

Can we train one language-universal model instead of training a separate model for each language of interest, without sacrificing accuracy? We address this question in context of dependency parsing, a core problem in NLP (see Fig. 3.1). We discuss modeling tools for unifying languages to enable cross-lingual supervision, as well as tools for differentiating between the characteristics of different languages. Equipped with these modeling tools, we show that language-universal dependency parsers can outperform monolingually-trained parsers in high-resource scenarios.

The same approach can also be used in low-resource scenarios (with no labeled examples or with a small number of labeled examples in the target language), as previously explored by Cohen et al. (2011), McDonald et al. (2011) and Täckström et al. (2013). We address both experimental settings (target language with and without labeled examples) and show that our model compares favorably to previous work in both settings.

In principle, the proposed approach is applicable to many NLP problems, including morphological, syntactic, and semantic analysis. However, in order to exploit the full potential of this approach, we need homogenous annotations in several languages for the task of interest (see §3.2.1). For this reason, we focus on dependency parsing, for which homogenous annotations are available in many languages.

Most of the material in this chapter was previously published in Ammar et al. (2016a).

3.2 Approach

The availability of homogeneous syntactic annotations in many languages (Petrov et al., 2012; McDonald et al., 2013; Nivre et al., 2015b; Agić et al., 2015; Nivre et al., 2015a) presents the opportunity to develop a parser that is capable of parsing sentences in multiple languages of interest. Such parser can potentially replace an array of language-specific monolingually-trained parsers (for languages with a large treebank).

Our goal is to train a dependency parser for a set of target languages L^t, given universal dependency annotations in a set of source languages L^s. When all languages in L^t have a large treebank, the mainstream approach has been to train one monolingual parser per target language and route sentences of a given language to the corresponding parser at test time. In contrast, our approach is to train one parsing model with the union of treebanks in L^s, then use this single

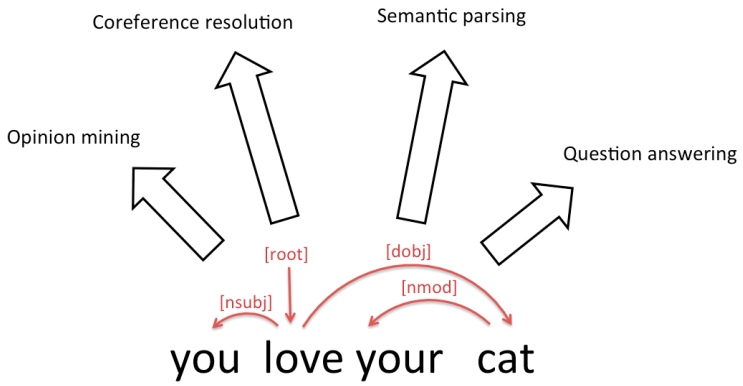


Figure 3.1: Dependency parsing is a core problem in NLP which is used to inform other tasks such as coreference resolution (e.g., Durrett and Klein, 2013), semantic parsing (e.g., Das et al., 2010) and question answering (e.g., Heilman and Smith, 2010).

trained model to parse text in any language in L^t, which we call "many languages, one parser" (MALOPA).

3.2.1 Homogenuous Annotations

Although multilingual dependency treebanks have been available for a decade via the 2006 and 2007 CoNLL shared tasks (Buchholz and Marsi, 2006; Nivre et al., 2007), the treebank of each language was annotated independently and with its own annotation conventions. McDonald et al. (2013) designed annotation guidelines which use similar dependency labels and conventions for several languages based on the Stanford dependencies. Two versions of this treebank were released: v1.0 (6 languages)[1] and v2.0 (11 languages).[2]. The dependency parsing community further developed these treebanks into the Universal Dependencies,[3] with a 6-month release schedule. So far, Universal Dependencies v1.0 (10 languages),[4] v1.1 (18 languages)[5] and v1.2 (34 languages)[6] have been released.

In MALOPA, we require that all source languages have a universal dependency treebank. We transform non-projective trees in the training treebanks to pseudo-projective trees using the "baseline" scheme in (Nivre and Nilsson, 2005).

In addition, we use the following data resources for each language in $L = L^t \cup L^s$:

- universal POS annotations for training a POS tagger (required),[7]
- a bilingual dictionary with another language in L for adding cross-lingual lexical information (optional),[8]

[1] https://github.com/ryanmcd/uni-dep-tb/blob/master/universal_treebanks_v1.0.tar.gz

[2] https://github.com/ryanmcd/uni-dep-tb/blob/master/universal_treebanks_v2.0.tar.gz

[3] http://universaldependencies.org/

[4] http://hdl.handle.net/11234/1-1464

[5] http://hdl.handle.net/11234/LRT-1478

[6] http://hdl.handle.net/11234/1-1548

[7] See §3.2.5 for details.

[8] Our best results make use of this resource. We require that all languages in L are (transitively) connected. The

$\textbf{Stack}_t$	$\textbf{Buffer}_t$	**Action**	**Dependency**	$\textbf{Stack}_{t+1}$	$\textbf{Buffer}_{t+1}$
u, v, S	B	REDUCE-RIGHT(r)	$u \xrightarrow{r} v$	u, S	B
u, v, S	B	REDUCE-LEFT(r)	$u \xleftarrow{r} v$	v, S	B
S	u, B	SHIFT	—	u, S	B

Table 3.1: Parser transitions indicating the action applied to the stack and buffer at time t and the resulting stack and buffer at time $t + 1$.

- language typology information (optional),[9]
- language-specific POS annotations (optional),[10] and
- a monolingual corpus (optional).[11]

3.2.2 Core Model

Recent advances (e.g., Graves et al. 2013, Sutskever et al. 2014) suggest that recurrent neural networks (RNNs) are capable of learning useful representations for modeling problems of sequential nature. Following Dyer et al. (2015), we use a RNN for transition-based dependency parsing. We describe the core model in this section, and modify it to enable language-universal parsing in the following sections. The core model can be understood as the sequential manipulation of three data structures:

- a buffer (from which we read the token sequence),
- a stack (which contains partially-built parse trees), and
- a list of actions previously taken by the parser.

The parser uses the arc-standard transition system (Nivre, 2004). At each timestep t, a transition action is applied that alters these data structures according to Table 3.1.

Along with the discrete transitions of the arc-standard system, the parser computes vector representations for the buffer, stack and list of actions at time step t denoted $\mathbf{b}_t$, $\mathbf{s}_t$, and $\mathbf{a}_t$, respectively (see Fig. 3.2). A stack-LSTM module is used to compute the vector representation for each data structure. Fig. 3.3 illustrates the content of each module for the first three steps in a toy example. The parser state[12] at time t is given by:

$$\mathbf{p}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t] + \mathbf{W}_{\text{bias}}\} \tag{3.1}$$

where the matrix $\mathbf{W}$ and the vector $\mathbf{W}_{\text{bias}}$ are learned parameters. The parser state $\mathbf{p}_t$ is then used to define a categorical distribution over possible next actions z:

$$p(z \mid \mathbf{p}_t) = \frac{\exp\left(\mathbf{g}_z^\top \mathbf{p}_t + q_z\right)}{\sum_{z'} \exp\left(\mathbf{g}_{z'}^\top \mathbf{p}_t + q_{z'}\right)} \tag{3.2}$$

bilingual dictionaries we used are based on unsupervised word alignments of parallel corpora, as described in Guo et al. (2016). See §3.2.3 for details.

[9] See §3.2.4 for details.

[10] Our best results make use of this resource. See §3.2.4 for details.

[11] This is only used for training word embeddings with 'multiCCA', 'multiCluster' and 'translation-invariance' in Table 3.6. We do not use this resource when we compare to previous work.

[12] Not to be confused with the state of the transition system (i.e., the content of the stack and the buffer).

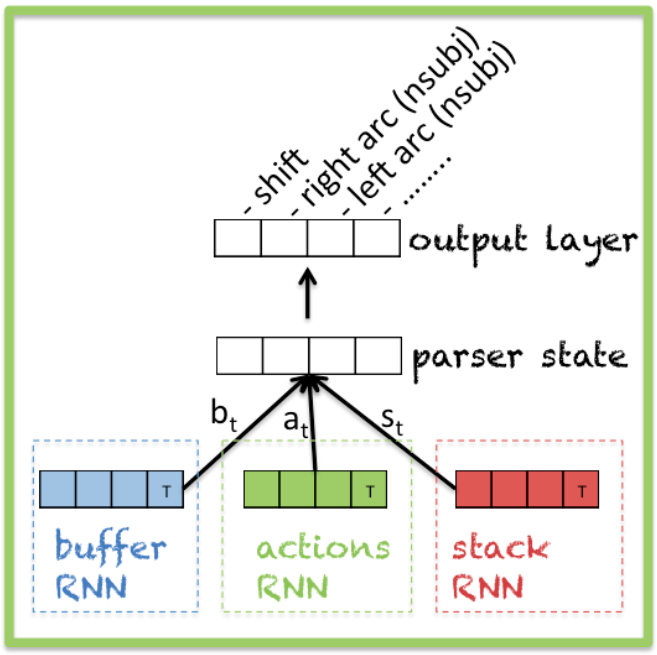


Figure 3.2: The parser computes vector representations for the buffer, stack and list of actions at time step t denoted $\mathbf{b}_t$, $\mathbf{s}_t$, and $\mathbf{a}_t$, respectively. The three vectors feed into a hidden layer denoted as 'parser state', followed by a softmax layer that represents possible outputs at time step t.

where $\mathbf{g}_z$ and q_z are parameters associated with action z. The total number of actions is twice the number of unique dependency labels in the treebank used for training plus one, but we only consider actions which meet the arc-standard preconditions in Table 3.1. The selected action is then used to update the buffer, stack and list of actions, and to compute $\mathbf{b}_{t+1}$, $\mathbf{s}_{t+1}$ and $\mathbf{a}_{t+1}$ accordingly.

The model is trained to maximize the log-likelihood of correct actions. At test time, the parser greedily chooses the most probable action in every time step until a complete parse tree is produced.

Token representations. The vector representations of input tokens feed into the stack-LSTM modules of the buffer and the stack. For monolingual parsing, we represent each token by concatenating the following vectors:

- a fixed, pretrained embedding of the word type,
- a learned embedding of the word,
- a learned embedding of the Brown cluster,
- a learned embedding of the fine-grained POS tag,
- a learned embedding of the coarse POS tag.

The next section describes the mechanisms we use to enable cross-lingual supervision.

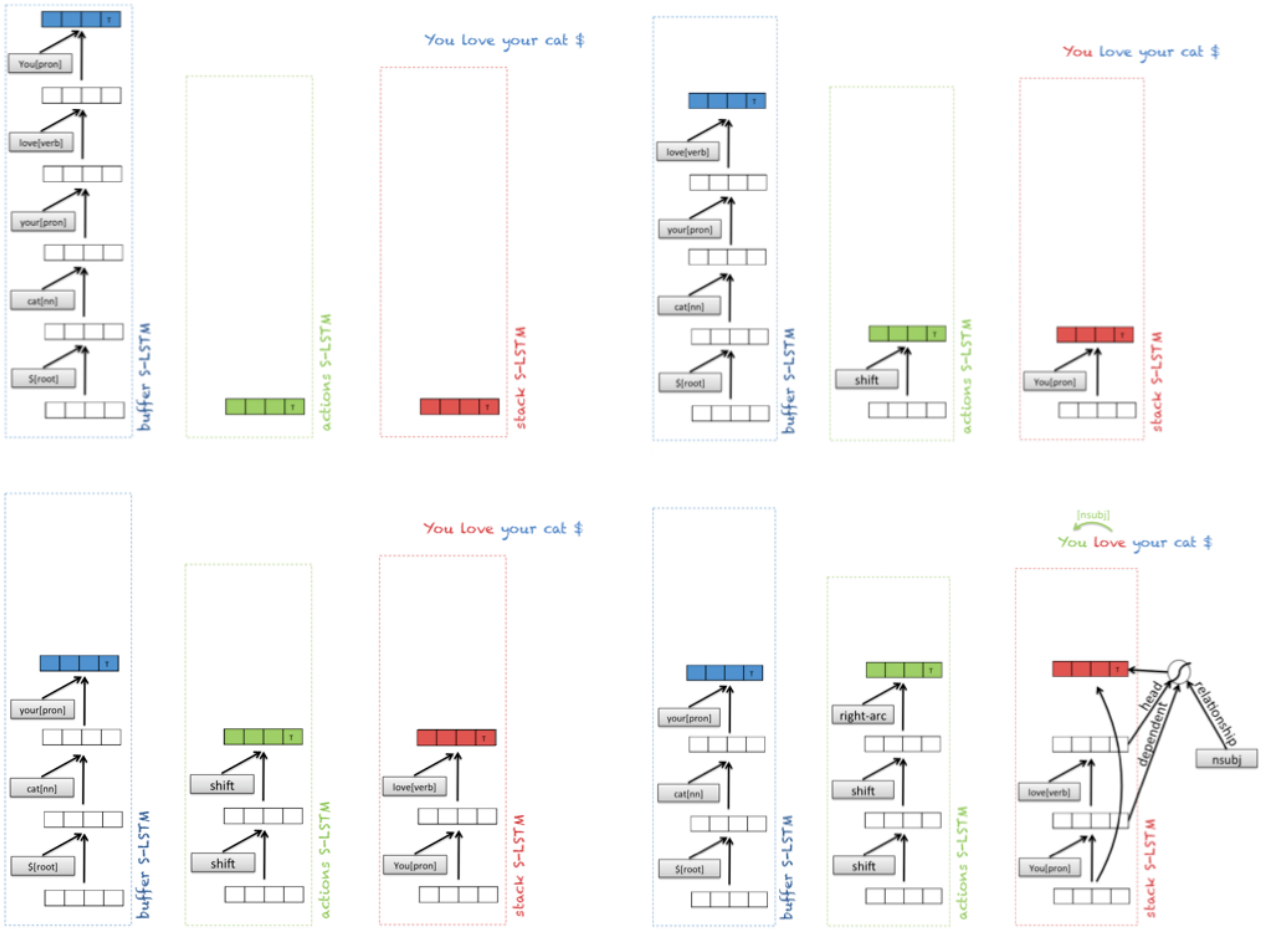


Figure 3.3: An illustration of the content of the buffer S-LSTM module, the actions S-LSTM module, and the stack S-LSTM module for the first three steps in parsing the sentence "You love your cat $".
Upper left: the buffer is initialized with all tokens in reverse order.
Upper right: simulating a 'shift' action, the head vector of the buffer S-LSTM backtracks, and two new hidden layers are computed for the actions S-LSTM and the stack S-LSTM.
Lower left: simulating another 'shift' action, the head vector of the buffer S-LSTM backtracks, and the two additional hidden layers are computed for the actions S-LSTM and the stack S-LSTM.
Lower right: simulating a 'right-arc(nsubj)', the head vector of the buffer S-LSTM backtracks, an additional hidden layer is computed for the actions S-LSTM. The stack S-LSTM first backtracks two steps, then a new hidden layer is computed as a function of the head, the dependent, and the relationship representations.

3.2.3 Language Unification

The key to unifying different languages in the model is to map language-specific representations of the input to language-universal representations. We apply this on two levels: part-of-speech tags and lexical items:

Coarse syntactic embeddings. We learn vector representations of multilingually-defined coarse POS tags (Petrov et al., 2012), instead of using language-specific tagsets. We train a simple delexicalized model where the token representation only consists of learned embeddings of coarse POS tags, which are shared across all languages to enable model transfer.

Lexical embeddings. Previous work has shown that sacrificing lexical features amounts to a substantial decrease in the performance of a dependency parser (Cohen et al., 2011; Täckström et al., 2012a; Tiedemann, 2015; Guo et al., 2015). Therefore, we extend the token representation in MALOPA by concatenating pretrained multilingual embeddings of word types. We also concatenate learned embeddings of multilingual word clusters. Before training the parser, we estimate Brown clusters of English words and project them via word alignments to words in other languages. This is similar to the 'projected clusters' method in Täckström et al. (2012a). To go from Brown clusters to embeddings, we ignore the hierarchy within Brown clusters and assign a unique parameter vector to each leaf.

3.2.4 Language Differentiation

Here, we describe how we tweak the behavior of MALOPA depending on the current input language.

Language embeddings. While many languages, especially ones that belong to the same family, exhibit *some* similar syntactic phenomena (e.g., all languages have subjects, verbs, and objects), substantial syntactic differences abound. Some of these differences are easy to characterize (e.g., subject-verb-object vs. verb-subject-object, prepositions vs. postpositions, adjective-noun vs. noun-adjective), while others are subtle (e.g., number and positions of negation morphemes). It is not at all clear how to translate descriptive facts about a language's syntax into features for a parser.

Consequently, training a language-universal parser on treebanks in multiple source languages requires caution. While exposing the parser to a diverse set of syntactic patterns across many languages has the potential to improve its performance in each, dependency annotations in one language will, in some ways, contradict those in typologically different languages.

For instance, consider a context where the next word on the buffer is a noun, and the top word on the stack is an adjective, followed by a noun. Treebanks of languages where postpositive adjectives are typical (e.g., French) will often teach the parser to predict REDUCE-LEFT, while those of languages where prepositive adjectives are more typical (e.g., English) will teach the parser to predict SHIFT.

Inspired by Naseem et al. (2012), we address this problem by informing the parser about the input language it is currently parsing. Let $\mathbf{l}$ be the input vector representation of a particular language. We consider three definitions for $\mathbf{l}$:

- a one-hot encoding of the language ID,

- a one-hot encoding of word-order properties,[13] and
- an encoding of all typological features in WALS.[14]

We use a hidden layer with $\tanh$ nonlinearity to compute the language embedding $\mathbf{l}'$ as:

$$\mathbf{l}' = \tanh(\mathbf{L} \times \mathbf{l} + \mathbf{L}_{\text{bias}})$$

where $\mathbf{L}$ and $\mathbf{L}_{\text{bias}}$ are additional model paramters. We modify the parsing architecture as follows:

- include $\mathbf{l}'$ in the token representation,
- include $\mathbf{l}'$ in the action vector representation, and
- let $\mathbf{p}_t = \max\{\mathbf{0}, \mathbf{W}[\mathbf{s}_t; \mathbf{b}_t; \mathbf{a}_t; \mathbf{l}'] + \mathbf{W}_{\text{bias}}\}$

Intuitively, the first two modifications allow the input language to influence the vector representation of the stack, the buffer and the list of actions. The third modification allows the input language to influence the parser state which in turn is used to predict the next action. In preliminary experiments, we found that adding the language embeddings at the token and action level is important. We also experimented with computing more complex functions of $(\mathbf{s}_t, \mathbf{b}_t, \mathbf{a}_t, \mathbf{l}')$ to define the parser state, but they did not help.

Fine-grained POS tag embeddings. Tiedemann (2015) shows that omitting fine-grained POS tags significantly hurts the performance of a dependency parser. However, those fine-grained POS tagsets are defined monolingually and are only available for a subset of the languages with universal dependency treebanks.

We extend the token representation to include a fine-grained POS embedding (in addition to the coarse POS embedding). During training, we stochastically dropout the fine-grained POS embedding with 50% probability (Srivastava et al., 2014) so that the parser can make use of fine-grained POS tags when available but stay reliable when the fine-grained POS tags are missing. Fig. 3.4 illustrates the parsing model with various components which enable cross-lingual supervision and language differentiation.

Block dropout. We introduce another modification which makes the parser more robust to noisy inputs and language-specific inputs which may or may not be provided at test time. The idea is to stochastically zero out the entire vector representation of a noisy input. While training the parser, we replace the vector representation $\mathbf{i}$ with another vector (of the same dimensionality) stochastically computed as: $\mathbf{i}' = (1-b)/\mu \times \mathbf{i}$, where b is a Bernoulli-distributed random variable with parameter μ which matches expected error rate on a development set.[15] For example, we use the block dropout to teach the parser to ignore the predicted POS tag embeddings all the time at first by initializing $\mu = 1.0$ (i.e., always dropout, setting $\mathbf{i}' = 0$), and dynamically update μ to match the error rate of the POS tagger on the development set. At test time, we always use the original vector, i.e., $\mathbf{i}' = \mathbf{e}$. Intuitively, this method extends the dropout method (Srivastava et al., 2014) to address structured noise in the input layer.

[13]The World Atlas of Language Structures (WALS; Dryer and Haspelmath, 2013) is an online portal documenting typological properties of 2,679 languages (as of July 2015). We use the same set of WALS features used by Zhang and Barzilay (2015), namely 82A (order of subject and verb), 83A (order of object and verb), 85A (order of adposition and noun phrase), 86A (order of genitive and noun), and 87A (order of adjective and noun).

[14]Since WALS features are not annotated for all languages, we use the average value of all languages in the same genus.

[15]Dividing by μ at training time alleviates the need to change the computation at test time.

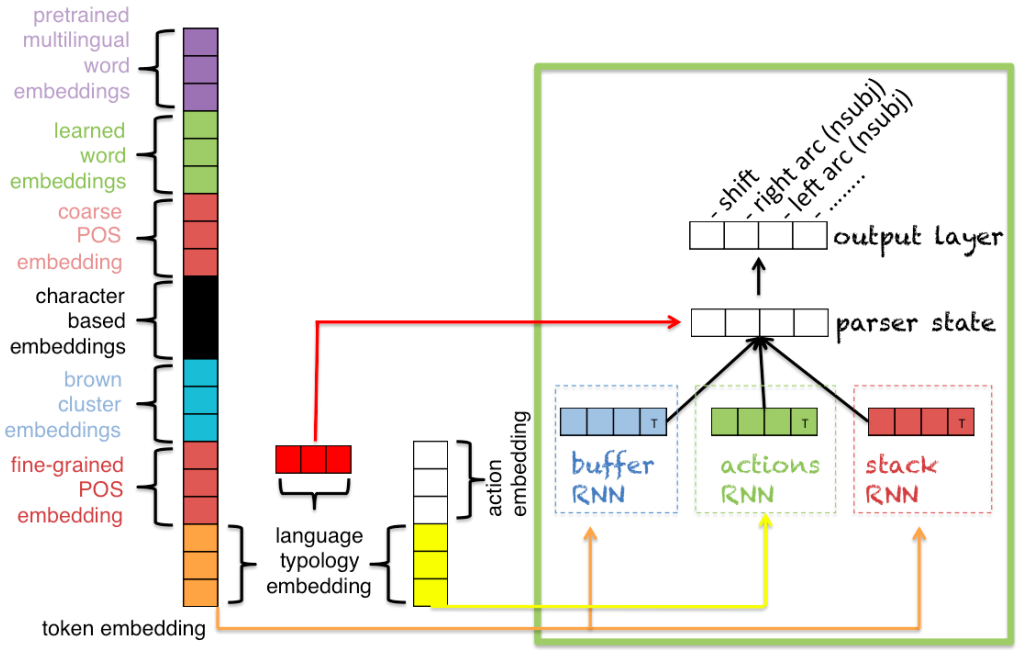


Figure 3.4: The vector representing language typology complements the token representation, the action representation and the input to the parser state.

3.2.5 Multi-task Learning for POS Tagging and Dependency Parsing

The model discussed thus far conditions on the POS tags of words in the input sentence. However, POS tags may not be available in real applications (e.g., parsing the web).

Let $x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_{2n}$ be the sequence of words, POS tags and parsing actions, respectively, for a sentence of length n. We define the joint distribution of a POS tag sequence and parsing actions given a sequence of words as follows:

$$p(y_1, \ldots, y_n, z_1, \ldots, z_{2n} \mid x_1, \ldots, x_n) = \prod_{i=1}^{n} p(y_i \mid x_1, \ldots, x_n) \times \prod_{j=1}^{2n} p(z_j \mid x_1, \ldots, x_n, y_1, \ldots, y_n, z_1, \ldots, z_{j-1})$$

where $p(z_j \mid \ldots)$ is defined in Eq. 3.2, and $p(y_i \mid x_1, \ldots, x_n)$ uses a bidirectional LSTM (Graves et al., 2013), similar to Huang et al. (2015).

The token representation that feeds into the bidirectional LSTM shares the same parameters of the token representation described earlier for the parser, but omits both POS embeddings. The output softmax layer defines a categorical distribution over possible POS tags at each position. This multi-task learning setup enables us to predict both POS tags and dependency trees with the same model.

3.3 Experiments

We evaluate our MALOPA parser in three data scenarios: when the target language has a large treebank (Table 3.3), a small treebank (Table 3.7) or no treebank (Table 3.8).

		German (de)	English (en)	Spanish (es)	French (fr)	Italian (it)	Portuguese (pt)	Swedish (sv)
UDT 2	train	14118 (264906)	39832 (950028)	14138 (375180)	14511 (351233)	6389 (149145)	9600 (239012)	4447 (66631)
	dev.	801 (12215)	1703 (40117)	1579 (40950)	1620 (38328)	399 (9541)	1211 (29873)	493 (9312)
	test	1001 (16339)	2416 (56684)	300 (8295)	300 (6950)	400 (9187)	1205 (29438)	1219 (20376)
UD 1.2	train	14118 (269626)	12543 (204586)	14187 (382436)	14552 (355811)	11699 (249307)	8800 (201845)	4303 (66645)
	dev.	799 (12512)	2002 (25148)	1552 (41975)	1596 (39869)	489 (11656)	271 (4833)	504 (9797)
	test	977 (16537)	2077 (25096)	274 (8128)	298 (7210)	489 (11719)	288 (5867)	1219 (20377)
	tags	-	50	-	-	36	866	134

Table 3.2: Number of sentences (tokens) in each treebank split in Universal Dependency Treebanks version 2.0 (UDT) and Universal Dependencies version 1.2 (UD) for the languages we experiment with. The last row gives the number of unique language-specific fine-grained POS tags used in a treebank.

Data. For experiments where the target language has a large treebank, we use the standard data splits for German (de), English (en), Spanish (es), French (fr), Italian (it), Portuguese (pt) and Swedish (sv) in the latest release (version 1.2) of Universal Dependencies (Nivre et al., 2015a), and experiment with both gold and predicted POS tags. For experiments where the target language has no treebank, we use the standard splits for these languages in the older universal dependency treebanks v2.0 (McDonald et al., 2013) and use gold POS tags, following the baselines (Zhang and Barzilay, 2015; Guo et al., 2016). Table 3.2 gives the number of sentences and words annotated for each language in both versions. We use the same multilingual Brown clusters and multilingual embeddings of Guo et al. (2016), kindly provided by the authors.

Optimization. We use stochastic gradient updates with an initial learning rate of $\eta_0 = 0.1$ in epoch #0 and update the learning rate in following epochs as $\eta_t = \eta_0/(1 + 0.1t)$. We clip the l_2 norm of the gradient to avoid exploding gradients. Unlabeled attachment score (UAS) on the development set determines early stopping. Parameters are initialized with uniform samples in $\pm\sqrt{6/(r+c)}$ where r and c are the sizes of the previous and following layer in the nueral network (Glorot and Bengio, 2010). The standard deviations of the labeled attachment score (LAS) due to random initialization in indiviual target languages are 0.36 (de), 0.40 (en), 0.37 (es), 0.46 (fr), 0.47 (it), 0.41 (pt) and 0.24 (sv). The standard deviation of the average LAS scores across langauges is 0.17.

When training the parser on multiple languages in MALOPA, instead of updating the parameters with the gradient of individual sentences, we use mini-batch updates which include one sentence sampled uniformly (without replacement) from each language's treebank, until all sentences in the smallest treebank are used (which concludes an epoch). We repeat the same process in following epochs. We found this to help prevent one source language with a larger treebank (e.g., German) from dominating parameter updates, at the expense of other source languages with a smaller treebank (e.g., Swedish).

3.3.1 Target Languages with a Treebank

Here, we evaluate our MALOPA parser when the target language has a treebank.

Baseline. For each target language, the strong baseline we use is a monolingually-trained S-LSTM parser with a token representation which concatenates: pretrained word embeddings (50

dimensions),[16] learned word embeddings (50 dimensions), coarse (universal) POS tag embeddings (12 dimensions), fine-grained (language-specific) POS tag embeddings (12 dimensions), and embeddings of Brown clusters (12 dimensions), and uses a two-layer S-LSTM for each of the stack, the buffer and the list of actions. We independently train one baseline parser for each target language, and share no model parameters. This baseline, denoted 'monolingual', achieves UAS score 93.0 and LAS score 91.5 when trained on the English Penn Treebank, which is comparable to Dyer et al. (2015).

MALOPA. We train MALOPA on the concantenation of training sections of all seven languages. To balance the development set, we only concatenate the first 300 sentences of each language's development section.

The first MALOPA parser we evaluate only uses coarse POS embeddings as the token representation.[17] As shown in Table 3.3, this parser consistently performs much worse than the monolingual baselines, with a gap of 12.5 LAS points on average.

Adding lexical embeddings to the token representation as described in §3.2.3 substantially improves the performance of MALOPA, recovering 83% of the gap in average performance.

We experimented with three ways to include language information in the token representation, namely: 'language ID', 'word order' and 'full typology' (see §3.2.4 for details), and found all three to improve the performance of MALOPA giving LAS scores 83.5, 83.2 and 82.5, respectively. It is interesting to see that the model is capable of learning more useful language embeddings when typological properties are not specified. Using 'language ID', we have now recovered another 12% of the original gap.

Finally, the best configuration of MALOPA adds fine-grained POS embeddings to the token representation.[18] Surprisingly, adding fine-grained POS embeddings improves the performance even for some languages where fine-grained POS tags are not available (e.g., Spanish), suggesting that the model is capable of predicting fine-grained POS tags for those languages via cross-lingual supervision. This parser outperforms the monolingual baseline in five out of seven target languages, and wins on average by 0.3 LAS points. We emphasize that this model is only trained once on all languages, and the *same model* is used to parse the test set of each language, which simplifies the distribution or deployment of multilingual parsing software.

We note that the fine-grained POS tags we used for Swedish and Portuguese encode morphological properties. Adding fine-grained POS tag embeddings for these two languages improved the results by 1.9 and 2.0 LAS points, respectively. This observation suggests that using morphological features as part of token representation is likely to result in further improvements. This is especially relevant since recent versions of the universal dependency treebanks include a universal annotations of morphological properties.

We note that, for some (model, test language) combinations, the improvements are small compared to the variance due to random initialization of model parameters. For example, the cell

[16]These embeddings are treated as fixed inputs to the parser, and are not optimized towards the parsing objective. We use the same embeddings used in Guo et al. (2016).

[17]We use the same number of dimensions for the coarse POS embeddings as in the monolingual baselines. The same applies to all other types of embeddings used in MALOPA.

[18]Fine-grained POS tags were only available for English, Italian, Portuguese and Swedish. Other languages reuse the coarse POS tags as fine-grained tags instead of padding the extra dimensions in the token representation with zeros.

UAS	target language							average
	de	en	es	fr	it	pt	sv	
monolingual	84.5	**88.7**	87.5	85.6	91.1	89.1	87.2	87.6
MALOPA	78.8	75.6	80.6	79.7	84.7	81.6	77.6	79.8
+lexical	83.0	85.6	87.3	85.3	90.6	86.5	86.4	86.3
+full typology	83.3	86.1	87.1	85.8	90.8	87.8	86.7	86.8
+word order	84.0	87.2	87.3	86.0	91.0	87.9	87.2	87.2
+language ID	84.2	87.2	87.2	86.1	**91.5**	87.5	87.2	87.2
+fine-grained POS	**84.7**	88.6	**88.1**	**86.4**	91.1	**89.4**	**88.2**	**88.0**

LAS	target language							average
	de	en	es	fr	it	pt	sv	
monolingual	**79.3**	**85.9**	83.7	81.7	88.7	85.7	83.5	84.0
MALOPA	70.4	69.3	72.4	71.1	78.0	74.1	65.4	71.5
+lexical	76.7	82.0	82.7	81.2	87.6	82.1	81.2	81.9
+full typology	77.8	82.5	82.6	81.5	88.0	83.7	81.8	82.5
+word order	78.5	84.3	83.4	81.7	88.3	84.1	82.6	83.2
+language ID	78.6	84.2	83.4	**82.4**	**89.1**	84.2	82.6	83.5
+fine-grained POS	78.9	85.4	**84.3**	**82.4**	89.0	**86.2**	**84.5**	**84.3**

Table 3.3: Dependency parsing: unlabeled and labeled attachment scores (UAS, LAS) for monolingually-trained parsers and MALOPA. Each target language has a large treebank (see Table 3.2). In this table, we use the universal dependencies verson 1.2 which only includes annotations for $\sim$ 13K English sentences, which explains the relatively low scores in English. When we instead use the universal dependency treebanks version 2.0 which includes annotations for $\sim$ 40K English sentences (originally from the English Penn Treebank), we achieve UAS score 93.0 and LAS score 91.5.

(+ full typology, it) in Table 3.2 shows an improvement of 0.4 LAS points while the standard deviation due to random initialization in Italian is 0.47 LAS. However, the average results across multiple languages show steady, robust improvements each of which far exceeds the standard deviation of 0.17 LAS.

Qualitative analysis. To gain a better understanding of the model behavior, we analyze certain classes of dependency attachments/relations in German, which has notably flexible word order, in Table 3.4. We consider the recall of left attachments (where the head word precedes the dependent word in the sentence), right attachments, root attachments, short-attachments (with distance $= 1$), long-attachments (with distance > 6), as well as the following relation groups: nsubj* (nominal subjects: `nsubj`, `nsubjpass`), dobj (direct object: `dobj`), conj (conjunct: `conj`), *comp (clausal complements: `ccomp`, `xcomp`), case (clitics and adpositions: `case`), *mod (modifiers of a noun: `nmod`, `nummod`, `amod`, `appos`), neg (negation modifier: `neg`). For each group, we report recall of both the attachment and relation weighted by the number of instances in the gold annotation. A detailed description of each relation can be found at `http://universaldependencies.org/u/dep/index.html`

Recall %	left	right	root	short	long	nsubj*	dobj	conj	*comp	case	*mod
monolingual	**89.9**	95.2	86.4	92.9	81.1	**77.3**	**75.5**	66.0	45.6	**93.3**	**77.0**
MALOPA	85.4	93.3	80.2	91.2	73.3	57.3	62.7	64.2	34.0	90.7	69.6
+lexical	**89.9**	93.8	84.5	92.6	78.6	73.3	73.4	66.9	35.3	91.6	75.3
+language ID	89.1	94.7	86.6	93.2	81.4	74.7	73.0	**71.2**	**48.2**	92.8	76.3
+fine-grained POS	89.5	**95.7**	**87.8**	**93.6**	**82.0**	74.7	74.9	69.7	46.0	**93.3**	76.3

Table 3.4: Recall of some classes of dependency attachments/relations in German.

We found that each of the three improvements (lexical embeddings, language embeddings and fine-grained POS embeddings) tends to improve recall for most classes. Unfortunately, MALOPA underperforms (compared to the monolingual baseline) in some classes nominal subjects, direct objects and modifiers of a noun. Nevertheless, MALOPA outperforms the baseline in some important classes such as root, long attachments and conjunctions.

Predicting language IDs and POS tags. In Table 3.3, we assume that both language ID of the input language and the POS tags are given at test time. However, this assumption may not be realistic in practical applications. Here, we quantify the degradation in parsing accuracy when language ID and POS tags are only given at training time, but must be predicted at test time. We do not use fine-grained POS tags in this part.

In order to predict language ID, we use the `langid.py` library (Lui and Baldwin, 2012)[19] and classify individual sentences in the test sets to one of the seven languages of interest, using the default models included in the library. The macro average language ID prediction accuracy on the test set across sentences is 94.7%. In order to predict POS tags, we use the model described in §3.2.5 with both input and hidden LSTM dimensions of 60, and with block dropout. The macro average accuracy of the POS tagger is 93.3%. Table 3.5 summarizes the four configurations: {gold language ID, predicted language ID} $\times$ {gold POS tags, predicted POS tags}. The performance of the parser suffers mildly (-0.8 LAS points) when using predicted language IDs, but suffers significantly (-5.1 LAS points) when using predicted POS tags. Nevertheless, the observed degradation in parsing performance when using predicted POS tags is comparable to the degradations reported by Tiedemann (2015).

The predicted POS results in Table 3.5 use block dropout. Without using block dropout, we lose an extra 0.2 LAS points in both configurations using predicted POS tags, averaging over all languages.

Different multilingual embeddings. Several methods have been proposed for pretraining multilingual word embeddings. We compare three of them: multiCCA and multiCluster (Ammar et al., 2016b) and robust projection (Guo et al., 2015). All embeddings are trained on the same data and use the same number of dimensions (100). Table 3.6 illustrates that the three methods perform comparably well on this task.

Small target treebank. Duong et al. (2015) considered a setup where the target language has a small treebank of $\sim$ 3K tokens, and the source language (English) has a large treebank of $\sim$ 205K. The parser proposed in Duong et al. (2015) is a neural network parser based on Chen and

[19] `https://github.com/saffsd/langid.py`

target langauge	de	en	es	fr	it	pt	sv	average
language ID accuracy %	96.3	78.0	100.0	97.6	98.3	94.0	98.8	94.7

target language	de	en	es	fr	it	pt	sv	average
POS tagging accuracy %	89.8	92.7	94.5	94.0	95.2	93.4	94.0	93.3

UAS		target language							average
language ID	coarse POS	de	en	es	fr	it	pt	sv	
gold	gold	84.2	87.2	87.2	86.1	91.5	87.5	87.2	87.2
predicted	gold	84.0	84.0	87.2	85.8	91.3	87.4	87.2	86.7
gold	predicted	78.9	84.8	85.4	84.0	89.0	84.4	81.0	83.9
predicted	predicted	78.5	79.7	85.4	83.8	88.7	83.2	80.9	82.8

LAS		target language							average
language ID	coarse POS	de	en	es	fr	it	pt	sv	
gold	gold	78.6	84.2	83.4	82.4	89.1	84.2	82.6	83.5
predicted	gold	78.5	80.2	83.4	82.1	88.9	83.9	82.5	82.7
gold	predicted	71.2	79.9	80.5	78.5	85.0	78.4	75.5	78.4
predicted	predicted	70.8	74.1	80.5	78.2	84.7	77.1	75.5	77.2

Table 3.5: Effect of automatically predicting language ID and POS tags with MALOPA on parsing accuracy.

multilingual embeddings	ave. UAS	ave. LAS
multiCluster	87.7	84.1
multiCCA	87.8	84.4
robust projection	87.8	84.2

Table 3.6: Effect of multilingual embedding estimation method on the multilingual parsing with MALOPA. UAS and LAS scores are macro-averaged across seven target languages.

LAS	target language				
	de	es	fr	it	sv
Duong et al.	61.8	**70.5**	67.2	71.3	62.5
MALOPA	**63.4**	**70.5**	**69.1**	**74.1**	**63.4**

Table 3.7: Small (3K token) target treebank setting: language-universal dependency parser performance.

Manning (2014), which shares most of the parameters between English and the target language, and uses an L_2 regularizer to tie the lexical embeddings of translationally-equivalent words. While not the primary focus of this paper,[20] we compare our proposed method to that of Duong et al. (2015) on five target languages for which multilingual lexical features are available from Guo et al. (2016). For each target language, we train the parser on the English training data in the UD version 1.0 corpus (Nivre et al., 2015b) and a small treebank in the target language.[21] Following Duong et al. (2015), we do not use any development data in the target languages, and we subsample the English training data in each epoch to the same number of sentences in the target language. We use the same hyperparameters specified before. Table 3.7 show that our proposed method outperforms Duong et al. (2015) by 1.4 LAS points on average.

3.3.2 Target Languages without a Treebank

McDonald et al. (2011) established that, when no treebank annotations are available in the target language, training on multiple source languages outperforms training on one (i.e., multi-source model transfer outperforms single-source model transfer). In this section, we evaluate the performance of our parser in this setup. We use two strong baseline multi-source model transfer parsers with no supervision in the target language:

- Zhang and Barzilay (2015) is a graph-based arc-factored parsing model with a tensor-based scoring function. It takes typological properties of a language as input. We compare to the best reported configuration (i.e., the column titled "OURS" in Table 5 of Zhang and Barzilay, 2015).
- Guo et al. (2016) is a transition-based neural-network parsing model based on Chen and Manning (2014). It uses a multilingual embeddings and Brown clusters as lexical features. We compare to the best reported configuration (i.e., the column titled "MULTI-PROJ" in Table 1 of Guo et al., 2016).

Following Guo et al. (2016), for each target language, we train the parser on six other languages in the Google Universal Dependency Treebanks version 2.0[22] (de, en, es, fr, it, pt, sv, excluding whichever is the target language). Our parser uses the same word embeddings and word clusters used in Guo et al. (2016), and does not use any typology information.[23]

[20]The setup cost involved in recruiting linguists, developing and revising annotation guidelines to annotate a new language ought to be higher than the cost of annotating 3K tokens.

[21]We thank Long Duong for providing the subsampled training corpora in each target language.

[22]`https://github.com/ryanmcd/uni-dep-tb/`

[23]In preliminary experiments, we found language embeddings to hurt the performance of the parser for target languages without a treebank.

UAS	target language						average
	de	es	fr	it	pt	sv	
Zhang and Barzilay (2015)	62.5	78.0	78.9	79.3	78.6	75.0	75.4
Guo et al. (2016)	65.0	79.0	77.6	78.4	**81.8**	78.2	76.3
this work	**65.2**	**80.2**	**80.6**	**80.7**	81.2	**79.0**	**77.8**

LAS	target language						average
	de	es	fr	it	pt	sv	
Zhang and Barzilay (2015)	54.1	68.3	68.8	69.4	72.5	62.5	65.9
Guo et al. (2016)	55.9	73.0	71.0	71.2	**78.6**	**69.5**	69.3
MALOPA	**57.1**	**74.6**	**73.9**	**72.5**	77.0	68.1	**70.5**

Table 3.8: Dependency parsing: unlabeled and labeled attachment scores (UAS, LAS) for multi-source transfer parsers in the simulated low-resource scenario where $L^t \cap L^s = \emptyset$.

The results in Table 3.8 show that, on average, our parser outperforms both baselines by more than 1 point in LAS, and gives the best LAS results in four (out of six) languages.

3.3.3 Parsing Code-switched Input

Code-switching presents a challenge for monolingually-trained NLP models (Barman et al., 2014). We hypothesize that our language-universal approach is a good fit for code-switched text. However, it is hard to test this hypothesis due to the lack of universal dependency treebanks with naturally-occurring code-switching.

Instead, we simulate an evaluation treebank with code-switching by replacing words in the English development set of the Universal Dependencies v1.2 with Spanish words. To account for the fact that Spanish words do not arbitrarily appear in code-switching with English, we only allow a Spanish word to substitute an English word under two conditions: (1) the Spanish word must be a likely translation of the English word, and (2) together with the (possibly modified) previous word in the treebank, the introduced Spanish word forms a bigram which appears in naturally-occurring code-switched tweets (from the EMNLP 2014 shared task on code switching (Lin et al., 2014)). 2.5% of English words in the development set were replaced with Spanish words. We use "pure" to refer to the original English development set,and "code-switched" to refer to the same development set after replacing 2.5% of English words with Spanish translations.

In order to quantify the degree to which monolingually-trained parsers make bad predictions when the input text is code-switched, we contrast the UAS performance of our joint model for tagging and parsing, trained on English treebanks with coarse POS tags only, and tested on pure vs. code-switched treebanks. We then repeat the same experiment with MALOPA parser trained on seven languages, with language ID and coarse POS tags only. The results in Table 3.9 suggest that (simulated) code-switched input adversely affects the performance of monolingually-trained parsers, but hardly affects the performance of our MALOPA parser.

UAS	pure	code-switched
monolingual	**85.0**	82.6
MALOPA	84.7	**84.8**

Table 3.9: UAS results on the first 300 sentences in the English development of the Universal Dependencies v1.2, with and without simulated code-switching.

3.4 Open Questions

Our results open the door for more research in multilingual NLP. Some of the questions triggered by our results are:

- Multilingual dependency parsing can be viewed as a domain adaptation problem where each language represents a different domain. Can we use the MALOPA approach in traditional domain adaptation settings?
- Can we combine the language-universal approach with other methods for indirect supervision (e.g., annotation projection, CRF autoencoders and co-training) to further improve performance in low-resource scenarios without hurting performance on high-resource scenarios?
- Can we obtain better results by sharing some of the model parameters for all members of the same language family?
- Can we apply the language-universal approach to more distant languages such as Arabic and Japanese?
- Can we apply the language-universal approach to more NLP problems such as named entity recognition and coreference resolution?

3.5 Related Work

Our work builds on the model transfer approach, which was pioneered by Zeman and Resnik (2008) who trained a parser on a source language treebank then applied it to parse sentences in a target language. Cohen et al. (2011) and McDonald et al. (2011) trained unlexicalized parsers on treebanks of multiple source languages and applied the parser to different languages. Naseem et al. (2012), Täckström et al. (2013), and Zhang and Barzilay (2015) used language typology to improve model transfer. To add lexical information, Täckström et al. (2012a) used multilingual word clusters, while Xiao and Guo (2014), Guo et al. (2015), Søgaard et al. (2015) and Guo et al. (2016) used multilingual word embeddings. Duong et al. (2015) used a neural network based model, sharing most of the parameters between two languages, and used an L_2 regularizer to tie the lexical embeddings of translationally-equivalent words. We incorporate these ideas in our framework, while proposing a novel neural architecture for embedding language typology (see §3.2.4) and another for consuming noisy structured inputs (block dropout). We also show how to replace an array of monolingually trained parsers with one multilingually-trained parser without sacrificing accuracy, which is related to Vilares et al. (2015).

Neural network parsing models which preceded Dyer et al. (2015) include Henderson (2003), Titov and Henderson (2007), Henderson and Titov (2010) and Chen and Manning (2014). Re-

lated to lexical features in cross-lingual parsing is Durrett et al. (2012) who defined lexico-syntactic features based on bilingual lexicons. Other related work include Östling (2015), which may be used to induce more useful typological to inform multilingual parsing. Tsvetkov et al. (2016) concurrently used a similar approach to learn language-universal language models based on morphology.

Another popular approach for cross-lingual supervision is to project annotations from the source language to the target language via a parallel corpus (Yarowsky et al., 2001; Hwa et al., 2005) or via automatically-translated sentences (Schneider et al., 2013; Tiedemann et al., 2014). Ma and Xia (2014) used entropy regularization to learn from both parallel data (with projected annotations) and unlabeled data in the target language. Rasooli and Collins (2015) trained an array of target-language parsers on fully annotated trees, by iteratively decoding sentences in the target language with incomplete annotations. One research direction worth pursuing is to find synergies between the model transfer approach and annotation projection approach.

3.6 Summary

In this chapter, we describe a general approach for training language-universal NLP models, and apply this approach to dependency parsing. The main ingredients of this approach are homogenuous annotations in multiple languages (e.g., the universal depedency treebanks), a core model with large capacity for representing complex functions (e.g., recurrent neural networks), mapping language-specific representations into a language-universal space (e.g., multilingual word embeddings), and mechanisms for differentiating between the behavior of different languages (e.g., language embeddings and fine-grained POS tags).

We show for the first time how to train language-universal models that perform competitively in multiple languages in both high- and low-resource scenarios.[24] We also show for the first time, using a simulated evaluation set, that language-universal models is a viable solution for processing code-switched text.

We note the importance of lexical features that connect multiple languages via bilingual dictionaries or parallel corpora. When the multilingual lexical features are based on out-of-domain resources such as the Bible, we observe a significant drop in performance. Developing competitive language-universal models with languages with small bilingual dictionaries remains an open problem for future research.

In principle, the proposed approach can be applied to many NLP problems such as named entity recognition, coreference resolution, question answering and textual entailment. However, in practice, the lack of homogenuous annotations in multiple languages for these tasks makes it hard to evaluate, let alone train, language-universal models. One possible approach to alleviate this practical difficulty is to use existing language-specific annotations for multiple languages and dedicate a small number of the model parameters to learn a mapping from a latent language-universal output to the language-specific annotations in a multi-task learning framework, which has been proposed in a different setup by Fang and Cohn (2016). We also hope that future multilingual annotation projects will develop annotation guidelines to encourage consistency

[24]The results in some (modification, test language) combinations were small relative to the standard deviation, suggesting that some of these modifications may not be important for those languages. Nevertheless, the average results across multiple languages show steady, robust improvements each of which far exceeds the standard deviation.

across languages, following the example of universal dependency treebanks.

Chapter 4

Multilingual Word Embeddings

4.1 Overview

The previous chapter discussed how to develop language-universal models for analyzing text. In this chapter, we focus on multilingual word embeddings, one of the important mechanisms for enabling cross-lingual supervision.

Vector-space representations of words are widely used in statistical models of natural language. In addition to improvements on standard monolingual NLP tasks (Collobert and Weston, 2008), shared representation of words *across* languages offers intriguing possibilities (Klementiev et al., 2012). For example, in machine translation, translating a word never seen in parallel data may be overcome by seeking its vector-space neighbors, if the embeddings are learned from both plentiful monolingual corpora and more limited parallel data. A second opportunity comes from transfer learning, in which models trained in one language can be deployed in other languages. While previous work has used hand-engineered features that are cross-lingually stable as the basis for model transfer (Zeman and Resnik, 2008; McDonald et al., 2011; Tsvetkov et al., 2014), automatically learned embeddings offer the promise of better generalization at lower cost (Klementiev et al., 2012; Hermann and Blunsom, 2014; Guo et al., 2016). We therefore conjecture that developing estimation methods for "massively" multilingual word embeddings (i.e., embeddings for words in a large number of languages) will play an important role in the future of multilingual NLP.

Novel contributions in this chapter include two methods for estimating multilingual embeddings which only require monolingual data in each language and pairwise bilingual dictionaries, and scale to a large number of languages. We also introduce our evaluation web portal for uploading arbitrary multilingual embeddings and evaluating them automatically using a suite of intrinsic and extrinsic evaluation methods.

The material in this chapter was previously published in Ammar et al. (2016b).

4.2 Estimation Methods

Let $\mathcal{L}$ be a set of languages, and let $\mathcal{V}^m$ be the set of surface forms (word types) in $m \in \mathcal{L}$. Let $\mathcal{V} = \bigcup_{m \in \mathcal{L}} \mathcal{V}^m$. Our goal is to estimate a partial **embedding** function $E : \mathcal{L} \times \mathcal{V} \rightarrow \mathbb{R}^d$ (allowing a surface form that appears in two languages to have different vectors in each). We would like to estimate this function such that: (i) semantically similar words in the same language are nearby, (ii) translationally equivalent words in different languages are nearby, and (iii) the domain of the function covers as many words in $\mathcal{V}$ as possible.

We use distributional similarity in a monolingual corpus M^m to model semantic similarity between words in the same language.[1] For cross-lingual similarity, either a parallel corpus $P^{m,n}$ or a bilingual dictionary $D^{m,n} \subset \mathcal{V}^m \times \mathcal{V}^n$ can be used. In some cases, we extract the bilingual dictionary from a parallel corpus. To do this, we align the corpus using fast_align (Dyer et al., 2013) in both directions. The estimated parameters of the word translation distributions are used to select pairs: $D^{m,n} = \big\{(u, v) \mid u \in \mathcal{V}^m, v \in \mathcal{V}^n, p_{m|n}(u \mid v) \times p_{n|m}(v \mid u) > \tau\big\}$, where the threshold τ trades off dictionary recall and precision.[2]

[1] Monolingual corpora are often an order of magnitude larger than parallel corpora. Therefore, multilingual word embedding models trained on monolingual corpora tend to have higher coverage.

[2] We fixed $\tau = 0.1$ for all language pairs based on manual inspection of the resulting dictionaries.

With three notable exceptions (see §4.2.3, §4.2.4, §4.5), previous work on multilingual embeddings only considered the bilingual case, $|\mathcal{L}| = 2$. In this section, we focus on estimating multilingual embeddings for $|\mathcal{L}| > 2$. We first describe two novel dictionary-based methods (multiCluster and multiCCA). Then, we review a variant of the multiSkip method (Guo et al., 2016) and the translation-invariance matrix factorization method (Gardner et al., 2015).[3]

4.2.1 MultiCluster embeddings

In this method, we decompose the problem into two simpler subproblems: $E = E_{\text{embed}} \circ E_{\text{cluster}}$, where $E_{\text{cluster}} : \mathcal{L} \times \mathcal{V} \rightarrow \mathcal{C}$ maps words to multilingual clusters $\mathcal{C}$, and $E_{\text{embed}} : \mathcal{C} \rightarrow \mathbb{R}^d$ assigns a vector to each cluster. We use a bilingual dictionary to find clusters of translationally equivalent words, then use distributional similarities of the clusters in monolingual corpora from all languages in $\mathcal{L}$ to estimate an embedding for each cluster. By forcing words from different languages in a cluster to share the same embedding, we create anchor points in the vector space to bridge languages. Fig. 4.1 illustrates this method with a schematic diagram.

More specifically, we define the clusters as the connected components in a graph where nodes are (language, surface form) pairs and edges correspond to translation entries in $D^{m,n}$. We assign arbitrary IDs to the clusters and replace each word token in each monolingual corpus with the corresponding cluster ID, and concatenate all modified corpora. The resulting corpus consists of multilingual cluster ID sequences. We can then apply any monolingual embedding estimator to obtain cluster embeddings; here, we use the skipgram model from Mikolov et al. (2013a).

Our implementation of the multiCluster method is available on GitHub.[4]

4.2.2 MultiCCA embeddings

In this method, we first use a monolingual estimator to independently embed words in each language of interest. We then pick a pivot language and and linearly project words from every other language to the vector space of the pivot language.[5]

In order to find the linear projection between two languages, we build on the work of Faruqui and Dyer (2014), who proposed a bilingual embedding estimation method based on canonical correlation analysis (CCA) and showed that the resulting embeddings for English words outperform monolingually-trained English embeddings on word similarity tasks. First, they use monolingual corpora to train monolingual embeddings for each language independently (E^m and E^n), capturing semantic similarity within each language separately. Then, using a bilingual dictionary $D^{m,n}$, they use CCA to estimate linear projections from the ranges of the monolingual embeddings E^m and E^n, yielding a bilingual embedding $E^{m,n}$. The linear projections are defined by $T_{m \rightarrow m,n}$ and $T_{n \rightarrow m,n} \in \mathbb{R}^{d \times d}$; they are selected to maximize the correlation between vector pairs $T_{m \rightarrow m,n} E^m(u)$ and $T_{n \rightarrow m,n} E^n(v)$, where $(u, v) \in D^{m,n}$. The bilingual embedding is then defined as $E_{\text{CCA}}(m, u) = T_{m \rightarrow m,n} E^m(u)$ (and likewise for $E_{\text{CCA}}(n, v)$).

We start by estimating, for each $m \in \mathcal{L} \setminus \{\text{en}\}$, the two projection matrices: $T_{m \rightarrow m,\text{en}}$ and $T_{\text{en} \rightarrow m,\text{en}}$; these are guaranteed to be non-singular, a useful property of CCA we exploit by invert-

[3] We developed the multiSkip method independently of Guo et al. (2016).

[4] `https://github.com/wammar/wammar-utils/blob/master/train-multilingual-embeddings.sh`

[5] We use English as the pivot language since English typically offers the largest corpora and wide availability of bilingual dictionaries.

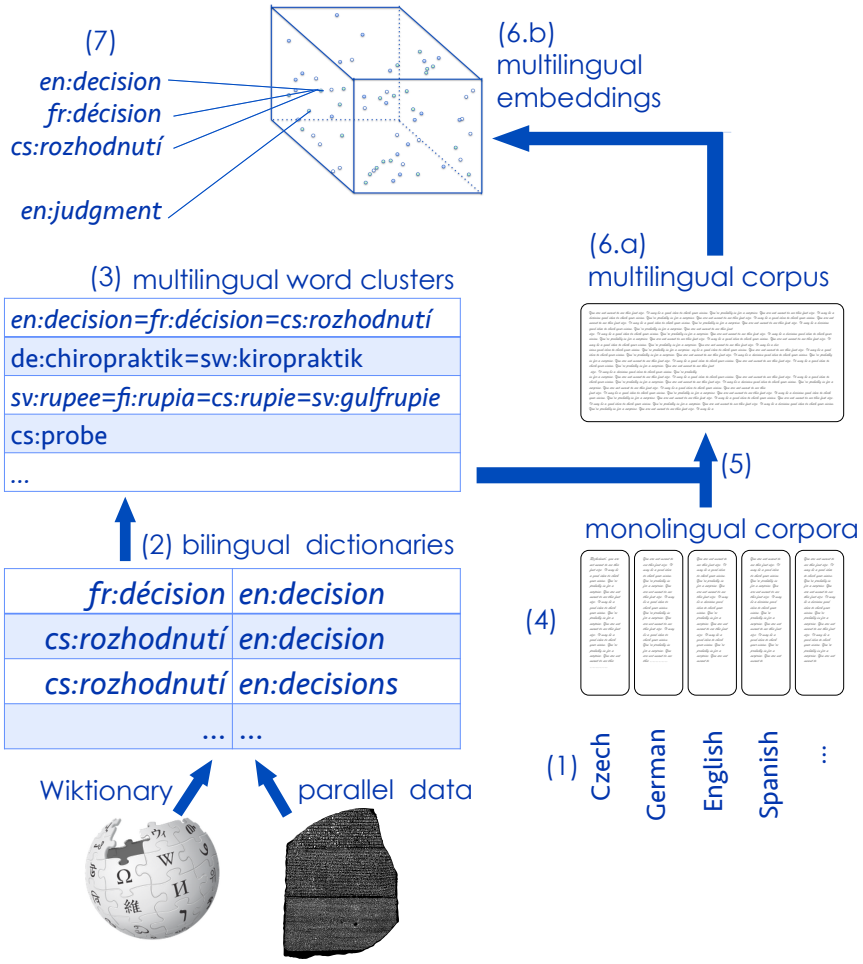


Figure 4.1: Steps for estimating multiCluster embeddings: 1) identify the set of languages of interest, 2) obtain bilingual dictionaries between pairs of languages, 3) group translationally equivalent words into clusters, 4) obtain a monolingual corpus for each language of interest, 5) replace words in monolingual corpora with cluster IDs, 6) obtain cluster embeddings, 7) use the same embedding for all words in the same cluster.

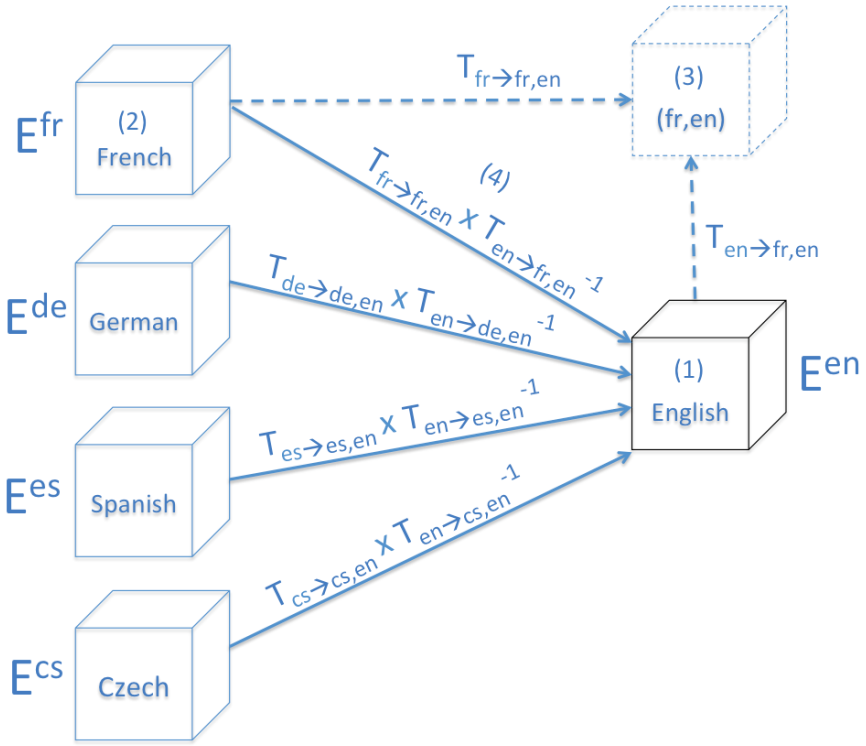


Figure 4.2: Steps for estimating multiCCA embeddings: 1) pretrain monolingual embeddings for the pivot language (English). For each other language m: 2) pretrain monolingual embeddings for m, 3) estimate linear projections $T_{m \to m,\text{en}}$ and $T_{\text{en} \to m,\text{en}}$, 4) project the embeddings of m into the embedding space of the pivot language.

ing the projection matrix. We then define the multilingual embedding as $E_{\text{CCA}}(\text{en}, u) = E^{\text{en}}(u)$ for $u \in \mathcal{V}^{\text{en}}$, and $E_{\text{CCA}}(m, v) = T^{-1}_{\text{en}\to m,\text{en}} T_{m\to m,\text{en}} E^m(v)$ for $v \in \mathcal{V}^m, m \in \mathcal{L} \setminus \{\text{en}\}$.

Our implementation of the multiCCA method is available on GitHub.[6]

4.2.3 MultiSkip embeddings

Luong et al. (2015a) proposed a method for estimating a bilingual embedding which only makes use of parallel data; it extends the skipgram model of Mikolov et al. (2013a). The skipgram model defines a distribution over words u that occur in a context window (of size K) of a word v (copied from Chapter 2 for convenience):

$$p(u \mid v) = \frac{\exp E_{\text{skipgram}}(m, v)^\top E_{\text{context}}(m, u)}{\sum_{u' \in \mathcal{V}^m} \exp E_{\text{skipgram}}(m, v)^\top E_{\text{context}}(m, u')}$$

In practice, this distribution can be estimated using a noise contrastive estimation approximation (Gutmann and Hyvärinen, 2012) while maximizing the log-likelihood:

$$\sum_{i \in \text{pos}(M^m)} \sum_{k \in \{-K,\ldots,-1,1,\ldots,K\}} \log p(u_{i+k} \mid u_i)$$

where $\text{pos}(M^m)$ are the indices of words in the monolingual corpus M^m.

To establish a bilingual embedding, with a parallel corpus $P^{m,n}$ of source language m and target language n, Luong et al. (2015a) estimate conditional models of words in both source and target positions. The source positions are selected as sentential contexts (similar to monolingual skipgram), and the bilingual contexts come from aligned words. The bilingual objective is to maximize:

$$\begin{aligned}
&\sum_{i \in m\text{-pos}(P_{m,n})} \sum_{k \in \{-K,\ldots,-1,1,\ldots,K\}} \log p(u_{i+k} \mid u_i) + \log p(v_{a(i)+k} \mid u_i) \\
&+ \sum_{j \in n\text{-pos}(P_{m,n})} \sum_{k \in \{-K,\ldots,-1,1,\ldots,K\}} \log p(v_{j+k} \mid v_j) + \log p(u_{a(j)+k} \mid v_j) \qquad (4.1)
\end{aligned}$$

where $m\text{-pos}(P_{m,n})$ and $n\text{-pos}(P_{m,n})$ are the indeces of the source and target tokens in the parallel corpus respectively, $a(i)$ and $a(j)$ are the positions of words that align to i and j in the other language. It is easy to see how this method can be extended for more than two languages by summing up the bilingual objective in Eq. 4.1 for all available parallel corpora.

Our implementation of the multiSkip method is available on GitHub.[7]

4.2.4 Translation-invariant matrix factorization

Gardner et al. (2015) proposed that multilingual embeddings should be translation invariant. Consider a matrix $X \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ which summarizes the pointwise mutual information statistics between pairs of words in monolingual corpora, and let $UV^\top$ be a low-rank decomposition of X where $U, V \in \mathbb{R}^{|\mathcal{V}|\times d}$. Now, consider another matrix $A \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ which summarizes bilingual

[6] https://github.com/mfaruqui/crosslingual-cca/blob/master/train-multilingual-embeddings.sh

[7] https://github.com/gmulcaire/multivec

alignment frequencies in a parallel corpus. Gardner et al. (2015) solves for a low-rank decomposition $UV^\top$ which both approximates X as well as its transformations $A^\top X$, XA and $A^\top XA$ by defining the following objective:

$$\min_{U,V} \|X - UV^\top\|^2 + \|XA - UV^\top\|^2 + \|A^\top X - UV^\top\|^2 + \|A^\top XA - UV^\top\|^2$$

The multilingual embeddings are then taken to be the rows of the matrix U.

4.3 Evaluation Portal

We discussed several methods for estimating multilingual embeddings, but how do we evaluate multilingual embeddings obtained using different methods? In order to facilitate research on multilingual word embeddings, we developed a web portal[8] to compare different estimation methods using a suite of evaluation tasks. The portal serves the following purposes:

- Download the monolingual and bilingual data we used to estimate multilingual embeddings in this thesis,
- Download standard development/test data sets for each of the evaluation metrics to help researchers working in this area report trustworthy and replicable results,[9]
- Upload arbitrary multilingual embeddings, scan which languages are covered by the embeddings, allow the user to pick among the compatible evaluation tasks, and receive evaluation scores for the selected tasks, and
- Register a new evaluation data set or a new evaluation metric via the github repository which mirrors the backend of the web portal.

The following subsections describe the evaluation methods available on the portal in some detail, and Table 4.1 lists the available languages for each method.

4.3.1 Word similarity

Word similarity datasets such as WS-353-SIM (Agirre et al., 2009) and MEN (Bruni et al., 2014) provide human judgments of semantic similarity. By ranking words by cosine similarity and by their empirical similarity judgments, a ranking correlation can be computed that assesses how well the estimated vectors capture human intuitions about semantic relatedness.

Some previous work on bilingual and multilingual embeddings has focused on monolingual word similarity to evaluate embeddings, e.g., Faruqui and Dyer (2014). This approach is limited because it cannot measure the degree to which embeddings from different languages are similar. Instead, we recommend reporting results on both monolingual datasets, e.g., Luong et al. (2013) and cross-lingual word similarity datasets, e.g., Camacho-Collados et al. (2015).

4.3.2 Word translation

This task directly assesses the degree to which translationally equivalent words in different languages are nearby in the embedding space. The evaluation data consists of word pairs which

[8] `http://128.2.220.95/multilingual`

[9] Except for the original RCV documents, which are restricted by the Reuters license and cannot be republished. All other data is available for download.

metric	language ISO 639-1 codes
document classification	da, de, en, it, fr, sv
dependency parsing	bg, cs, da, de, el, en, es, fi, fr, hu, it, sv
(multi)QVEC-CCA/(multi)QVEC	da, en, it
word similarity	de, en, es, fa, fr, it, pt
word translation	bg, cs, da, de, el, en, es, fi, fr, hu, it, sv, zh, af, ca, iw, cy, ar, ga, zu, et, gl, id, ru, nl, pt, la, tr, ne, lv, lt, tg, ro, is, pl, yi, be, hy, hr, jw, ka, ht, fa, mi, bs, ja, mg, tl, ms, uz, kk, sr, mn, ko, mk, so, uk, sl, sw

Table 4.1: Evaluation metrics on the corpus and languages for which evaluation data are available.

are known to be translationally equivalent. The score for one word pair $(l_1, w_1), (l_2, w_2)$ both of which are covered by an embedding E is 1 if

$$\text{cosine}\big(E(l_1, w_1), E(l_2, w_2)\big) \geq \text{cosine}\big(E(l_1, w_1), E(l_2, w_2')\big), \forall w_2' \in G^{l_2}$$

where G^{l_2} is the set of words of language l_2 in the evaluation dataset, and cosine is the cosine similarity function. Otherwise, the score for this word pair is 0. The overall score is the average score for all word pairs covered by the embedding function. This is a variant of the method used by Mikolov et al. (2013b) to evaluate bilingual embeddings.

4.3.3 Correlation-based evaluation tasks

QVEC: The main idea behind QVEC is to quantify the linguistic content of word embeddings by maximizing the correlation with a manually-annotated linguistic resource. Let the number of common words in the vocabulary of the word embeddings and the linguistic resource be N. To quantify the semantic content of embeddings, a semantic linguistic matrix $\mathbf{S} \in \mathbb{R}^{P \times N}$ is constructed from a semantic database, with a column vector for each word. Each word vector is a distribution of the word over P linguistic properties, based on annotations of the word in the database. Let $\mathbf{X} \in \mathbb{R}^{D \times N}$ be embedding matrix with every row as a dimension vector $\mathbf{x} \in \mathbb{R}^{1 \times N}$. D denotes the dimensionality of word embeddings. Then, $\mathbf{S}$ and $\mathbf{X}$ are aligned to maximize the cumulative correlation between the aligned dimensions of the two matrices. Specifically, let $\mathbf{A} \in \{0, 1\}^{D \times P}$ be a matrix of alignments such that $a_{ij} = 1$ iff $\mathbf{x}_i$ is aligned to $\mathbf{s}_j$, otherwise $a_{ij} = 0$. If $r(\mathbf{x}_i, \mathbf{s}_j)$ is the Pearson's correlation coefficient between vectors $\mathbf{x}_i$ and $\mathbf{s}_j$, then QVEC is defined as:

$$\text{QVEC} = \max_{\mathbf{A}: \sum_j 0 \leq a_{ij} \leq 1,} \sum_{i=1}^{X} \sum_{j=1}^{S} r(\mathbf{x}_i, \mathbf{s}_j) \times a_{ij}$$

The constraint $\sum_j a_{ij} \leq 1$, warrants that one distributional dimension is aligned to at most one linguistic dimension.

QVEC has been shown to correlate strongly with downstream semantic tasks (Tsvetkov et al., 2015b). However, it suffers from two major weaknesses. First, it is not invariant to linear transformations of the embeddings' basis, whereas the bases in word embeddings are generally arbitrary (Szegedy et al., 2014). Second, a sum of correlations produces an unnormalized score: the more dimensions in the embedding matrix the higher the score. This precludes comparison of models of different dimensionality. QVEC-CCA simultaneously addresses both problems.

QVEC-CCA: Instead of using cumulative dimension-wise correlations to measure the correlation between the embedding matrix $\mathbf{X}$ and the linguistc matrix $\mathbf{S}$, QVEC-CCA uses canonical correlation analysis (CCA). CCA finds two sets of basis vectors, one for $\mathbf{X}^\top$ and the other for $\mathbf{S}^\top$, such that the correlations between the projections of the matrices onto these basis vectors are maximized. Formally, CCA finds a pair of basis vectors $\mathbf{v}$ and $\mathbf{w}$ such that

$$\text{QVEC-CCA} = \text{CCA}(\mathbf{X}^\top, \mathbf{S}^\top) = \max_{\mathbf{v},\mathbf{w}} r(\mathbf{X}^\top\mathbf{v}, \mathbf{S}^\top\mathbf{w})$$

Thus, QVEC-CCA ensures invariance to the matrices bases rotation, and since it is a single correlation, it produces a score in $[-1, 1]$. Both QVEC and QVEC-CCA rely on a matrix of linguistic properties constructed from a manually crafted linguistic resource.

multiQVEC and multiQVEC-CCA: Instead of only constructing the linguistic matrix based on monolingual annotations, multiQVEC and multiQVEC-CCA use supersense tag annotations in several languages. However, as of the time of this writing, the annotations are only available in three languages: English (Miller et al., 1993), Danish (Martínez Alonso et al., 2015; Martínez Alonso et al., 2016) and Italian (Montemagni et al., 2003).

4.3.4 Extrinsic tasks

In order to evaluate how useful the word embeddings are for a downstream task, we use the embedding vector as a dense feature representation of each word in the input, and deliberately remove any other feature available for this word (e.g., prefixes, suffixes, part-of-speech). For each task, we train one model on the aggregate training data available for several languages, and evaluate on the aggregate evaluation data in the same set of languages. We apply this for multilingual document classification and multilingual dependency parsing.

For document classification, we follow Klementiev et al. (2012) in using the RCV corpus of newswire text, and train a classifier which differentiates between four topics. While most previous work used this data only in a bilingual setup, we simultaneously train the classifier on documents in seven languages,[10] and evaluate on the development/test section of those languages. For this task, we report the average classification accuracy on the test set.

For dependency parsing, we use one epoch of stochastic gradient descent to train the stack-LSTM parser of Dyer et al. (2015) on a subset of the languages in the universal dependencies v1.1[11], and test on the same languages, reporting unlabeled attachment scores. We remove all part-of-speech and morphology features from the data, and prevent the model from optimizing the word embeddings used to represent each word in the corpus, thereby forcing the parser to rely completely on the provided (pretrained) embeddings as the token representation. Although

[10] Danish, German, English, Spanish, French, Italian and Swedish.

[11] http://hdl.handle.net/11234/LRT-1478

omitting other features (e.g., parts of speech) hurts the performance of the parser, it emphasizes the contribution of the word embeddings being studied.

4.4 Experiments

Our experiments are designed to show two primary sets of results: (i) how well the intrinsic evaluation metrics correlate with downstream tasks that use multilingual word vectors (§4.4.1) and (ii) which estimation methods perform better according to each evaluation metric (§4.4.2).

4.4.1 Correlations between intrinsic vs. extrinsic evaluation metrics

In this experiment, we consider four intrinsic evaluation metrics (cross-lingual word similarity, word translation, multiQVEC and multiQVEC-CCA) and two extrinsic evaluation metrics (multilingual document classification and multilingual parsing).

Data: All evaluation data sets we used are available for download on the evaluation portal. For the cross-lingual word similarity task, we use the 307 English-Italian word pairs in the multilingual MWS353 dataset (Leviant and Reichart, 2015). For the word translation task, we use a subset of 647 translation pairs from Wiktionary in English, Italian and Danish. For multiQVEC and multiQVEC-CCA, we used the 41 supersense tag annotations (26 for nouns and 15 for verbs) as described in §4.3. For the downstream tasks, we use the English, Italian and Danish subsets of the RCV corpus and the universal dependencies v1.1.

Setup: Estimating correlations between the intrinsic evaluation metrics and downstream task performance requires a sample of different vector embeddings with their intrinsic and extrinsic task scores. To create this sample, we trained a total of 17 different multilingual embeddings for three languages (English, Italian and Danish): twelve variants of multiCluster embeddings, one variant of multiCCA embeddings, one variant of multiSkip embeddings and two variants of translation-invariance embeddings.

Results: Table 4.2 shows Pearson's correlation coefficients of eight (intrinsic metric, extrinsic metric) pairs. Although each of two proposed methods multiQVEC and multiQVEC-CCA correlate better with a different extrinsic task, we establish (i) that intrinsic methods previously used in the literature (cross-lingual word similarity and word translation) correlate poorly with downstream tasks, and (ii) that the correlation-based metrics (multiQVEC and multiQVEC-CCA) correlate better with both downstream tasks, compared to cross-lingual word similarity and word translation.[12]

4.4.2 Evaluating multilingual estimation methods

We now turn to evaluating the four estimation methods described in §4.2. We use the proposed methods (i.e., multiCluster and multiCCA) to train multilingual embeddings in 59 languages

[12]Although supersense annotations exist for other languages, the annotations are inconsistent across languages and may not be publicly available, which is a disadvantage of the multiQVEC and multiQVEC-CCA metrics. Therefore, we recommend that future multilingual supersense annotation efforts use the same set of supersense tags used in other languages. If the word embeddings are primarily needed for encoding syntactic information, one could use tag dictionaries based on the universal POS tag set (Petrov et al., 2012) instead of supersense tags.

(→) extrinsic task (↓) intrinsic metric	document classification	dependency parsing
word similarity	0.386	0.007
word translation	0.066	-0.292
multiQVEC	0.635	0.444
multiQVEC-CCA	0.896	0.273

Table 4.2: Correlations between intrinsic evaluation metrics (rows) and downstream task performance (columns).

Task	multiCluster	multiCCA
dependency parsing	48.4 [**72.1**]	**48.8** [69.3]
doc. classification	90.3 [52.3]	**91.6** [**52.6**]
mono. wordsim	14.9 [**71.0**]	**43.0** [**71.0**]
cross. wordsim	12.8 [**78.2**]	**66.8** [**78.2**]
word translation	30.0 [**38.9**]	**83.6** [31.8]
mono. QVEC	7.6 [**99.6**]	**10.7** [99.0]
multiQVEC	8.3 [86.4]	**8.7** [**87.0**]
mono. QVEC-CCA	53.8 [**99.6**]	**63.4** [99.0]
multiQVEC-CCA	37.4 [86.4]	**42.0** [**87.0**]

Table 4.3: Results for multilingual embeddings that cover 59 languages. Each row corresponds to one of the embedding evaluation metrics we use (higher is better). Each column corresponds to one of the embedding estimation methods we consider; i.e., numbers in the same row are comparable. Numbers in square brackets are coverage percentages.

for which bilingual translation dictionaries are available.[13] In order to compare our methods to baselines which use parallel data (i.e., multiSkip and translation-invariance), we also train multilingual embeddings in a smaller set of 12 languages for which high-quality parallel data are available.[14]

Training data: We use Europarl en-xx parallel data for the set of 12 languages. We obtain en-xx bilingual dictionaries from two different sources. For the set of 12 languages, we extract the bilingual dictionaries from the Europarl parallel corpora. For the remaining 47 languages, dictionaries were formed by translating the 20k most common words in the English monolingual corpus with Google Translate, ignoring translation pairs with identical surface forms and multi-word translations.

Evaluation data: Monolingual word similarity uses the MEN dataset in Bruni et al. (2014) as a development set and Stanford's Rare Words dataset in Luong et al. (2013) as a test set. For the cross-lingual word similarity task, we aggregate the RG-65 datasets in six language pairs (fr-es, fr-de, en-fr, en-es, en-de, de-es). For the word translation task, we use Wiktionary to extract translationally-equivalent word pairs to evaluate multilingual embeddings for the set of 12 languages. Since Wiktionary-based translations do not cover all 59 languages, we use Google Translate to obtain en-xx bilingual dictionaries to evaluate the embeddings of 59 languages. For QVEC and QVEC-CCA, we split the English supersense annotations used in Tsvetkov et al. (2015b) into a development set and a test set. For multiQVEC and multiQVEC-CCA, we use supersense annotations in English, Italian and Danish. For the document classification task, we use the multilingual RCV corpus in seven languages (da, de, en, es, fr, it, sv). For the dependency parsing task, we use the universal dependencies v1.1 in twelve languages (bg, cs, da, de, el, en, es, fi, fr, hu, it, sv).

Setup: All word embeddings in the following results are 512-dimensional vectors. Methods which indirectly use skipgram (i.e., multiCCA, multiSkip, and multiCluster) are trained using 10 epochs of stochastic gradient descent, and use a context window of size 5. The translation-invariance method use a context window of size 3.[15] We only estimate embeddings for words or clusters which occur 5 times or more in the monolingual corpora. In a postprocessing step, all vectors are normalized to unit length. MultiCluster uses a maximum cluster size of 1,000 and 10,000 for the set of 12 and 59 languages, respectively. In the English tasks (monolingual word similarity, QVEC, QVEC-CCA), skipgram embeddings (Mikolov et al., 2013a) and multiCCA embeddings give identical results (since we project words in other languages to the English vector space, estimated using the skipgram model). The trained embeddings are available for download on the evaluation portal.

We note that intrinsic evaluation of word embeddings (e.g., word similarity) typically ignores test instances which are not covered by the embeddings being studied. When the vocabulary used in two sets of word embeddings is different, which is often the case, the intrinsic evaluation score

[13]The 59-language set is { bg, cs, da, de, el, en, es, fi, fr, hu, it, sv, zh, af, ca, iw, cy, ar, ga, zu, et, gl, id, ru, nl, pt, la, tr, ne, lv, lt, tg, ro, is, pl, yi, be, hy, hr, jw, ka, ht, fa, mi, bs, ja, mg, tl, ms, uz, kk, sr, mn, ko, mk, so, uk, sl, sw }.

[14]The 12-language set is {bg, cs, da, de, el, en, es, fi, fr, hu, it, sv}.

[15]Training translation-invariance embeddings with larger context window sizes using the matlab implementation provided by Gardner et al. (2015) is computationally challenging.

	Task	multiCluster	multiCCA	multiSkip	invariance
extrinsic metrics	dependency parsing	**61.0** [**70.9**]	58.7 [69.3]	57.7 [68.9]	59.8 [68.6]
	document classification	**92.1** [48.1]	**92.1** [**62.8**]	90.4 [45.7]	91.1 [31.3]
intrinsic metrics	monolingual word similarity	38.0 [57.5]	43.0 [**71.0**]	33.9 [55.4]	**51.0** [23.0]
	multilingual word similarity	58.1 [74.1]	**66.6** [**78.2**]	59.5 [67.5]	58.7 [63.0]
	word translation	43.7 [45.2]	35.7 [**53.2**]	46.7 [39.5]	**63.9** [30.3]
	monolingual QVEC	10.3 [98.6]	**10.7** [**99.0**]	8.4 [98.0]	8.1 [91.7]
	multiQVEC	**9.3** [82.0]	8.7 [**87.0**]	8.7 [**87.0**]	5.3 [74.7]
	monolingual QVEC-CCA	62.4 [98.6]	63.4 [**99.0**]	58.9 [98.0]	**65.8** [91.7]
	multiQVEC-CCA	43.3 [82.0]	41.5 [**87.0**]	36.3 [75.6]	**46.2** [74.7]

Table 4.4: Results for multilingual embeddings that cover Bulgarian, Czech, Danish, Greek, English, Spanish, German, Finnish, French, Hungarian, Italian and Swedish. Each row corresponds to one of the embedding evaluation metrics we use (higher is better). Each column corresponds to one of the embedding estimation methods we consider; i.e., numbers in the same row are comparable. Numbers in square brackets are coverage percentages.

for each set may be computed based on a different set of test instances, which may bias the results in unexpected ways. For instance, if one set of embeddings only covers frequent words while the other set also covers infrequent words, the scores of the first set may be inflated because frequent words appear in many different contexts and are therefore easier to estimate than infrequent words. To partially address this problem, we report the coverage of each set of embeddings in square brackets. When the difference in coverage is large, we repeat the evaluation using only the intersection of vocabularies covered by all embeddings being evaluated. Extrinsic evaluations are immune to this problem because the score is computed based on all test instances regardless of the coverage.[16]

Results [59 languages]. We train the proposed dictionary-based estimation methods (multiCluster and multiCCA) for 59 languages, and evaluate the trained embeddings according to nine different metrics in Table 4.3. The results show that, when trained on a large number of languages, multiCCA consistently outperforms multiCluster according to all evaluation metrics. Note that most differences in coverage between multiCluster and multiCCA are relatively small.

It is worth noting that the mainstream approach of estimating one vector representation per word type (rather than word token) ignores the fact that the same word may have different semantics in different contexts. The multiCluster method exacerbates this problem by estimating one vector representation per cluster of translationally equivalent words. The added semantic ambiguity severely hurts the performance of multiCluster with 59 languages, but it is still competitive with 12 languages (see below).

Results on [12 languages]. We compare the proposed dictionary-based estimation methods to parallel text-based methods in Table 4.4. The ranking of the four estimation methods is not consistent across all evaluation metrics. This is unsurprising since each metric evaluates different

[16]We followed previous work in reporting intrinsic results only on the subset of test words which are covered by the induced embeddings.

traits of word embeddings, as detailed in §4.3. However, some patterns are worth noting in Table 4.4.

In five of the evaluations (including both extrinsic tasks), the best performing method is a dictionary-based one proposed in this chapter. In the remaining four intrinsic methods, the best performing method is the translation-invariance method. MultiSkip ranks last in five evaluations, and never ranks first. Since our implementation of multiSkip does not make use of monolingual data, it only learns from monolingual contexts observed in parallel corpora, it misses the opportunity to learn from contexts in the much larger monolingual corpora. Trained for 12 languages, multiCluster is competitive in four evaluations (and ranks first in three).

We note that multiCCA consistently achieves better coverage than the translation-invariance method. For intrinsic measures, this confounds the performance comparison. A partial solution is to test only on word types for which all four methods have a vector; this subset is in no sense a representative sample of the vocabulary. In this comparison (provided in the supplementary material), we find a similar pattern of results, though multiCCA outperforms the translation-invariance method on the monolingual word similarity task. Also, the gap (between multiCCA and the translation-invariance method) reduces to 0.7 in monolingual QVEC-CCA and 2.5 in multiQVEC-CCA.

The linearity assumption of multiCCA. A shortcoming of the multiCCA method is that it assumes a linear transformation between the monolingual embedding of a word in one language and the monolingual embedding of its translation in another language, even though the two monolingual embeddings were independently trained. When this assumption is met, CCA projections of translationally equivalent words are perfectly correlated (i.e., have sample correlation coefficient of 1.0) for all dimensions of the new space. Fig. 4.3 plots the sample correlation coefficients for each dimension in the solution found by CCA, based on a dictionary of 35,524 English-Bulgarian translations, illustrating that this is not the case. While Faruqui and Dyer (2014) only uses a subset of dimensions with the highest correlations, multiCCA requires using all the dimensions for the projection to be invertible.

To relax this assumption while training embeddings for only two languages, Lu et al. (2015) finds nonlinear projections to maximize the correlation between translationally equivalent words in a new vector space, based on the deep canonical correlation analysis method of Andrew et al. (2013). However, unlike the linear projections we used in multiCCA, the nonlinear projections obtained via deep canonical correlation analysis are not invertible. To alleviate the need for an inverse projection to the pivot language, one possibility is to simultaneously optimize nonlinear projections from all languages of interest to the same new vector space. However, more translation pairs may be necessary for optimizing a larger number of parameters.

4.5 Related Work

There is a rich body of literature on bilingual embeddings, including work on machine translation (Zou et al., 2013; Hermann and Blunsom, 2014; Cho et al., 2014; Luong et al., 2015a; Luong et al., 2015b, *inter alia*),[17] cross-lingual dependency parsing (Guo et al., 2015; Guo et al., 2016),

[17]Hermann and Blunsom (2014) showed that the bicvm method can be extended to more than two languages, but the released software library only supports bilingual embeddings.

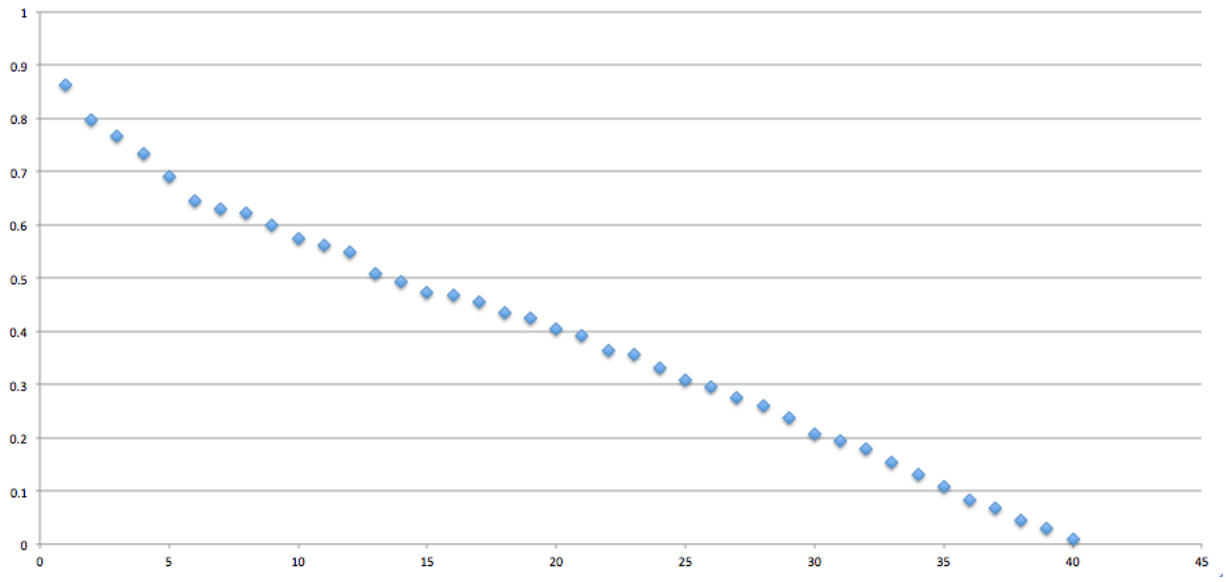


Figure 4.3: Sample correlation coefficients (perfect correlation = 1.0) for each dimension in the solution found by CCA, based on a dictionary of 35,524 English-Bulgarian translations.

and cross-lingual document classification (Klementiev et al., 2012; Gouws et al., 2014; Kočiskỳ et al., 2014). Al-Rfou' et al. (2013) trained word embeddings for more than 100 languages, but the embeddings of each language are trained independently (i.e., embeddings of words in different languages do not share the same vector space). Word clusters are a related form of distributional representation; in clustering, cross-lingual distributional representations were proposed as well (Och, 1999; Täckström et al., 2012b). Haghighi et al. (2008) used CCA to learn bilingual lexicons from monolingual corpora.

4.6 Summary

We introduced two estimation methods for multilingual word embeddings, multiCCA and multiCluster, which only require bilingual dictionaries and monolingual corpora, and used them to train embeddings for 59 languages. We found the embeddings estimated using our dictionary-based methods to outperform those estimated using other methods for two downstream tasks: multilingual dependency parsing and multilingual document classification. We also created a web portal for users to upload their multilingual embeddings and easily evaluate them on nine evaluation metrics, with two modes of operation (development and test) to encourage sound experimentation practices.

Chapter 5

Conditional Random Field Autoencoders

5.1 Overview

The previous chapter used bilingual dictionaries to estimate multilingual word embeddings. When a sizable parallel corpus is available for a pair of languages, unsupervised word alignment is often used to extract the bilingual dictionaries needed for estimating multilingual word embeddings. In this chapter, we describe a CRF-based model for unsupervised structured prediction, and apply it to unsupervised word alignment and POS induction.

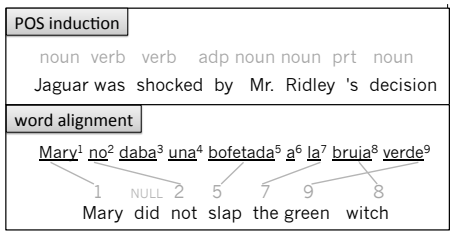
Conditional random fields (CRF, Lafferty et al. 2001) are a popular choice for modeling linguistic structure, as well as other structures in computational biology, and computer vision. CRFs enable efficient inference while incorporating rich features that capture useful domain-specific insights. Despite their ubiquity in supervised settings, CRFs play less of a role in *unsupervised* structure learning, a problem which traditionally requires jointly modeling observations and the latent structures of interest. Efficient inference in such joint models requires adhering to inconvenient independence assumptions when designing features, limiting the expressive power of joint models. For example, a first-order hidden Markov model (HMM) requires that $y_i \perp x_{i+1} \mid y_{i+1}$ for a latent sequence $\mathbf{y} = \langle y_1, y_2, \ldots \rangle$ generating a sequence of observations $\mathbf{x} = \langle x_1, x_2, \ldots \rangle$, while a first-order CRF allows y_i to directly depend on $\ldots, x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+1}, \ldots$.

We describe a model for unsupervised structure learning which leverages the power and flexibility of CRFs without sacrificing their attractive computational properties or changing the semantics of well-understood feature sets. Our approach replaces the standard joint model of observed data and latent structure with a two-layer **conditional random field autoencoder** that first generates latent structure with a CRF (conditional on the observed data) and then (re)generates the observations conditional on just the predicted structure. The proposed architecture provides several mechanisms for encouraging the learner to use its latent variables to find intended (rather than common but irrelevant) correlations in the data. First, hand-crafted feature representations—engineered using knowledge about the problem—provide a key mechanism for incorporating inductive bias. Second, we reconstruct transformations of the structured observation which are known to correlate with the hidden structure, while preserving the original observation at the input layer. Third, the same model can be used to simultaneously learn from labeled and unlabeled examples. In addition to the modeling flexibility, our approach is computationally efficient; under a set of mild independence assumptions regarding the reconstruction model, inference required for learning is no more expensive than when training a supervised CRF with the same independence assumptions.

The material in this chapter was previously published in Ammar et al. (2014), Lin et al. (2014) and Lin et al. (2015).

5.2 Approach

Our goal is to make use of unlabeled examples for predicting the hidden linguistic structure of text in low-resource languages. We are given a training set of structured observations (e.g., sentences), and structural constraints on the linguistic structure. Examples of linguistic structures include syntactic categories of words in the input sentence, correspondences between words in a source sentence and its translation in a target language, and spanning trees that describe (head, modifier) relations in a sentence. The main intuition behind the proposed model is that **a good**



	POS induction	**Word alignment**
$\mathcal{U}$	corpus of sentences	parallel corpus
$\mathbf{x}$	sentence	target sentence
$\mathcal{X}$	vocabulary	target vocabulary
$\boldsymbol{\phi}$	(opt.) tag dictionary	source sentence
$\mathbf{y}$	POS sequence	sequence of indices in the source sentence
$\mathcal{Y}$	POS tag set	$\{\text{NULL}, 1, 2, \ldots, \lvert\boldsymbol{\phi}\rvert\}$

Figure 5.1:
Left: Examples of structured observations (in black), hidden structures (in grey), and side information (underlined).
Right: Model variables for POS induction and word alignment. A parallel corpus consists of pairs of sentences ("source" and "target").

linguistic structure, modeled using a CRF, should serve as a good *encoding* of the input. Such an encoding can then be used to *reconstruct* the input with high probability.

5.2.1 Notation

Let each observation be denoted $\mathbf{x} = \langle x_1, \ldots, x_{|\mathbf{x}|} \rangle \in \mathcal{X}^{|\mathbf{x}|}$, a variable-length tuple of discrete variables, $x \in \mathcal{X}$. The hidden variables $\mathbf{y} = \langle y_1, \ldots, y_{|\mathbf{y}|} \rangle \in \mathcal{Y}^{|\mathbf{y}|}$ form a tuple whose length is determined by $\mathbf{x}$, also taking discrete values.[1] We assume that the set of possible instantiations of $\mathbf{y}$ which meet the structural constraints is significantly smaller than $|\mathcal{X}|^{|\mathbf{x}|}$, which is typical in NLP problems. Fig. 5.1 (right) describes $\mathbf{x}$, $\mathcal{X}$, $\mathbf{y}$ and $\mathcal{Y}$ for two NLP tasks.

Our model introduces a new observed variable, $\hat{\mathbf{x}} = \langle \hat{x}_1, \ldots, \hat{x}_{|\hat{\mathbf{x}}|} \rangle$, which represents a reconstruction of the input $\mathbf{x}$.

5.2.2 Model

Although the proposed approach applies to hidden structures with various structural constraints (e.g., morphological segmentations, dependency trees, constituent trees), we focus on *sequential* latent structures with first-order Markov properties, i.e., $y_i \perp y_j \mid \{y_{i-1}, y_{i+1}\}$, as illustrated in Fig. 5.2 (right). This class of latent structures is a popular choice for modeling a variety of problems such as human action recognition (Yamato et al., 1992), bitext word alignment (Brown et al., 1993; Vogel et al., 1996; Blunsom and Cohn, 2006), POS tagging (Merialdo, 1994; Johnson, 2007), acoustic modeling (Jelinek, 1997), gene finding (Lukashin and Borodovsky, 1998), and transliteration (Reddy and Waxmonsky, 2009; Ammar et al., 2012a), among others. Importantly, we make no assumptions about conditional independence between any y_i and $\mathbf{x}$.

Eq. 5.1 gives the parameteric form of our model for POS induction. $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$ are the parameters of the encoding and reconstruction models, respectively. $\mathbf{g}$ is a vector of clique-local feature

[1] In the interest of notational simplicity, we conflate random variables with their values.

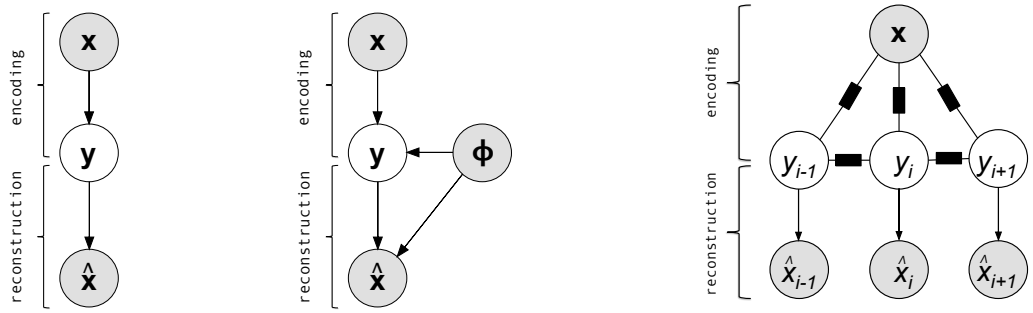


Figure 5.2: Graphical model representations of CRF autoencoders.
Left: the basic autoencoder model where the observation $\mathbf{x}$ generates the hidden structure $\mathbf{y}$ (encoding), which then generates $\hat{\mathbf{x}}$ (reconstruction).
Center: side information (ϕ) is added.
Right: a factor graph showing first-order Markov dependencies among elements of the hidden structure $\mathbf{y}$.

functions.[2]

$$p_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\hat{\mathbf{x}} \mid \mathbf{x}) = \sum_{\mathbf{y}} p_{\boldsymbol{\lambda}}(\mathbf{y} \mid \mathbf{x}) p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \mid \mathbf{y}) = \sum_{\mathbf{y} \in \mathcal{Y}^{|\mathbf{x}|}} \frac{e^{\sum_{i=1}^{|\mathbf{x}|} \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}, y_i, y_{i-1}, i)}}{\sum_{\mathbf{y}' \in \mathcal{Y}^{|\mathbf{x}|}} e^{\sum_{i=1}^{|\mathbf{x}|} \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{x}, y'_i, y'_{i-1}, i)}} \prod_{i=1}^{|\mathbf{x}|} p_{\boldsymbol{\theta}}(\hat{x}_i \mid y_i) \tag{5.1}$$

Encoding and reconstruction. We model the *encoding* part with a CRF, which allows us to exploit features with global scope in the structured observation $\mathbf{x}$, while keeping exact inference tractable for many problems (since the model does not generate $\mathbf{x}$, only conditions on it). The *reconstruction* part, on the other hand, grounds the model by generating a copy of the structured observations. We use simple distributions (multinomials and multivariate Gaussians) to independently generate $\hat{x}_i$ given y_i. Fig. 5.2 (right) is an instance of the model for POS induction with a sequential latent structure; each $\hat{x}_i$ is generated from $p_{\boldsymbol{\theta}}(\hat{x}_i \mid y_i)$.

The need to efficiently add inductive bias via feature engineering, while learning from unlabeled examples, has been the primary drive for developing CRF autoencoders. We emphasize the importance of allowing the model designer to define intuitive feature templates in a flexible manner. For example features which describe morphology, word spelling information, and other linguistic knowledge were shown to improve POS induction (Smith and Eisner, 2005), word alignment (Dyer et al., 2011), and other unsupervised learning problems. The proposed model enables the model designer to define such features at a lower computational cost, *and* enables more expressive features with global scope in the structured input. For example, we found that using predictions of other models as features is an effective method for model combination in unsupervised word alignment tasks, and found that conjoining sub-word-level features of consecutive words help disambiguate their POS labels.

[2]We define y_0 to be a fixed "start" tag. Note that the cliques here are *inside* $\mathbf{y}$; they are not visible in the high-level view of Fig. 5.2 (left), but are visible in Fig. 5.2 (right).

Extension: side information. Our model can be easily extended to condition on more context in the encoding part, the reconstruction part, or in both parts. Let $\boldsymbol{\phi}$ represent *side information*: additional context which we condition on in both the encoding and reconstruction models. In our running example, side information includes a POS tag dictionary (i.e., list of possible tags for each word), a common form of "weak supervision" shown to help unsupervised POS learners (Smith and Eisner, 2005; Ravi and Knight, 2009; Li et al., 2012; Garrette and Baldridge, 2013). In word alignment, where $y_i = j$ indicates that x_i translates to the jth source token, we treat the source sentence as side information, making its word forms available for feature extraction.

Extension: partial reconstruction. In our running POS example, the reconstruction model $p_{\boldsymbol{\theta}}(\hat{x}_i \mid y_i)$ defines a distribution over words given tags. Because word distributions are heavy-tailed, estimating such a distribution reliably is quite challenging. Our solution is to define a deterministic function that maps $\pi : \mathcal{X} \rightarrow \hat{\mathcal{X}}$ such that the dimensionality of $\hat{\mathcal{X}}$ is smaller than that of $\mathcal{X}$. For POS tagging, we experiment with two kinds of partial reconstructions: Brown clusters and dense word embeddings.

Other linguistic structures. We presented the CRF autoencoder in terms of sequential Markovian assumptions for ease of exposition; however, this framework can be used to model arbitrary hidden structures. For example, instantiations of this model can be used for unsupervised learning of parse trees (Klein and Manning, 2004), semantic role labels (Swier and Stevenson, 2004), and coreference resolution (Poon and Domingos, 2008) (in NLP), motif structures (Bailey and Elkan, 1995) in computational biology, and objects (Weber et al., 2000) in computer vision. The requirements for applying the CRF autoencoder model are:

- An encoding graphical model defining $p_{\boldsymbol{\lambda}}(\mathbf{y} \mid \mathbf{x})$. The encoder may be any model family where *supervised* learning from $\langle \mathbf{x}, \mathbf{y} \rangle$ pairs is efficient.
- A reconstruction model that defines $p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \mid \mathbf{y}, \boldsymbol{\phi})$ such that inference over $\mathbf{y}$ given $\langle \mathbf{x}, \hat{\mathbf{x}} \rangle$ is efficient.
- The independencies among $\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{x}}$ are not strictly weaker than those among $\mathbf{y} \mid \mathbf{x}$.

5.2.3 Learning

When no labeled examples are available for training, model parameters are selected to maximize the regularized conditional log likelihood of reconstructed observations $\hat{\mathbf{x}}$ given the structured observation $\mathbf{x}$:

$$\ell\ell(\boldsymbol{\lambda}, \boldsymbol{\theta}) = R_1(\boldsymbol{\lambda}) + R_2(\boldsymbol{\theta}) + \sum_{(\mathbf{x}, \hat{\mathbf{x}}) \in \mathcal{U}} \log \sum_{\mathbf{y}} p_{\boldsymbol{\lambda}}(\mathbf{y} \mid \mathbf{x}) \times p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \mid \mathbf{y}) \tag{5.2}$$

In our experiments, we use a squared L_2 regularizer for the CRF parameters $\boldsymbol{\lambda}$, and use a symmetric Dirichlet prior for the parameters $\boldsymbol{\theta}$.

It is easy to modify this objective to learn from both labeled examples $\mathcal{L}$ and unlabeled exam-

ples $\mathcal{U}$ as follows:

$$\begin{aligned}\ell\ell_{\text{semi}}(\boldsymbol{\lambda}, \boldsymbol{\theta}) =& R_1(\boldsymbol{\lambda}) + R_2(\boldsymbol{\theta}) \\ &+ \frac{w_{\text{unlabeled}}}{|\mathcal{U}|} \times \sum_{(\mathbf{x},\hat{\mathbf{x}})\in\mathcal{U}} \log \sum_{\mathbf{y}} p_{\boldsymbol{\lambda}}(\mathbf{y} \mid \mathbf{x}) \times p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \mid \mathbf{y}) \\ &+ \frac{w_{\text{labeled}}}{|\mathcal{L}|} \times \sum_{(\mathbf{x},\hat{\mathbf{x}},\mathbf{y})\in\mathcal{L}} \log p_{\boldsymbol{\lambda}}(\mathbf{y} \mid \mathbf{x}) + \log p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \mid \mathbf{y}) \end{aligned} \tag{5.3}$$

where $w_{\text{unlabeled}}$ and w_{labeled} are hyperparameters to control the contribution of labeled vs. unlabeled examples.

Convexity analysis. The optimization problem we need to solve to train a CRF autoencoder model for unsupervised POS induction with a simple categorical distribution for reconstructing $\hat{\mathbf{x}} = \mathbf{x}$ is:

$$\begin{aligned}&\arg\min_{\lambda,\theta} - \sum_{(\mathbf{x},\hat{\mathbf{x}})\in\mathcal{U}} \log \sum_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) \times p(\hat{\mathbf{x}} \mid \mathbf{y}) \\ &\quad = - \sum_{(\mathbf{x},\hat{\mathbf{x}})\in\mathcal{U}} \log \sum_{\mathbf{y}} \frac{\exp \boldsymbol{\lambda}^\top \sum_{i=1}^{|\mathbf{x}|} \mathbf{g}(\mathbf{x}, y_i, y_{i-1})}{\sum_{\mathbf{y}'} \exp \boldsymbol{\lambda}^\top \sum_{i=1}^{|\mathbf{x}|} \mathbf{g}(\mathbf{x}, y'_i, y'_{i-1})} \times \prod_{i=1}^{|\mathbf{x}|} \theta_{\hat{x}_i|y_i} \\ &\text{subject to} \sum_{\hat{x}\in\mathcal{X}} \theta_{\hat{x}|y} = 1, 0 \le \theta_{\cdot|y} \le 1, \forall y \in \mathcal{Y}\end{aligned} \tag{5.4}$$

For each label $y \in \mathcal{Y}$, the constraints on $\theta_{\cdot|y}$ describe a probability simplex (a special case of polyhedra) which is a convex set. The feasible set is the intersection of the probability simplexes for all $y \in \mathcal{Y}$ which is also convex. Given a convex feasible set, in order to show that Eq. 5.4 is convex, it suffices to show that $\log \sum_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}) \times p(\hat{\mathbf{x}} \mid \mathbf{y})$ is convex. We can rewrite this as:

$$\log \sum_{\mathbf{y}} e^{\lambda^\top \sum_{i=1}^{|\mathbf{x}|} \mathbf{g}(\mathbf{x}, y_i, y_{i-1})} - \log \sum_{\mathbf{y}} e^{\lambda^\top \sum_{i=1}^{|\mathbf{x}|} \mathbf{g}(\mathbf{x}, y_i, y_{i-1})} \times \prod_{i=1}^{|\mathbf{x}|} \theta_{\hat{x}_i|y_i} \tag{5.5}$$

The first term is convex in (λ, θ), since it is a log-sum-exp function (with an affine transformation of λ). The second term is concave in λ but non-concave (and non-convex) in θ since multiplication does not preserve concavity. Even if we hold θ constant, the objective is the difference between two convex terms which is non-convex in general.

We note one exception which makes $h_{\mathbf{s},\mathbf{t}}$ convex in θ: when $\lambda = 0$, $p_\lambda(\mathbf{y} \mid \mathbf{x}) = \frac{1}{(|\mathbf{x}|)\times|\mathcal{Y}|}$; i.e., the conditional distribution over possible values of $\mathbf{y}$ is uniform. The second term then reduces to $-\log \sum_{\mathbf{y}} \prod_{i=1}^{|\mathbf{x}|} \theta_{\hat{x}_i|y_i}$ which can be re-written as $-\sum_{i=1}^{|\mathbf{x}|} \log \sum_{y_i} \theta_{\hat{x}_i|y_i}$. This is a non-positive combination of concave functions (i.e., the logarithmic function) and is therefore convex.

5.2.4 Optimization

Although the optimization problem in Eq. 5.4 is non-convex in general, we found locally-optimal solutions to be useful in practice, provided that we start with a good initialization for model

parameters.[3] We use block-coordinate descent, iteratively alternating between optimizing with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$.

Optimizing w.r.t. $\boldsymbol{\lambda}$. We use gradient-based methods to optimize CRF parameters $\boldsymbol{\lambda}$ in the reduced, unconstrained problem:

$$\min_{\boldsymbol{\lambda}} \ell\ell(\boldsymbol{\lambda}) = \sum_{\langle \mathbf{x}, \hat{\mathbf{x}} \rangle \in \mathcal{U}} \log \sum_{\mathbf{y}} p(\mathbf{y}, \hat{\mathbf{x}} \mid \mathbf{x}) \tag{5.6}$$

We use L-BFGS (Liu et al., 1989), a batch quasi-Newoton method well suited for problems with a large number of parameters.[4] Since processing all examples in a large training set can be expensive, we also experiment with stochastic gradient descent (SGD) using an approximation of the gradient based on a few examples only. One epoch (i.e., full pass over the training set) of SGD constitutes **many updates** of λ, and incurs approximately the same runtime cost as **one update** of L-BFGS.

Optimizing w.r.t. $\boldsymbol{\theta}$. Here, we are only concerned about the following reduced problem:

$$\min_{\boldsymbol{\theta}} \ell\ell(\boldsymbol{\theta}) = \sum_{\langle \mathbf{x}, \hat{\mathbf{x}} \rangle \in \mathcal{U}} \log \sum_{\mathbf{y}} p(\mathbf{y}, \hat{\mathbf{x}} \mid \mathbf{x}) \text{ s.t.} \sum_{\hat{x}} \theta_{\hat{x}|y} = 1, 0 \leq \theta_{\cdot|y} \leq 1, \forall y \in \mathcal{Y} \tag{5.7}$$

We use batch Expectation Maximization (EM), a popular method for optimizing parameters of generative models with latent variables. In each iteration of batch EM, we update θ by solving: $\min_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}^{\text{old}}}[\log p_{\boldsymbol{\theta}}(\mathbf{y}, \hat{\mathbf{x}} \mid \mathbf{x})]$ subject to the multinomial distribution constraints on θ. Each iteration consists of two steps:

- E-step: compute the sufficient statistics (μ) for estimating θ given θ^{old}. The sufficient statistic for each parameter in θ turn out to be the expected number of times that parameter is being used to generate an observation.[5]
- M-step: estimate θ given μ (by projecting to the probability simplex).

We also experimented with an online EM variant proposed by Cappé and Moulines (2009), also known as "stepwise EM". The following pseudocode, adapted from Liang and Klein (2009), outlines both batch and online EM algorithms.

Batch EM:

$\mu :=$ initialize
for each EM iteration $t = 1, \ldots, T$:
— $\mu' := 0$
— for each example $i : \langle \mathbf{x}, \hat{\mathbf{x}} \rangle$
—— $m'_i := \sum_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{x}}; \theta(\mu)) f(\mathbf{y}, \mathbf{x}, \hat{\mathbf{x}})$[inference]
—— $\mu' := \mu' + m'_i$ [accumulate new]
— $\mu := \mu'$ [replace old with new]

[3]We use zero initialization of the CRF parameters, and initialize the reconstruction model parameters with a basic first-order HMM model.

[4]In our experiments, the number of parameters in $\boldsymbol{\lambda}$ is in the order of 10^6.

[5]The expectation here is governed by $p_{\theta^{old}}(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{x}})$.

Online EM:

$\mu :=$ initialize, $k := 0$

for each EM iteration $t = 1, \ldots, T$:

— for each example $i : \langle \mathbf{s}, \mathbf{t}, \hat{\mathbf{t}} \rangle$ in random order

—— $m'_i := \sum_{\mathbf{y}} p(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{x}}; \theta(\mu)) f(\mathbf{y}, \mathbf{x}, \hat{\mathbf{x}})$ [inference]

—— $\mu := (1 - \eta_k)\mu + \eta_k m'_i; k := k + 1$ [interpolate]

In the outlines above, μ is a vector of expected counts for corresponding parameters in θ, f is a function that maps a sentence pair and its alignment to a feature vector, m'_i are the expected counts for a given sentence pair, and $\theta(\mu)$ is shorthand for the parameter values θ projected from sufficient statistics μ as in the M-step. Note that a projection is only necessary when μ changes though. So this operation is only performed once in batch EM, but many times in online EM.

5.3 Experiments

We evaluate our approach on three tasks: POS induction, word alignment, and token-level language identification in code-switched text.

5.3.1 POS Induction

POS induction is a classic NLP problem which aims at discovering syntactic classes of tokens in a monolingual corpus, with a predefined number of classes. An example of a POS-tagged English sentence is in Fig. 5.1.

Data. We use the plain text from CoNLL-X (Buchholz and Marsi, 2006) and CoNLL 2007 (Nivre et al., 2007) training data in seven languages to train the models: Arabic, Basque, Danish, Greek, Hungarian, Italian and Turkish. For evaluation, we obtain gold-standard POS tags by deterministically mapping the language-specific POS tags from the shared task training data to the corresponding universal POS tag set[6] (Petrov et al., 2012). Some experiments also use the Zulu corpus of Spiegler et al. (2010).

Setup. We configure our model (as well as baseline models) to induce $|\mathcal{Y}| = 12$ classes. We use zero initialization of the CRF parameters, and initialize the reconstruction model parameters using the emission parameters of an HMM (trained with five iterations of batch EM). In each block-coordinate ascent iteration, we run one L-BFGS iteration (including a line search) to optimize $\boldsymbol{\lambda}$, followed by one EM iteration to optimize $\boldsymbol{\theta}$. We stop training after 70 block-coordinate ascent iterations.

Evaluation. Since we do not use a tagging dictionary, the word classes induced by our model are unidentifiable. We use two cluster evaluation metrics commonly used for POS induction: a) V-measure (Rosenberg and Hirschberg, 2007) is an entropy-based metric which explicitly measures the homogeneity and completeness of predicted clusters (again, higher is better), b) many-to-one (Johnson, 2007) infers a mapping across the syntactic clusters in the gold vs. predicted labels (higher is better).

[6] `http://code.google.com/p/universal-pos-tags/`

CRF Autoencoder Model Instantiation. Table 5.1 (right) describes the symbols and variables we use in context of the POS induction problem. We use a first-order linear CRF for the encoding part with the following feature templates:

- $\langle y_i, y_{i-1} \rangle, \forall i$
- $\langle y_i, \text{sub}_j(x_i) \rangle, \forall i, j$
- $\langle y_i, \text{sub}_j(x_i), \text{sub}_k(x_{i-1}) \rangle, \forall i, j, k$
- $\langle y_i, \text{sub}_j(x_i), \text{sub}_k(x_{i+1}) \rangle, \forall i, j, k$

Where $\text{sub}_j(x_i)$ is one of the following sub-word-level feature percepts:

- Prefixes and suffixes of lengths two and three, iff the affix appears in more than 0.02% of all word types,
- Whether the word contains a digit,
- Whether the word contains a hyphen,
- Whether the word starts with a capital letter,
- Word shape features which map sequences of the same character classes into a single character (e.g., 'McDonalds' $\rightarrow$ 'AaAa', '-0.5' $\rightarrow$ '#0#0'),
- The lowercased word, iff it appears more than 100 times in the corpus.

We experiment with two reconstruction models. In both models, we condition on a POS tag and reconstruct a representation of the corresponding word. First, we use a categorical distribution over 100 Brown clusters.[7] Second, we use a multivariate Gaussian distribution to generate a pretrained vector representation. The probability density assigned to a vector $\hat{x} \in \mathbb{R}^d$ by a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is:

$$p(\hat{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\exp\left(-\frac{1}{2}(\hat{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\hat{x} - \boldsymbol{\mu})\right)}{\sqrt{(2\pi)^d |\boldsymbol{\Sigma}|}}. \tag{5.8}$$

Output predictions are the best value of the latent structure according to the posterior $p(\mathbf{y} \mid \mathbf{x}, \hat{\mathbf{x}}, \boldsymbol{\phi})$.

Baselines. We use two baselines:

- hmm: a standard first-order hidden Markov model learned with EM;[8]
- fhmm: a hidden Markov model with logistic regression emissions, as implemented by Berg-Kirkpatrick et al. (2010).

Hyperparameters. We use a squared L_2 regularizer for CRF parameters $\boldsymbol{\lambda}$, and a symmetric Dirichlet prior for categorical parameters $\boldsymbol{\theta}$ with the same regularization strength for all languages. The fhmm baseline also uses a squared L_2 regularizer for the log-linear parameters. The hyperparameters of our model, as well as baseline models, were tuned to maximize many-to-one accuracy for The English Penn Treebank. The fhmm model uses L_2 strength $= 0.3$. The crfa model uses L_2 strength $= 2.5$, $\alpha = 0.1$.

[7] To obtain the Brown clusters, we use Liang (2005) with data from `http://corpora.informatik.uni-leipzig.de/`

[8] Among 32 Gaussian initializations of model parameters, we use the HMM model which gives the highest likelihood after 30 EM iterations.

Multinomial emissions. Fig. 5.3 compares predictions of the CRF autoencoder model with multinomial emissions in seven languages to those of a featurized first-order HMM model Berg-Kirkpatrick et al. (2010) and a standard (feature-less) first-order HMM, using the V-measure evaluation metric (Rosenberg and Hirschberg, 2007) (higher is better). First, we note the large gap between both feature-rich models on the one hand, and the feature-less HMM model on the other hand. Second, we note that CRF autoencoders outperform featurized HMMs in all languages, except Italian, with an average relative improvement of 12%.

We conclude that feature engineering is an important source of inductive bias for unsupervised structured prediction problems, which is a primary motivation for the proposed model. Similar conclusions can be drawn from Fig. 5.4 which uses the many-to-one evaluation metric (Johnson, 2007), albeit the difference in performance between CRF autoencoders and featurized HMMs, on average, is much smaller.

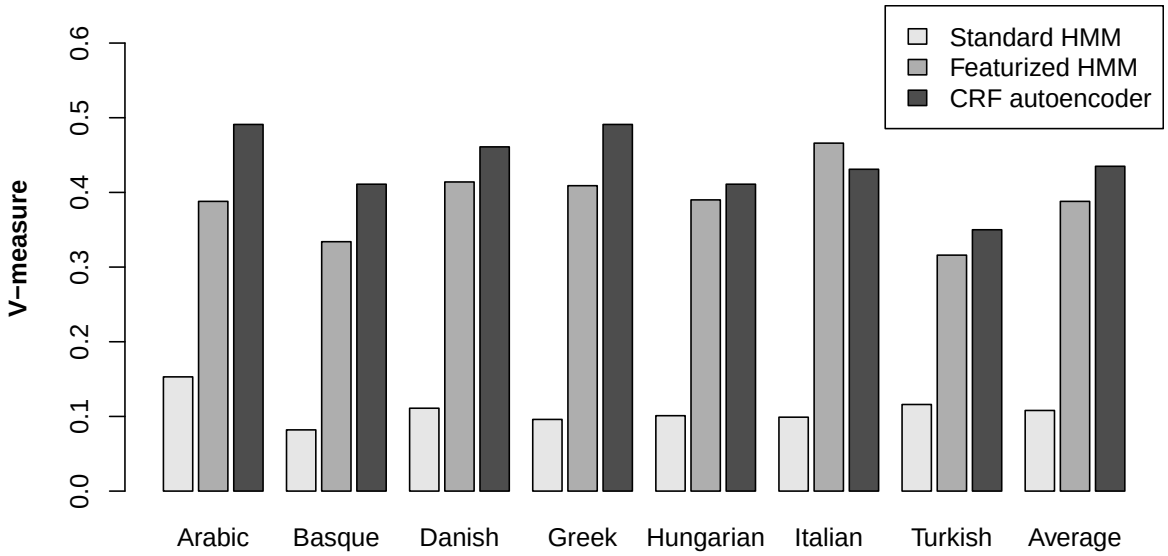


Figure 5.3: V-measure of induced parts of speech in seven languages, with multinomial emissions.

Gaussian emissions. We now replace the multinomial emissions in both the standard HMM model and the CRF autoencoder model with a multivariate Gaussian model that generates pre-trained word embeddings. We use the Skip-gram model to pretrain embddings with the word2vec tool. We optimize the mean parameters of the Gaussian using EM. We tuned the hyperparameters on the English PTB corpus, then fixed them for all languages. The hyperparameters for training word embeddings are: window size $= 1$, number of dimensions $= 100$. In lieu of inferring the

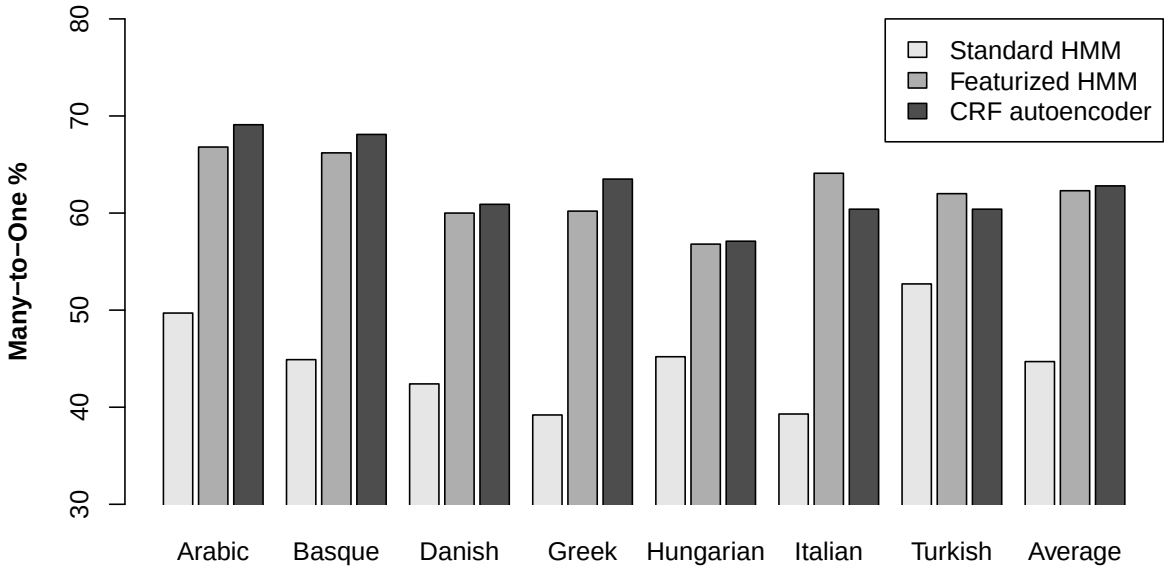


Figure 5.4: Many-to-one accuracy% of induced parts of speech in seven languages, with multinomial emissions.

covariance parameters, we used unit spherical covariance and scaled the vectors by a scalar (5) tuned on the English corpus.

Fig. 5.5 contrasts the three models with multinomial emissions: HMM with categorical emissions, HMM with featurized multinomial emissions, CRF autoencoder with multinomial reconstructions; with two models which use word embeddings: HMM with Gaussian emissions, and CRF autoencoder with Gaussian reconstructions. The results show that using Gaussian models to reconstruct pretrained word embeddings consistently improves POS induction. Surprisingly, the feature-less Gaussian HMM model outperforms the strong feature-rich models: Multinomial Featurized HMM and Multinomial CRF Autoencoder. The Gaussian CRF Autoencoder still has the best V-measure, closely followed by the Gaussian HMM model. This set of results suggests that word embeddings and hand-engineered features play complementary roles in POS induction.

How to pretrain word embeddings for POS induction? We experiment with two models for inducing word embeddings:

- **Skip-gram embeddings** (Mikolov et al., 2013a) are based on a log bilinear model that predicts an unordered set of context words given a target word (see chapter 2 for more details). Bansal et al. (2014) found that smaller context window sizes tend to result in embeddings with syntactic information. We confirm this finding in our experiments.
- **SENNA embeddings** (Collobert et al., 2011) are based on distinguishing true n-grams from corruptioned forms with a multilayer neural network. In contrast to skip-gram embeddings,

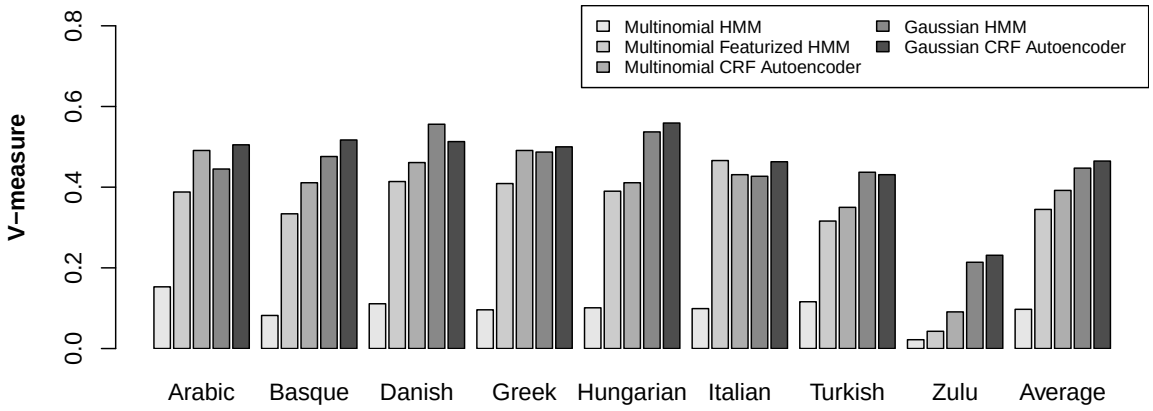


Figure 5.5: POS induction results of 5 models (VMeasure, higher is better). Models which use word embeddings (i.e., Gaussian HMM and Gaussian CRF Autoencoder) outperform all baselines on average across languages.

these are trained sensitive to word order.

We downloaded the English SENNA embeddings made available by Collobert et al. (2011). Obtaining SENNA embeddings in other languages was computationally infeasible. We expect SENNA embeddings to encode more syntactic information than skip-gram embeddings for two reasons: (i) Unlike skip-gram, SENNA is sensitive to word order, (ii) The English SENNA embeddings we used were trained in a semi-supervised multi-task learning setup, where one of the tasks was POS tagging as discussed in §4.5 in (Collobert et al., 2011) which gives these embeddings an unfair advantage compared to skip-gram.

Using a CRF autoencoder model, we compared SENNA embeddings to skip–gram embeddings on a subset of the English PTB corpus. Indeed, without any scaling, SENNA induces better parts of speech, yielding a V-measure score of 0.57, compared to 0.51 for skip-gram. This shows the impact of the model used to obtain embeddings on downstream tasks. However, we use skip-gram model in the remaining experiments because it is much faster to train and only requires unlabeled data.

We also measure how the number of dimensions (d) in pretrained word embeddings ($d \in \{20, 50, 100, 200\}$) affect POS induction. The results in Fig. 5.6 (left) suggest that the number of dimensions used in word embeddings has a modest effect on POS induction results.

Finally, we vary the window size for the context surrounding target words ($w \in \{1, 2, 4, 8, 16\}$). Fig. 5.6 (right) illustrates that the window size has a great impact on performance, with the best result obtained with $w = 2$. Notably, larger window sizes appear to produce word embeddings with less syntactic information. This result confirms the observations of Bansal et al. (2014).

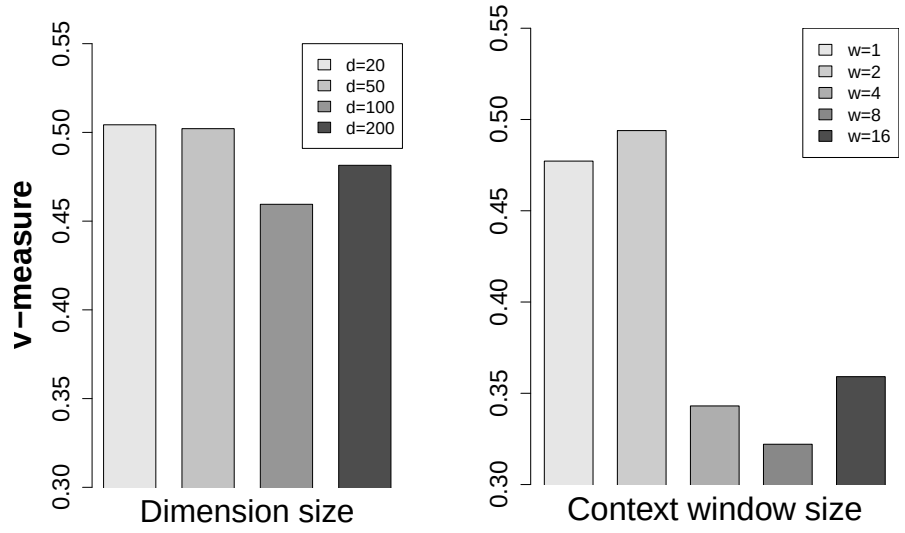


Figure 5.6:
Left: Effect of dimension size on POS induction on a subset of the English PTB corpus. Window size is set to 1 for all configurations.
Right: Effect of context window size on V-measure of POS induction on a subset of the English PTB corpus. $d = 100$, scale ratio $= 5$.

5.3.2 Word Alignment

Word alignment is an important step in the training pipeline of most statistical machine translation systems (Koehn, 2010).[9] Given a sentence in the source language and its translation in the target language, the task is to find which *source* token, if any, corresponds to each token in the *target* translation. We make the popular assumption that each token in the target sentence corresponds to zero or one token in the source sentence. Fig. 5.1 shows a Spanish sentence and its English translation with word alignments. As shown in Table 5.1 (Right), an observation $\mathbf{x}$ consists of tokens in the target sentence, while side information $\boldsymbol{\phi}$ are tokens in the source sentence. Conditioned on a source word, we use a categorical (i.e., multinomial) distribution to generate the corresponding target word according to the inferred alignments.

Data. We consider three language-pairs: Czech-English, Urdu-English, and Chinese-English. For Czech-English, we use 4.3M bitext tokens for training from the NewsCommentary corpus, WMT10 data set for development, and WMT11 for testing. For Urdu-English, we use the train (2.4M bitext tokens), development, and test sets provided for NIST open MT evaluations 2009. For Chinese-English, we use the BTEC train (0.7M bitext tokens), development, and test sets

[9]It is worth noting that recent advances in neural machine translation (published after our work on word alignment was done), e.g., Bahdanau et al. (2015), obviate the need to do word alignment for machine translation. Other uses for unsupervised word alignment includes extraction of bilingual dictionaries, which are used extensively in chapter 4.

(travel domain).

CRF autoencoder model instantiation. For word alignment, we define the reconstruction model as follows: $p_{\boldsymbol{\theta}}(\hat{\mathbf{x}} \mid \mathbf{y}, \boldsymbol{\phi}) = \prod_{i=1}^{|\mathbf{x}|} \theta_{\hat{x}_i|\phi_{y_i}}$, where $\hat{x}_i$ is the Brown cluster[10] of the word at position i in the target sentence. We use a squared L_2 regularizer for the log-linear parameters $\boldsymbol{\lambda}$ and a symmetric Dirichlet prior for the categorical parameters $\boldsymbol{\theta}$ with the same regularization strength for all language pairs (L_2 strength $= 0.01$, Dirichlet $\alpha = 1.5$). The hyperparameters were optimized to minimize Alignment Error Rate (AER) on a development dataset of French-English bitext. The reconstruction model parameters $\boldsymbol{\theta}$ are initialized with the parameters taken from IBM Model 1 after five EM iterations (Brown et al., 1993). In each block-coordinate ascent iteration, we use L-BFGS to optimize $\boldsymbol{\lambda}$, followed by two EM iterations to optimize $\boldsymbol{\theta}$. Training converges when the relative improvement in objective value falls below 0.03 in one block-coordinate ascent iteration, typically in less than 10 iterations of block-coordinate ascent.

We follow the common practice of training two word alignment models for each dataset, one with English as the target language (forward) and another with English as the source language (reverse). We then use the grow-diag-final-and heuristic (Koehn et al., 2003) to symmetrize alignments before extracting translation rules.

Features. We use the following features: deviation from diagonal word alignment $|\frac{y_i}{|\boldsymbol{\phi}|} - \frac{i}{|\mathbf{x}|}|$; log alignment jump $\log |y_i - y_{i-1}|$; agreement with forward, reverse and symmetrized baseline alignments of mgiza++ and fast_align; Dice measure of the word pair x_i and ϕ_{y_i}; difference in character length between x_i and ϕ_{y_i}; orthograhpic similarity between x_i and ϕ_{y_i}, punctuation token aligned to a non-punctuation token; punctuation token aligned to an identical token; 4-bit prefix of the Brown cluster of x_i conjoined with 4-bit prefix of the Brown cluster of ϕ_{y_i}; forward and reverse probability of the word pair x_i, ϕ_{y_i} with fast_align, as well as their product. We note here that the outputs of other unsupervised aligners are standard (and important!) features in supervised CRF aligners (Blunsom and Cohn, 2006); however, they are nonsensical in a joint model over alignments and sentence pairs.

Baselines. Due to the cost of estimating feature-rich generative models for unsupervised word alignment on the data sizes we are using (e.g., fhmm and dyer-11), we only report the per-sentence computational cost of inference on these baselines. For alignment quality baselines, we report on results from two state-of-the-art baselines that use multinomial parameterizations which support M-step analytic solutions, rather than feature-rich parameterizations: fast_align (Dyer et al., 2013)[11] and model 4 (Brown et al., 1993). fast_align is a recently proposed reparameterization of IBM Model 2 (Brown et al., 1993). model 4, as implemented in mgiza++ (Gao and Vogel, 2008) is the most commonly used word alignment tool in machine translation systems.

Evaluation. When gold standard word alignments are available (i.e., for Czech-English), we use AER (Och and Ney, 2003) to evaluate the alignment predictions of each model. We also perform an extrinsic evaluation of translation quality for all data sets, using case-insensitive BLEU (Papineni et al., 2002) of a hierarchical MT system built using the word alignment predictions of each model.

[10] We use (Liang, 2005) with 80 word classes.

[11] https://github.com/clab/fastalign

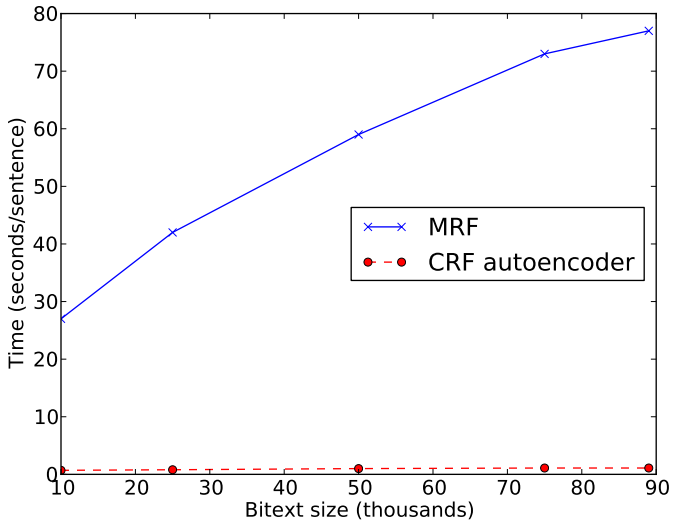


Figure 5.7: Average inference runtime per sentence pair for word alignment in seconds (vertical axis), as a function of the number of *sentences* used for training (horizontal axis).

Bitext word alignment results. First, we consider an intrinsic evaluation on a Czech-English dataset of manual alignments, measuring the alignment error rate (AER; (Och and Ney, 2003)). We also perform an extrinsic evaluation of translation quality for all data sets, using case-insensitive BLEU (Papineni et al., 2002) of a machine translation system (cdec (Dyer et al., 2010)) built using the word alignment predictions of each model.

AER for variants of each model (forward, reverse, and symmetrized) are shown in Table 5.1 (left). Our model signifcantly outperforms both baselines. Bleu scores on the three language pairs are shown in Table 5.1; alignments obtained with our CRF autoencoder model improve translation quality of the Czech-English and Urdu-English translation systems, but not of Chinese-English. This is unsurprising, given that Chinese orthography does not use letters, so that source-language spelling and morphology features our model incorporates introduce only noise here. Better feature engineering, or more data, is called for.

We have argued that the feature-rich CRF autoencoder will scale better than its feature-rich alternatives. Fig. 5.7 shows the average per-sentence inference runtime for the CRF autoencoder compared to exact inference in the undirected joint model of Dyer et al. (2011) with a similar feature set, as a function of the number of sentences in the corpus. For CRF autoencoders, the average inference runtime grows slightly due to the increased number of parameters, while for Dyer et al. (2011) it grows substantially with the vocabulary size.[12,13]

[12]We only compare runtime, instead of alignment quality, because retraining the MRF model with exact inference was too expensive.

[13]We did not observe a similar pattern when comparing the runtimes of the CRF autoencoder and the feature HMM of Berg-Kirkpatrick et al. (2010) who informed us in personal communication of a computational trick to avoid expensive log operations in the forward-backward algorithm which sped up their training by an order of magnitude. Nevertheless, the asymptotic runtime analysis of inference in the CRF autoencoder model is more favorable, as shown in §5.2.2.

direction	fast_align	model 4	crf-auto
forward	27.7	31.5	**27.5**
reverse	25.9	24.1	**21.1**
symmetric	25.2	22.2	**19.5**

pair	fast_align	model 4	crf-auto
cs-en	$15.2_{\pm 0.3}$	$15.3_{\pm 0.1}$	$\mathbf{15.5}_{\pm 0.1}$
ur-en	$20.0_{\pm 0.6}$	$20.1_{\pm 0.6}$	$\mathbf{20.8}_{\pm 0.5}$
zh-en	$\mathbf{56.9}_{\pm 1.6}$	$56.7_{\pm 1.6}$	$56.1_{\pm 1.7}$

Table 5.1:
Left: AER results (%) for Czech-English word alignment. Lower values are better.
Right: BLEU translation quality scores (%) for Czech-English, Urdu-English and Chinese-English. Higher values are better.

Stochastic optimization of $\boldsymbol{\lambda}$. We experiment with stochastic gradient descent (SGD) using an approximation of the gradient based on a few examples only. One epoch (i.e., full pass over the training set) of SGD constitutes *many updates* of λ, and incurs approximately the same runtime cost as *one update* of L-BFGS. In the following experiments on word alignments, we use a training set of $N = 134296$ parallel Finnish-English sentence pairs, selected such that $|\mathbf{s}| \leq 5 \wedge |\mathbf{t}| \leq 5$. The t-th SGD update takes the form:

$$\boldsymbol{\lambda}^{(t)} = \boldsymbol{\lambda}^{(t-1)} - \gamma_t \nabla_{\boldsymbol{\lambda}} - \log p_{\boldsymbol{\lambda}^{(t-1)}}(\hat{\mathbf{x}} \mid \mathbf{x}, \phi) \qquad \text{where } t = 1, \ldots, \infty \tag{5.9}$$

We explore different strategies for updating the learning rate (step size) γ_t at the t-th iteration:

- Fixed learning rate: $\gamma_t = \gamma, \forall t$
- Diminishing learning rate with geometric decay (Bottou, 2012): $\gamma_t = \gamma_0/(1 + \gamma_0 \eta t)$
- Diminishing learning rate with exponentially variable decay rate: $\gamma_t = \gamma_{t-1}/(1 + \gamma_{t-1} \eta t)$
- Diminishing learning rate with exponentially constant decay rate: $\gamma_t = \gamma_{t-1}/(1 + \eta)$
- Epoch-fixed learning rate: $\gamma_{\{t:(k-1)*N \leq t < k*N\}} = \frac{1}{k}$ where $k = 1, 2, \ldots$ is the epoch index and N is the epoch size.

where γ_0 is the initial learning rate, and $\eta > 0$ is a decay hyper-parameter (larger values of η result in faster decay).

We quantify progress by reporting the attained value of the objective function (to be minimized) after each epoch, and compare to batch L-BFGS as our baseline. For practical purposes, we are primarily interested in the progress made by each optimization method for one or two epochs, but we plot the objective values after $k = 1, \ldots, 32$ epochs to also see how each method converges. All optimization methods are initialized with the same value of $\lambda^0 = 0$.

Fig. 5.8 compares L-BFGS to SGD with a constant learning rate (left) and with diminishing learning rates (right). L-BFGS converges to a better value of the objective, but since we cannot afford to run many epochs in the inner loop of the block coordinate descent algorithm. Therefore, we focus on the first few epochs and find that all variants of SGD achieve much better results. In the first few epochs, the geometric decay strategy of Bottou (2012) with $\eta = 0.001$ performs best (right, pink), closely followed by fixed learning rate with $\gamma = 0.03$ (left, blue). Fig. 5.9 (left) contrasts L-BFGS to SGD with the epoch-fixed learning strategy. Despite having no hyper-parameters, this update strategy for epoch-fixed learning rate appears to be quite effective. In addition to having a significant head start compared to L-BFGS, it also converges to approximately the same objective value after 32 training epochs.

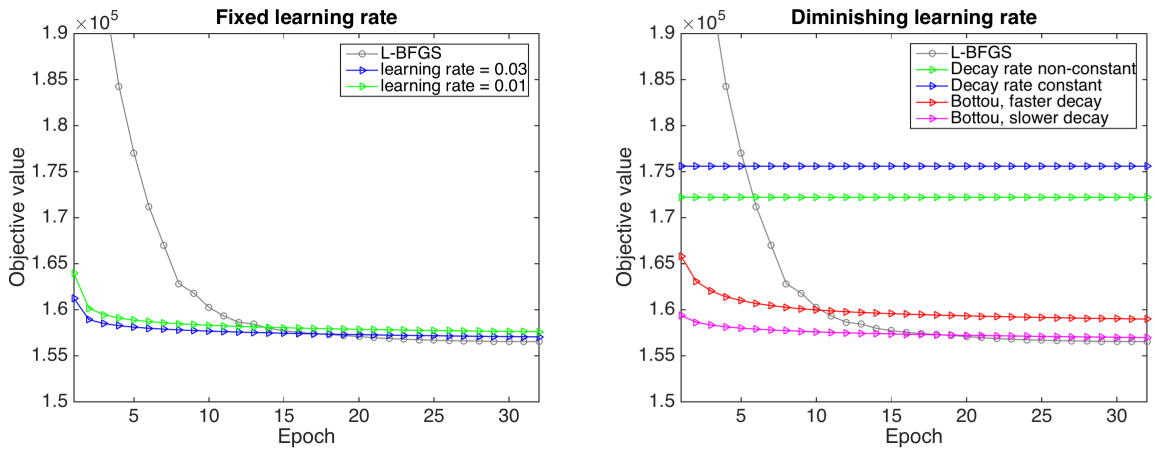


Figure 5.8:
Left: L-BFGS vs. SGD with a constant learning rate.
Right: L-BFGS vs. SGD with diminishing γ_t.

Fig. 5.9 (left) contrasts two ways of choosing the index of examples $(\hat{\mathbf{x}}, \mathbf{x}, \phi)$ from an unlabeled training set U to use for the t-th SGD update. In the cyclic order, we choose $i_t = 1, 2, \ldots, N, 1, 2, \ldots$. In the randomized order (Bertsekas, 2011), every N consecutive indices $\{i_{(k-1)*N}, \ldots, i_{k*N}\}$ is a uniform random permutation of $\{1, \ldots, N\}$, for $k = 1, 2, \ldots$. As shown in Fig. 5.9 (left), our experiments showed no noticeable difference between the cyclic vs. randomized order.

Instead of the SGD update in Eq. 5.9, the averaged stochastic gradient descent (ASGD) algorithm updates the parameters $\boldsymbol{\lambda}$ as follows:

$$\hat{\lambda} = \lambda^{(t-1)} - \gamma_t \nabla_\lambda - \log p_{\boldsymbol{\lambda}^{(t-1)}}(\hat{\boldsymbol{x}} \mid \boldsymbol{x}, \phi)$$
$$\lambda^{(t)} = \frac{1}{t}(\hat{\lambda} + \sum_{j=1}^{t-1} \lambda^{(j)}) \qquad \text{where } t = 1, \ldots, \infty \tag{5.10}$$

ASGD has an optimal (local) convergence rate of $O(\frac{1}{t})$ (Bottou, 2012), assuming the learning rates decrease slower than t^{-1}. However, our empirical results suggest that there is no practical difference between the objective values obtained with SGD vs. ASGD.

Finally, we experiment with performing SGD updates in parallel on multiple processors in the same machine. All processors share the same memory to store parameter values using the Message Passing Interface (MPI) standards. We found that the reduction in the criterion value is indistinguishable from using a single-core (see Fig. 5.9).

Stochastic optimization of $\boldsymbol{\theta}$. The main idea behind the online EM algorithm is to interpolate between the sufficient statistics inferred from the current example (with weight η_k) and the accumulated sufficient statistics inferred from previous iterations of the algorithm (with weight $1 - \eta_k$). Following Liang and Klein (2009), we use the following formula to control the interpolation parameter η_k in the kth iteration of online EM: $\eta_k = (k + 2)^{-\alpha}$. It is instructive to consider extreme values of α and how they affect the interpolation: $\alpha = 0$ puts all the weight

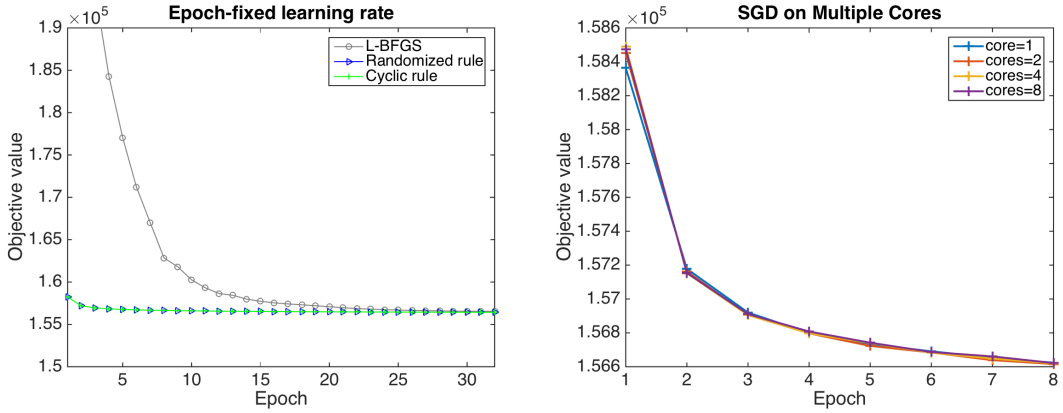


Figure 5.9:
Left: L-BFGS vs. SGD with cyclic vs. randomized order (and with epoch-fixed γ).
Right: Asynchronous SGD updates with 1, 2, 4, and 8 processors.

on the current example, while $\alpha = \infty$ puts all the weight on previous examples. It is therefore appropriate to regard α as an (inverse) learning rate, or rather a "stickiness" rate. Our empirical results in Fig. 5.10 (left) confirm the significant effect of α. However, none of the learning rates we tried resulted in faster convergence than batch EM.

Due to the projection step (M-step) in expectation maximization with multinomial-constrained parameters, the stochastic updates affects all parameters, including those with zero expected counts in the new example. As a result, it is critical for online EM to use more than one example (i.e., mini-batches) to compute the new sufficient statistics m'_i. In Fig. 5.10 (right), we experiment with mini-batches of size $10^2, 10^3, 10^4$. At the limit, when the mini-batch size is ∞, we recover batch EM. None of the mini-batch sizes we tried resulted in faster convergence than batch EM.

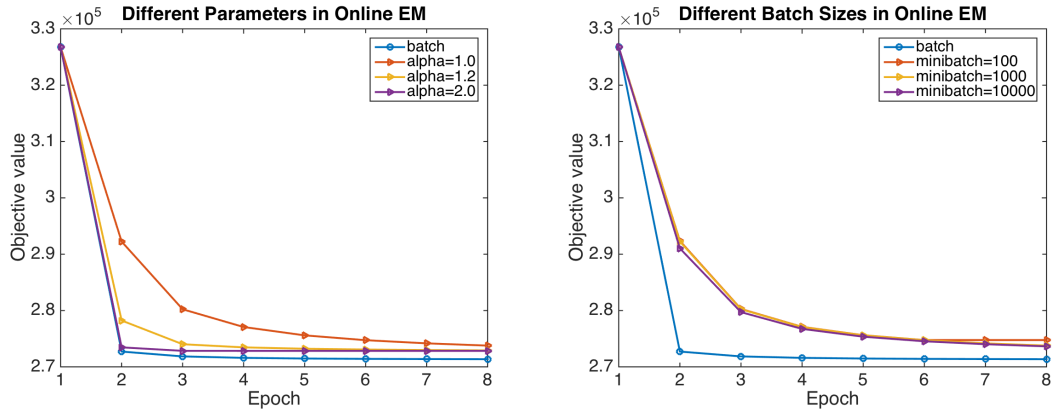


Figure 5.10:
Left: Batch vs. online EM with different values of α (stickiness parameter).
Right: Batch EM vs. online EM with mini-batch sizes of $10^2, 10^3, 10^4$.

5.3.3 Token-level Language Identification

Code switching occurs when a multilingual speaker uses more than one language in the same conversation or discourse (Sankoff, 1998). It is pervasive in social media due to its informal nature (Lui and Baldwin, 2014). Automatic identification of the points at which code switching occurs is important for two reasons: (1) to help sociolinguists analyze the frequency, circumstances and motivations related to code switching (Gumperz, 1982), and (2) to inform other NLP models which may be able to process code-switched input such as our language-universal parser in chapter 3.

Data. The first workshop on computational approaches to code switching in EMNLP 2014 organized a shared task (Solorio et al., 2014) on identifying code switching. The shared task training data consists of code-switched tweets with token-level annotations. The data is organized in four language pairs: English–Spanish (En-Es), English–Nepali (En-Ne), Mandarin–English (Zh-En) and Modern Standard Arabic–Arabic dialects (MSA-ARZ). For each tweet in the data set, the user ID, tweet ID, and a list of tokens' start offset and end offset are provided. Each token is annotated with one of the following labels: `lang1`, `lang2`, `ne` (i.e., named entities), `mixed` (i.e., mixed parts of lang1 and lang2), `ambiguous` (i.e., cannot be identified given context), and `other`. Since the official test sets were not provided, we use a subset of the data provided for training as a test set. Statistics of our train/test data splits are given in Table 5.2.

lang. pair	split	tweets	tokens	users
En–Ne	all	9,993	146,053	18
	train	7,504	109,040	12
	test	2,489	37,013	6
En–Es	all	11,400	140,738	9
	train	7,399	101,451	6
	test	4,001	39,287	3
Zh–En	all	994	17,408	995
	train	662	11,677	663
	test	332	5,731	332
MSA–ARZ	all	5,862	119,775	7
	train	4,800	95,352	6
	test	1,062	24,423	1

Table 5.2: Total number of tweets, tokens, and Twitter user IDs for each language pair. For each language pair, the first line represents all data provided to shared task participants. The second and third lines represent our train/test data split for the experiments reported in this chapter. Since Twitter users are allowed to delete their tweets, the number of tweets and tokens reported in the third and fourth columns may be less than the number of tweets and tokens originally annotated by the shared task organizers.

Baseline system. We model token-level language ID as a sequence of labels using a linear-chain CRF (Lafferty et al., 2001) which models the conditional probability of a label sequence

$\mathbf{y}$ given a token sequence $\mathbf{x}$. We use L-BFGS to learn the feature weights $\boldsymbol{\lambda}$, maximizing the L_2-regularized log-likelihood of labeled examples $\mathcal{L}$. We use the following features:

- character n-grams (loweredcased tri- and quad-grams)
- prefixes and suffixes of lengths 1, 2, 3 and 4
- unicode page of the first character[14]
- case (first-character-uppercased vs. all-characters-uppercased vs. all-characters-alphanumeric)
- tweet-level language ID predictions from two off-the-shelf language identifiers: `cld2`[15] and `ldig`[16]

Semi-supervised learning with CRF autoencoders. While constructing a code-switched data set that is annotated at the token level requires remarkable manual effort, collecting raw tweets is easy and fast. We use CRF autoencoders to leverage both labeled and unlabeled data in a unified framework, as described in §5.2.3. The encoding model is identical to the CRF baseline. The reconstruction model uses categorical distributions to independently generate tokens conditional on the corresponding labels. We again use block coordinate descent with two subroutines: EM to optimize the parameters of the reconstruction model, and L-BFGS to optimize the parameters of the encoding model.

Word embeddings. We use `word2vec` (Mikolov et al., 2013a) to train 100-dimensional word embeddings from a large Twitter corpus of about 20 million tweets extracted from the live stream, in multiple languages. We define an additional real-valued feature function in the CRF autoencoder model for each of the 100 dimensions, conjoined with the label y_i. A binary feature indicating the absence of word embeddings is fired for out-of-vocabulary words (i.e., words for which we do not have word embeddings). The token-level coverage of the word embeddings for each of the languages or dialects used in the training data is reported in Table 5.3.

Word list features While some words are ambiguous, many words frequently occur in only one of the two languages being considered. An easy way to identify the label of such unambiguous words is to check whether they belong to the vocabulary of either language. Moreover, named entity recognizers typically rely on gazetteers of named entities to improve their performance. More generally, we use features based on word lists we construct for this task. Using K word lists $\{l_1, \ldots, l_K\}$, when a token x_i is labeled with y_i, we fire a binary feature that conjoins $\langle y_i, \delta(x_i \in l_1), \ldots, \delta(x_i \in l_K)\rangle$, where δ is an indicator boolean function. We use the following word lists:

- Hindi and Nepali Wikipedia article titles
- multilingual named entities from the JRC dataset[17] and CoNLL 2003 shared task
- word types in monolingual corpora in MSA, ARZ, En and Es.
- set difference between the following pairs of word lists: MSA-ARZ, ARZ-MSA, En-Es, Es-En.

[14] `http://www.unicode.org/charts/`
[15] `https://code.google.com/p/cld2/`
[16] `https://github.com/shuyo/ldig`
[17] `http://datahub.io/dataset/jrc-names`

language	embeddings coverage	word lists coverage
ARZ	30.7	68.8
En	73.5	55.7
MSA	26.6	76.8
Ne	14.5	77.0
Es	62.9	78.0
Zh	16.0	0.7

Table 5.3: The type-level coverage percentages of annotated data according to word embeddings (second column) and according to word lists (third column), per language.

Instead of using the Devanagari script the Nepali and Hindi words in available code-switched dataset are romanized. This renders some of our word lists based on the Devanagari script useless. Therefore, we transliterate the Hindi and Nepali named entities lists using the IAST scheme,[18] and drop all accent marks on the characters.

Setup. We use two sets of unlabeled data: (1) $\mathcal{U}_{\text{test}}$ which only includes the test set,[19] and (2) $\mathcal{U}_{\text{all}}$ which includes the test set as well as *all* available tweets by the set of users who contributed any tweets in $\mathcal{L}$. We use the supervised objective to initialize the parameters of the encoding model in the CRF autoencoder model. The categorical distributions of the reconstruction model are initialized with discrete uniforms. We set the weight of the labeled data log-likelihood $c_{\text{labeled}} = 0.5$, the weight of the unlabeled data log-likelihood $c_{\text{unlabeled}} = 0.5$, the L_2 regularization strength $c_{L_2} = 0.3$, the concentration parameter of the Dirichlet prior $\alpha = 0.1$, the number of L-BFGS iterations $c_{\text{LBFGS}} = 4$, and the number of EM iterations $c_{EM} = 4$.[20] We stop training after 50 block coordinate descent iterations.

Results. The CRF baseline results are reported in the first line in Table 5.4. For three language pairs, the overall token-level accuracy ranges between 94.6% and 95.2%. In the fourth language pair, MSA-ARZ, the baseline accuracy is 80.5% which indicates the relative difficulty of this task. The confusion matrix between the labels lang1 (i.e., MSA) and lang2 (i.e., ARZ) for this language pair is given by Table 5.5, which illustrates that most errors are due to classifying ARZ tokens as MSA.

The second and third lines in Table 5.4 show the results when we use CRF autoencoders with the unlabeled test set ($\mathcal{U}_{\text{test}}$), and with all unlabeled tweets ($\mathcal{U}_{\text{all}}$), respectively. While semi-supervised learning did not hurt accuracy on any of the languages, it only resulted in a tiny increase in accuracy for the Arabic dialects task.

The fourth line in Table 5.4 extends the CRF autoencoder model (third line) by adding unsupervised word embedding features. This results in an improvement of 0.6% for MSA-ARZ, 0.5% for En-Es, 0.1% for En-Ne and Zh-En. We note that the coverage of word embeddings in

[18] `http://en.wikipedia.org/wiki/International_Alphabet_of_Sanskrit_Transliteration`

[19] This is potentially useful when the test set belongs to a different domain than the labeled examples.

[20] Hyper-parameters c_{L_2} and α were tuned using cross-validation. The remaining hyper-parameters were not tuned.

configuration	En–Ne	MSA–ARZ	En–Es	Zh–En
CRF	95.2	80.5	94.6	94.9
+$\mathcal{U}_{\text{test}}$	95.2	80.6	94.6	94.9
+$\mathcal{U}_{\text{all}}$	95.2	80.7	94.6	94.9
+emb.	95.3	81.3	95.1	95.0
+lists	97.0	81.2	96.7	95.3

Table 5.4: Token level accuracy (%) results for each of the four language pairs.

label	predicted MSA	predicted ARZ
true MSA	93.9%	5.3%
true ARZ	32.1%	65.2%

Table 5.5: Confusion between MSA and ARZ in the Baseline configuration.

MSA, ARZ and Es is better than the coverage in Ne and Zh (see Table 5.3). We conclude that further improvements on En-Ne and Zh-En may be expected if they are better represented in the corpus used to learn word embeddings.

The fifth line builds on the fourth line by adding word list features. This results in an improvement of 1.7% in En-Ne, 1.6% in En-Es, 0.4% in Zh-En. While the overall accuracy degrades by 0.1% in MSA-ARZ, closer inspection in 5.6 reveals that it improves the F-Measure of the named entities at the expense of both MSA (lang1) and ARZ (lang2).

Although the reported semi-supervised results did not improve on the CRF baseline, more work needs to be done in order to make a final conclusion. For instance, we did not tune the hyperparameters c_{labeled} and $c_{\text{unlabeled}}$ which control the contribution of labeled vs. unlabeled examples to the training objective. It may also be important to use a less sparse reconstruction model, e.g., generate word embeddings using a multivariate Gaussian model. We may get better semi-supervised results with a smaller number of labeled examples (see Table 5.7 for the number of labeled and unlabeled examples used in our experiments). The reconstruction model with in our experiments is very sparse.

5.4 Open Questions

The first question that remains to be answered is whether CRF autoencoders can be effectively used for semi-supervised learning. In §5.2.3, we discussed how to modify the training objective of CRF autoencoders to use both labeled and unlabeled examples. We used a premature implementation of this model as our submission in the code-switching shared task at EMNLP

Config	`lang1`	`lang2`	`ne`
+lists	84.1%	76.5%	73.7%
-lists	84.2%	77.1%	71.5%

Table 5.6: F-Measures of two Arabic configurations. `lang1` is MSA. `lang2` is ARZ.

lang. pair	$\lvert\mathcal{U}_{\text{test}}\rvert$	$\lvert\mathcal{U}_{\text{all}}\rvert$	$\lvert\mathcal{L}\rvert$
En–Ne	2489	6230	7504
MSA–ARZ	1062	2520	4800
Zh–En	332	332	663
En–Es	4001	7177	7399

Table 5.7: Number of tweets in $\mathcal{L}$, $\mathcal{U}_{\text{test}}$ and $\mathcal{U}_{\text{all}}$ used for semi-supervised learning of CRF autoencoders models.

2014 (Lin et al., 2015), but more work needs to be done to assess the utility of CRF autoencoder models in semi-supervised learning. For example, we did not tune the hyperparameters which control the contribution of labeled vs. unlabeled examples to the training objective. We used a naïve parameterization of the reconstruction model (categorical distribution with one parameter for each surface form). Another proposal for using CRF autoencoder in semi-supervised learning is to define a prior distribution over the CRF autoencoder model parameters, using the labeled examples to determine the mean and variance of the prior distribution.

With the widespread use of neural networks, another question presents itself: is it beneficial to implement a neural network realization of CRF autoencoders? While it is possible to simulate the CRF autoencoder models we discussed earlier with a neural network architecture, including exact inference and marginalizing out the latent structure variables, it is not clear whether the added complexity is necessary. Also, is it beneficial to jointly learn the word embeddings instead of pretraining them like we did in this chapter?

Another question is how to determine whether CRF autoencoder models are a good fit for a given structured prediction problem. We identified a number of characteristics to help answer this question, e.g., no or few labeled examples are available, the number of possible values of the latent structure are significantly smaller than that of the observed structure, conditional random fields have successfully been used to model the problem in the fully-supervised setting. However, it is also not clear whether CRF autoencoders are a good fit for real-valued latent variables, latent structures with loops, or problems outside NLP.

5.5 Related Work

This work relates to several strands of work in unsupervised learning. Unsupervised learning with flexible feature representations has long been studied, and there are broadly two types of models that support this. Both are fully generative models that define joint distributions over $\mathbf{x}$ and $\mathbf{y}$. We will refer to these as the "undirected" and "directed" alternatives. We discuss these next and then turn to less closely related methods.

Undirected models. The undirected alternative uses an undirected model to encode the distribution through local potential functions parameterized using features. Such models "normalize globally," requiring during training the calculation of a partition function summing over all values

of both (in our notation):

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z(\boldsymbol{\theta})} \exp \boldsymbol{\lambda}^\top \bar{\mathbf{g}}(\mathbf{x}, \mathbf{y}) \tag{5.11}$$
$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{x} \in \mathcal{X}^*} \sum_{\mathbf{y} \in \mathcal{Y}^{|\mathbf{x}|}} \exp \boldsymbol{\lambda}^\top \bar{\mathbf{g}}(\mathbf{x}, \mathbf{y})$$

where $\bar{\mathbf{g}}$ collects all the local factorization by cliques of the graph, for clarity. The key difficulty is in the summation over all possible observations. Approximations have been proposed, including contrastive estimation, which sums over subsets of $\mathcal{X}^*$ (Smith and Eisner, 2005; Vickrey et al., 2010) (applied variously to POS learning by Haghighi and Klein (2006) and word alignment by Dyer et al. (2011)) and noise contrastive estimation (Mnih and Teh, 2012).

Directed models. The directed alternative avoids the global partition function by factorizing the joint distribution in terms of locally normalized conditional probabilities, which are parameterized in terms of features. For unsupervised sequence labeling, the model was called a "feature HMM" by Berg-Kirkpatrick et al. (2010). The local emission probabilities $p(x_i \mid y_i)$ in a first-order HMM for POS tagging are reparameterized as follows (again, using notation close to ours):

$$p_{\boldsymbol{\lambda}}(x_i \mid y_i) = \frac{\exp \boldsymbol{\lambda}^\top \mathbf{g}(x_i, y_i)}{\sum_{x \in \mathcal{X}} \exp \boldsymbol{\lambda}^\top \mathbf{g}(x, y_i)} \tag{5.12}$$

The features relating hidden to observed variables must be local within the factors implied by the directed graph. We show below that this locality restriction excludes features that are useful (§5.3.1).

Put in these terms, our proposed autoencoding model is a hybrid directed-undirected model.

Asymptotic runtime complexity of inference. The models just described cannot condition on arbitrary amounts of $\mathbf{x}$ without increasing inference costs. Despite the strong independence assumptions of those models, the computational complexity of inference required for learning with CRF autoencoders is better, as will be shown here.

Consider learning the parameters of an undirected model by maximizing likelihood of the observed data. Computing the gradient for a training instance $\mathbf{x}$ requires time

$$\mathrm{O}\left(|\boldsymbol{\lambda}| + |\mathcal{U}| \times |\mathbf{x}| \times |\mathcal{Y}| \times (|\mathcal{Y}| \times |F_{y_{i-1}, y_i}| + |\mathcal{X}| \times |F_{x_i, y_i}|)\right),$$

where $F_{x_i - y_i}$ are the emission-like features used in an arbitrary assignment of x_i and y_i. When the multiplicative factor $|\mathcal{X}|$ is large, inference is slow compared to CRF autoencoders.

In directed models (Berg-Kirkpatrick et al., 2010), each iteration requires time

$$\mathrm{O}\left(|\boldsymbol{\lambda}| + |\mathcal{U}| \times |\mathbf{x}| \times |\mathcal{Y}| \times (|\mathcal{Y}| \times |F_{y_{i-1}, y_i}| + |F_{x_i, y_i}|) + |\boldsymbol{\theta}'| \times \max(|F_{y_{i-1}, y_i}|, |F_{\mathcal{X}, y_i}|)\right),$$

where F_{x_i, y_i} are the active emission features used in an arbitrary assignment of x_i and y_i, $F_{\mathcal{X}, \dagger)}$ is the union of all emission features used with an arbitrary assignment of y_i, and $\boldsymbol{\theta}'$ are the local emission and transition probabilities. When $|\mathcal{X}|$ is large, the last term $|\boldsymbol{\theta}'| \times \max(|F_{y_{i-1}, y_i}|, |F_{\mathcal{X}, y_i}|$ dominates runtime.

In contrast, the asymptotic runtime complexity of one iteration of block coordinate descent in the linear-chian CRF autoencoder model in Fig. 5.2 (right), is:

$$\mathrm{O}\left(|\boldsymbol{\theta}| + |\boldsymbol{\lambda}| + |\mathcal{U}| \times |\mathbf{x}|_{max} \times |\mathcal{Y}|_{max} \times (|\mathcal{Y}|_{max} \times |F_{y_{i-1},y_i}| + |F_{\mathbf{x},y_i}|)\right) \tag{5.13}$$

where F_{y_{i-1},y_i} are the active "label bigram" features used in $\langle y_{i-1}, y_i \rangle$ factors, $F_{\mathbf{x},y_i}$ are the active emission-like features used in $\langle \mathbf{x}, y_i \rangle$ factors. $|\mathbf{x}|_{max}$ is the maximum length of an observation sequence. $|\mathcal{Y}|_{max}$ is the maximum cardinality[21] of the set of possible assignments of y_i. Compared to directed and undirected models discussed earlier which only allow feature functions with domain $\langle y_{i-1}, y_i, x_i \rangle$, CRF autoencoders enable feature functions with domain $\langle \mathbf{x}, y_i \rangle$ and also provides for a better asymptotic runtime complexity.

Autoencoders and other "predict self" methods. Our framework borrows its general structure, Fig. 5.2 (left), as well as its name, from *neural network* autoencoders. The goal of neural autoencoders is to learn feature representations that improve generalization in otherwise supervised learning problems (Vincent et al., 2008; Collobert and Weston, 2008; Socher et al., 2010).

Daumé III (2009) introduced a reduction of an unsupervised problem instance to a series of single-variable supervised classifications; the first series of these construct a latent structure $\mathbf{y}$ given the entire $\mathbf{x}$, then the second series reconstruct the input again using only $\mathbf{y}$. The approach can make use of any supervised learner; if feature-based probabilistic models were used, a $|\mathcal{X}|$ summation (akin to Eq. 5.12) would be required. On unsupervised POS induction, this approach performed on par with the undirected model of Smith and Eisner (2005).

Minka (2005) proposed cascading a generative model and a discriminative model, where *class labels* (to be predicted at test time) are *marginalized out* in the generative part first, and then (re)generated in the discriminative part. In CRF autoencoders, *observations* (available at test time) are *conditioned on* in the discriminative part first, and then (re)generated in the generative part.

Posterior regularization. Introduced by Ganchev et al. (2010), posterior regularization imposes constraints on the learned model's posterior, i.e., $p(\mathbf{y} \mid \mathbf{x})$; a similar idea was proposed independently Bellare et al. (2009). For example, in POS induction, every sentence might be expected to contain at least one verb. This is imposed as a soft constraint, i.e., a feature whose expected value under the model's posterior is fixed to a predefined value. Such expectation constraints are specified directly by the domain-aware model designer. The approach was applied to unsupervised POS induction, word alignment, and parsing. Though they applied posterior regularization to directed generative models that were not featurized, the idea is orthogonal to the model family and could be applied as well with a CRF autoencoder.

5.6 Summary

We have presented a model for unsupervised learning of latent structures. The technique allows features with global scope in observation variables with favorable asymptotic inference runtime. We achieve this by embedding a CRF as the encoding model in the input layer of an autoencoder,

[21] In POS induction, $|\mathcal{Y}|$ is a constant, the number of syntactic classes which we configure to 12 in our experiments. In word alignment, $|\mathcal{Y}|$ is the size of the source sentence plus one, therefore $|\mathcal{Y}|_{max}$ is the maximum length of a source sentence in the bitext corpus.

and using a local model to reconstruct the observations in the output layer. A key advantage of the proposed model is scalability, since inference is no more expensive than a supervised CRF model. We applied the model to POS induction, bitext word alignment, obtaining competitive results on both tasks. We also discussed how to use the model for semi-supervised learning, along with preliminary results in this setup for token-level language identification.

In POS induction, we found that multivariate Gaussian distributions (over the vector space of pretrained word embeddings) provide a better alternative to conventional multinomial emission distributions in generative models as well as CRF autoencoders. In word alignment, we found stochastic optimization of the encoding model parameters to be more effective than using L-BFGS, and studied the effects of the learning rate update strategy, parallelization, mini-batch size and averaging. In token-level language identification, we found that using unlabeled examples did not help, but adding word list features and word embedding features in the encoding part of the CRF autoencoder model improves the performance.

Chapter 6

Conclusion

6.1 Contributions

In this thesis, we advocate for a novel language-universal approach to multilingual NLP in which one statistical model trained on cross-lingually consistent annotations in multiple languages is used to process natural language input in multiple languages. The proposed approach addresses several practical difficulties in multilingual NLP such as maintaining a large number of monolingual models, and the impracticality of using models specifically designed for low-resource scenarios. We empirically show the merits of this approach by developing a language-universal dependency parser. Due to the importance of lexical features in many NLP problems, we propose novel methods for estimating multilingual representations of lexical items in multiple languages with limited resources. We also propose the CRF autoencoder model for unsupervised learning with features, and use it to infer word alignments in parallel corpora.

The detailed list of contributions is:

- We introduced the language-universal approach for training multilingual NLP models.
- We developed MALOPA, a dependency parser that exemplifies this approach, and made the code available at `https://github.com/clab/language-universal-parser`, with a web-based demo at `http://128.2.220.95/multilingual/parse/`.
- In high-resource scenarios, we showed that MALOPA outperforms strong monolingually-trained baselines on average and in five out of seven target languages.
- In low-resource scenarios, we showed that MALOPA consistently outperforms the previous state-of-the-art in training dependency parsers with cross-lingual supervision and a small treebank in the target language (Duong et al., 2015).
- In low-resource scenarios, we also showed that MALOPA outperforms the previous state-of-the-art in training dependency parsers with cross-lingual supervision and no treebank in the target language (Guo et al., 2016).
- We studied the effects of predicting language ID and predicting POS tags on dependency parsing with MALOPA. On average, predicting language ID and POS tags hurt the labeled attachment accuracy by 0.8 and 5.1 absolute points, respectively.
- We showed promising results for MALOPA on a synthetic treebank with code switching.
- We developed the multiCluster and multiCCA methods for estimating multilingual word embeddings. Instead of parallel corpora, both methods use bilingual dictionaries which are available for more languages than are parallel corpora. We used both methods to train multilingual word embeddings for fifty-nine languages.
- We showed that dictionary-based methods outperform previous methods for estimating multilingual word embeddings on multilingual word similarity, monolingual QVEC, multilingual QVEC, dependency parsing and document classification.
- We developed a web-based evaluation portal for multilingual word embeddings at `http://128.2.220.95/multilingual/`, which will substantially reduce the overhead of trying new methods for estimating and evaluating multilingual word embeddings.
- We introduced the CRF autoencoder framework for unsupervised learning of structured predictors.

- We implemented a CRF autoencoder model for part-of-speech induction, and showed that it outperform HMMs (with and without features).
- Using a Gaussian reconstruction model, we showed that pretrained word embeddings can significantly improve part-of-speech induction in CRF autoencoders (also in HMMs), and studied the effects of the vector size and context window size.
- We implemented a CRF autoencoder model for word alignment, and showed that it outperforms two strong baselines: fast_align (Dyer et al., 2013) and mgiza++ (Gao and Vogel, 2008).
- We compared different optimization methods for training the CRF autoencoder model, and found some variants of stochastic gradient descent to converge much faster than L-BFGS.
- We make our implementations of the CRF autoencoder for POS induction, word alignment and token-level language identification available at `https://github.com/ldmt-muri/alignment-with-openfst`.

6.2 Future Directions

The work presented in this thesis can be extended in many directions.

(Even) better models for low-resource languages. We only used unlabeled exampled in low-resource languages to estimate distributional word representations. Other semi-supervised learning methods can potentially be used to learn from labeled examples in all high-resource languages as well as unlabeled examples in low-resource languages. In particular, it would be interesting to train a language-universal CRF autoencoder model on both labeled and unlabeled examples.

More distant target languages. While our proposed language-universal parser is a viable solution for parsing any target language with a bilingual dictionary, the parsing performance suffers in target languages which are very typologically different from all source languages used to train the parser. To address this problem, recent annotation projection methods such as Tiedemann (2014) or Rasooli and Collins (2015) can be used to expose the parser to more diverse syntactic patterns during training.

Better use of linguistic typology. We show that providing the language-universal parser with information about the input language improves the parsing performance, which is expected. However, we found that the model benefits more from language ID than from typological properties. How can we help the parser identify the typological similarities and distinctions between different languages, especially for languages not observed in training? It would be interesting to see how different subsets of typological properties impact the results. It is also possible that more complex neural architectures such as Tsvetkov et al. (2015b) are needed to make use of the typological properties.

Better evaluation on code switched text. We found the language-universal dependency parser to be reliable when the input text has synthetic code-switching. It would be interesting to annotate a small treebank with naturally-occurring code switching for a more accurate evaluation.

End-to-end universal analyzers of natural language. The language-universal approach promises a simpler pipeline for processing multilingual input by replacing an array of language-specific dependency parsing models with one language-universal model. However, dependency parsing is only one of many components in a typical NLP pipeline. Given the superior capacity of deep neural networks for modeling complex functions, as MALOPA and many other models demonstrate, is it feasible to replace the entire NLP pipeline with a neural network which reads the sequence of tokens in any language at the input layer and produces the desired output of the NLP pipeline (e.g., the answer to a question) at the output layer? We can still use multi-task learning to make use of intermediate linguistic abstractions (e.g., tokenization, part-of-speech tags, named entities, syntactic trees, coreferents and semantic parses), when available, to reduce the complexity of the problem. Collobert et al. (2011) pursued this approach and showed excellent results on multiple NLP tasks in English. It would be interesting to combine the work of Collobert et al. (2011) with the ideas discussed in this thesis to develop *end-to-end* universal analyzers of natural language.

Appendix A

Adding New Evaluation Tasks to the Evaluation Portal

The evaluation portal has been designed such that it is easy for other researchers to add more evaluation tasks. We detail the steps for adding a new evaluation task to the portal as follows:

1. Clone the Git repository. The backend of the evaluation portal is hosted as a public GitHub repository.[1] No special permissions are needed to clone the repository.

```
% requirements: unix, git, bash, 50GB of disk space
git clone \
  git@github.com:wammar/multilingual-embeddings-eval-portal.git
cd multilingual-embeddings-eval-portal
% remember the root directory for the repository on this machine
export $EVAL_ROOT=`pwd`
```

2. Copy evaluation dataset files. Every evaluation task (e.g., word similarity) is associated with one more evaluation datasets. The directory `$EVAL_ROOT/eval-data/` has one subdirectory per each evaluation task (e.g., `$EVAL_ROOT/eval-data/wordsim/`) Pick a descriptive but short name for the new evaluation task (e.g., `ner`), then create a subdirectory for it. For example:

```
% create a directory for the evaluation task 'ner'
mkdir $EVAL_ROOT/eval-data/ner
```

Create a subdirectory for each evaluation dataset. Use the suffix *dev* or *test* in the subdirectory name to designate whether the dataset is intended for development or testing. It is also recommended that the subdirectory refers to the languages covered by this evaluation task. For example, the subdirectory `ud1.1-bg+cs+da+de+el+en+es+fi+fr+hu+it+sv-dev` includes the development sections of the universal dependency treebanks v1.1 in Bulgarian, Czech, Danish, German, Greek, English, Spanish, Finnish, French, Hungarian, Italian and Swedish. We recommend adding evaluation datasets in pairs for development and testing to encourage sound experimentation practices. For example:

[1] `https://github.com/wammar/multilingual-embeddings-eval-portal/`

```
mkdir $EVAL_ROOT/eval-data/ner/conll03-en+de-dev
mkdir $EVAL_ROOT/eval-data/ner/conll03-en+de-test
```

Subdirectories referring to different datasets of the same task (e.g., `conll03-en+de-dev` and `conll03-en+de-test`) must have the same file structure. For example, all word similarity evaluation datasets under `$EVAL_ROOT/eval-data/wordsim/` must contain a file with the name `annotated_word_pairs` which contains word pairs and their annotated similarity score. Feel free to use a single file or multiple files and subdirectories inside

`conll03-en+de-test`, but make sure the same file structure is replicated in all other subdirectories in `$EVAL_ROOT/eval-data/ner/`. For example:

```
% for downstream evaluation tasks such as named entity
% recognition, the evaluation dataset may consist of
% two files of gold annotations: one for training an
% NER model, and another for evaluating the model.
cp ~/conll03/train.conll \
 $EVAL_ROOT/eval-data/ner/conll03-en+de-test/train.conll
cp ~/conll03/test.conll \
 $EVAL_ROOT/eval-data/ner/conll03-en+de-test/evaluate.conll

% replicate the same file structure for the dev dataset
cp ~/conll03/train.conll \
 $EVAL_ROOT/eval-data/ner/conll03-en+de-dev/train.conll
cp ~/conll03/dev.conll \
 $EVAL_ROOT/eval-data/ner/conll03-en+de-dev/evaluate.conll
```

3. Copy evaluation scripts. Create the subdirectory

`$EVAL_ROOT/[EVAL_TASK]_scripts` (replace `[EVAL_TASK]` with the name you picked for the evaluation task such as `ner`), and copy all necessary scripts to compute the evaluation metric in that directory. For example:

```
mkdir $EVAL_ROOT/ner_scripts
cp ~/stanford-ner-2015-12-09/stanford-ner-3.6.0.jar \
 $EVAL_ROOT/ner_scripts/
cp -r ~/stanford-ner-2015-12-09/lib \
 $EVAL_ROOT/ner_scripts/
```

4. Write a thin python wrapper. The wrapper must implement the method

`def evaluate(eval_data_dir, embeddings_filename)` which takes two arguments (path to the directory name of a compatible evaluation dataset such as

`$EVAL_ROOT/eval-data/ner/conll03-en+de-test` and path to an embeddings filename), and returns a tuple (score, coverage). Each line in the embeddings file consists of the same number of fields, separated by a single space character. The first field is a word (e.g., `en:dog`). The remaining fields specify the word embedding as a vector of real numbers. Both the score and the coveage should be in the range [0,1].

The wrapper must also implement the method

`def get_relevant_word_types(eval_data_dir)` which returns the list of word types needed to perform the evaluation with a given dataset. Since word embedding files tend be large, we filter out words which are not needed to save memory and disk space.

The wrapper script should live at

`$EVAL_ROOT/eval_[EVAL_TASK].py` (again, replace

`[EVAL_TASK]` with the name you picked for the evaluation task such as `ner`). Instead of writing the boilerplate from scratch, copy one of the exiting wrappers and reimplement the methods `evaluate` and

`get_relevant_word_types`. For example:

```
cp $EVAL_ROOT/eval_wordsim.py $EVAL_ROOT/eval_ner.py
% edit the implementation of the methods 'evaluate'
% and 'get_relevant_word_types' as needed
edit $EVAL_ROOT/eval_ner.py
```

5. Test the python wrapper. Call the wrapper from command line to make sure the returned score and coverage are correct. For example:

```
% requirements: python 2.7.3
cd $EVAL_ROOT
python eval_ner.py \
 --eval-dir eval-data/ner/conll03-en+de-test \
 --embeddings-file ~/sample-embeddings.vec
```

6. Tell the frontend about the new evaluation task and datasets. The frontend of the evaluation portal uses the file

`$EVAL_ROOT/tasks_datasets` to find out which evaluation tasks are available, and what evaluation datasets are compatible with them. Each line in this file consists of five space-delimited fields:

1. the name of a wrapper script (e.g., `eval_ner.py`),
2. the relative path to the directory of an evaluation dataset (e.g.,

   `eval-data/ner/conll03-en+de-dev`),
3. `dev` or `test`,
4. a bar-delimited list of 2-letter ISO 639-2 codes of languages covered in this dataset (e.g., `en|de`), and
5. the label to display for this task/dataset on the evaluation portal, followed by a tag for describing the task/dataset in `http://128.2.220.95/multilingual/tasks/`. Underscores will be replaced with spaces in the HTML page. E.g.,

   `[dev]_Monolingual_Word_Similarity#en-men-tr-3k`.

7. Make it official. Email waleed.ammar@gmail.com your github account along with a brief description of the evaluation task/datasets you would like to contribute. We will add you as a collaborator on GitHub so that you can push your changes to the main branch. Then, git add/commit/push your changes. For example:

```
git add $EVAL_ROOT/eval-data/ner
git add $EVAL_ROOT/ner_scripts
git add $EVAL_ROOT/eval_ner.py
git add $EVAL_ROOT/tasks_datasets
git commit -m 'adding named entity recognition \
 evaluation task with conll03 dev/test datasets'
git push
```

References

[Agić et al.2015] Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Jan Hajič, Anders Trærup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Krister Lindén, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Héctor Alonso Martínez, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Piitulainen, Barbara Plank, Prokopis Prokopidis, Sampo Pyysalo, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.1. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. 3.2

[Agirre et al.2009] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proc. of NAACL*, pages 19–27. 4.3.1

[Al-Rfou' et al.2013] Rami Al-Rfou', Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *CONLL*.

[Ammar et al.2012a] Waleed Ammar, Chris Dyer, and Noah A. Smith. 2012a. Transliteration by sequence labeling with lattice encodings and reranking. In *NEWS workshop at ACL*. 5.2.2

[Ammar et al.2012b] Waleed Ammar, Shomir Wilson, Norman Sadeh, and Noah A Smith. 2012b. Automatic categorization of privacy policies: A pilot study.

[Ammar et al.2013] Waleed Ammar, Victor Chahuneau, Michael Denkowski, Greg Hanneman, Wang Ling, Austin Matthews, Kenton Murray, Nicola Segall, Alon Lavie, and Chris Dyer. 2013. The cmu machine translation systems at wmt 2013: Syntax, synthetic translation options, and pseudo-references. In *Proc. of WMT*.

[Ammar et al.2014] Waleed Ammar, Chris Dyer, and Noah A. Smith. 2014. Conditional random field autoencoders for unsupervised structured prediction. In *Proc. of NIPS*.

[Ammar et al.2016a] Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016a. Many languages, one parser. In *Arxiv*.

[Ammar et al.2016b] Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016b. Massively multilingual word embeddings. In *Arxiv*. 3.3.1

[Andrew et al.2013] Galen Andrew, Raman Arora, Jeff A. Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *ICML*.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*.

[Bailey and Elkan1995] Timothy L Bailey and Charles Elkan. 1995. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine learning*. 5.2.2

[Bamman et al.2013] David Bamman, Adam Anderson, and Noah A Smith. 2013. Inferring social rank in an old assyrian trade network. *Digital Humanities*. 2.1.1

[Bansal et al.2014] Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[Barman et al.2014] Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *EMNLP 2014 workshop on computational approaches to code switching*. 3.1, 3.3.3

[Bellare et al.2009] Kedar Bellare, Gregory Druck, and Andrew McCallum. 2009. Alternating projections for learning with expectation constraints. In *Proc. of UAI*.

[Bender2011] Emily Bender. 2011. On achieving and evaluating language-independence in NLP. In *Linguistic Issues in Language Technology. Special Issue on Interaction of Linguistics and Computational Linguistics*. 3.1

[Berg-Kirkpatrick et al.2010] Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proc. of NAACL*. 5.5

[Bertsekas2011] Dimitri P Bertsekas. 2011. Incremental gradient, subgradient, and proximal methods for convex optimization: A survey. *Optimization for Machine Learning*, 2010:1–38. 5.3.2

[Blunsom and Cohn2006] Phil Blunsom and Trevor Cohn. 2006. Discriminative word alignment with conditional random fields. In *Proc. of Proceedings of ACL*. 5.2.2, 5.3.2

[Bottou2012] Léon Bottou. 2012. Stochastic gradient descent tricks. 5.3.2, 5.3.2

[Brown et al.1993] P F Brown, V J Della Pietra, S A Della Pietra, and R L Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. In *Computational Linguistics*. 5.2.2, 5.3.2, 5.3.2

[Bruni et al.2014] Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *JAIR*. 4.3.1

[Buchholz and Marsi2006] Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL-X*. 3.2.1, 5.3.1

[Camacho-Collados et al.2015] José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A unified multilingual semantic representation of concepts. In *Proc. of ACL*.

[Cappé and Moulines2009] Olivier Cappé and Eric Moulines. 2009. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*.

[Chen and Manning2014] Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP*.

[Cho et al.2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proc. of EMNLP*. 4.5

[Cohen et al.2011] Shay B. Cohen, Dipanjan Das, and Noah A. Smith. 2011. Unsupervised structure prediction with non-parallel multilingual guidance. In *Proc. of EMNLP*. 3.2.3

[Collobert and Weston2008] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*. 4.1, 5.5

[Collobert et al.2011] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. In *Proc. of JMLR*. 5.3.1

[Comrie1989] Bernard Comrie. 1989. *Language universals and linguistic typology: Syntax and morphology*. University of Chicago press. 2.1.3

[Das et al.2010] Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *Proc. of NAACL*.

[Daumé III2009] Hal Daumé III. 2009. Unsupervised search-based structured prediction. In *Proc. of ICML*.

[Dryer and Haspelmath2013] Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

[Duong et al.2015] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. 2015. A neural network model for low-resource universal dependency parsing. In *Proc. of EMNLP*. 6.1

[Durrett and Klein2013] Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proc. of EMNLP*.

[Durrett et al.2012] Greg Durrett, Adam Pauls, and Dan Klein. 2012. Syntactic transfer using a bilingual lexicon. In *Proc. of EMNLP*.

[Dyer et al.2010] Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proc. of ACL*. 5.3.2

[Dyer et al.2011] Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A. Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proc. of ACL-HLT*. 5.2.2

[Dyer et al.2013] Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *Proc. of NAACL*. 4.2, 5.3.2, 6.1

[Dyer et al.2015] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

[Fang and Cohn2016] Meng Fang and Trevor Cohn. 2016. Learning when to trust distant supervision: An application to low-resource pos tagging using cross-lingual projection. In *Proc. of CoNLL*.

[Faruqui and Dyer2014] Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. *Proc. of EACL*.

[Ganchev et al.2010] Kuzman Ganchev, João Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049.

[Gao and Vogel2008] Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *In Proc. of the ACL workshop*. 5.3.2, 6.1

[Gardner et al.2015] Matt Gardner, Kejun Huang, Evangelos Papalexakis, Xiao Fu, Partha Talukdar, Christos Faloutsos, Nicholas Sidiropoulos, and Tom Mitchell. 2015. Translation invariant word embeddings. In *Proc. of EMNLP*. 4.2

[Garrette and Baldridge2013] Dan Garrette and Jason Baldridge. 2013. Learning a part-of-speech tagger from two hours of annotation. In *Proc. of NAACL-HLT*, pages 138–147. 5.2.2

[Glorot and Bengio2010] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proc. of AISTATS*. 3.3

[Gouws et al.2014] Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2014. Bilbowa: Fast bilingual distributed representations without word alignments. *arXiv preprint arXiv:1410.2455*. 4.5

[Graves et al.2013] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proc. of ICASSP*. 3.2.5

[Gumperz1982] John J. Gumperz. 1982. *Discourse Strategies*. Studies in Interactional Sociolinguistics. Cambridge University Press. 5.3.3

[Guo et al.2015] Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proc. of ACL*. 3.2.3, 3.3.1, 4.5

[Guo et al.2016] Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proc. of AAAI*. 3.3, 4.1, 4.2, 4.5, 6.1

[Gutmann and Hyvärinen2012] Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *JMLR*. 2.2.2, 4.2.3

[Haghighi and Klein2006] Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proc. of NAACL-HLT*.

[Haghighi et al.2008] Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proc. of ACL*.

[Harris1954] Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162. 2.2.2

[Heilman and Smith2010] Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proc. of NAACL*.

[Henderson and Titov2010] James Henderson and Ivan Titov. 2010. Incremental sigmoid belief networks for grammar learning. *JMLR*.

[Henderson2003] James Henderson. 2003. Inducing history representations for broad coverage statistical parsing. In *Proc. of NAACL-HLT*.

[Hermann and Blunsom2014] Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual models for compositional distributed semantics. *arXiv preprint arXiv:1404.4641*. 4.1, 4.5

[Huang et al.2015] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. In *arXiv preprint arXiv:1508.01991*.

[Hwa et al.2005] Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering*, 11(03):311–325. 3.5

[Jelinek1997] Frederick Jelinek. 1997. *Statistical Methods for Speech Recognition*. MIT Press. 5.2.2

[Johnson2007] Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers? In *Proc. of EMNLP*. 5.2.2, 5.3.1, 5.3.1

[Kahki et al.2011] Ali El Kahki, Kareem Darwish, Ahmed Saad El Din, Mohamed Abd El-Wahab, Ahmed Hefny, and Waleed Ammar. 2011. Improved transliteration mining using graph reinforcement. In *Proc. of EMNLP*.

[Klein and Manning2004] Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proc. of ACL*. 5.2.2

[Klementiev et al.2012] Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proc. of COLING*. 4.1, 4.5

[Kočiskỳ et al.2014] Tomáš Kočiskỳ, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. *arXiv preprint arXiv:1405.0947*. 4.5

[Koehn et al.2003] Philipp Koehn, F. J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL*. 5.3.2

[Koehn2010] Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge. 5.3.2

[Lafferty et al.2001] John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*. 5.3.3

[Leviant and Reichart2015] Ira Leviant and Roi Reichart. 2015. Judgment language matters: Towards judgment language informed vector space modeling. In *arXiv preprint arXiv:1508.00106*. 4.4.1

[Lewis et al.2016] Paul M. Lewis, Gary F. Simons, and Charles D. Fennig. 2016. Ethnologue: Languages of the world. 2.1.1

[Li et al.2012] Shen Li, João Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proc. of EMNLP*. 5.2.2

[Liang and Klein2009] Percy Liang and Dan Klein. 2009. Online em for unsupervised models. In *Proc. of ACL*.

[Liang2005] Percy Liang. 2005. Semi-supervised learning for natural language. In *Thesis, MIT*. 10

[Lin et al.2014] Chu-Cheng Lin, Waleed Ammar, Chris Dyer, and Lori Levin. 2014. The cmu submission for the shared task on language identification in code-switched data. In *First Workshop on Computational Approaches to Code Switching at EMNLP*. 3.3.3

[Lin et al.2015] Chu-Cheng Lin, Waleed Ammar, Lori Levin, and Lori Levin. 2015. Unsupervised pos induction with word embeddings. In *Proc. of NAACL*. 5.4

[Liu et al.1989] D. C. Liu, J. Nocedal, and C. Dong. 1989. On the limited memory bfgs method for large scale optimization. In *Proc. of Mathematical Programming*. 5.2.4

[Lu et al.2015] Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proc. of NAACL-HLT*.

[Lui and Baldwin2012] Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proc. of ACL*. 3.3.1

[Lui and Baldwin2014] Marco Lui and Timothy Baldwin. 2014. Accurate language identification of twitter messages. In *Proc. of LASM*. 5.3.3

[Lukashin and Borodovsky1998] Alexander V Lukashin and Mark Borodovsky. 1998. Genemark. hmm: new solutions for gene finding. *Nucleic acids research*, 26(4):1107–1115. 5.2.2

[Luong et al.2013] Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proc. of CoNLL*.

[Luong et al.2015a] Minh-Thang Luong, Hieu Pham, and Christopher Manning. 2015a. Bilingual word representations with monolingual quality in mind. In *Proc. of NAACL-HLT*. 4.5

[Luong et al.2015b] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proc. of ACL*. 4.5

[Ma and Xia2014] Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proc. of ACL*.

[Martin2004] Samuel Elmo Martin. 2004. *A reference grammar of Japanese*. University of Hawaii Press.

[Martínez Alonso et al.2015] Héctor Martínez Alonso, Anders Johannsen, Sussi Olsen, Sanni Nimb, Nicolai Hartvig Sørensen, Anna Braasch, Anders Søgaard, and Bolette Sandford Pedersen. 2015. Supersense tagging for Danish. In *Proc. of NODALIDA*, page 21. 4.3.3

[Martínez Alonso et al.2016] Héctor Martínez Alonso, Anders Johannsen, Sussi Olsen, Sanni Nimb, and Bolette Sandford Pedersen. 2016. An empirically grounded expansion of the supersense inventory. In *Proc. of the Global Wordnet Conference*. 4.3.3

[Matthews et al.2014] Austin Matthews, Waleed Ammar, Archna Bhatia, Weston Feely, Greg Hanneman, Eva Schlinger, Swabha Swayamdipta, Yulia Tsvetkov, Alon Lavie, and Chris Dyer. 2014. The cmu machine translation systems at wmt 2014. In *Proc. of WMT*.

[McCarthy et al.1955] John McCarthy, Marvin Minsky, Nathan Rochester, and Claude Shannon. 1955. A proposal for the dartmouth summer research project on artificial intelligence.

[McDonald et al.2011] Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proc. of EMNLP*. 4.1

[McDonald et al.2013] Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of ACL*. 3.2, 3.3

[Merialdo1994] B Merialdo. 1994. Tagging english text with a probabilistic model. In *Proc. of Computational Linguistics*. 5.2.2

[Mikolov et al.2013a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proc. of ICLR*. 2.2.2, 4.4.2, 5.3.1, 5.3.3

[Mikolov et al.2013b] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*.

[Miller et al.1993] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunker. 1993. A semantic concordance. In *Proc. of HLT*, pages 303–308. 4.3.3

[Minka2005] Tom Minka. 2005. Discriminative models, not discriminative training. Technical report, Technical Report MSR-TR-2005-144, Microsoft Research.

[Mnih and Teh2012] Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proc. of ICML*. 5.5

[Montemagni et al.2003] Simonetta Montemagni, Francesco Barsotti, Marco Battista, Nicoletta Calzolari, Ornella Corazzari, Alessandro Lenci, Antonio Zampolli, Francesca Fanciulli, Maria Massetani, Remo Raffaelli, et al. 2003. Building the italian syntactic-semantic treebank. In *Treebanks*, pages 189–210. Springer. 4.3.3

[Naseem et al.2012] Tahira Naseem, Regina Barzilay, and Amir Globerson. 2012. Selective sharing for multilingual dependency parsing. In *Proc. of ACL*.

[Nivre and Nilsson2005] Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proc. of ACL*. 3.2.1

[Nivre et al.2007] Joakim Nivre, Johan Hall, Sandra Kubler, Ryan McDonald, Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *Proc. of CoNLL*. 3.2.1, 5.3.1

[Nivre et al.2015a] Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015a. Universal dependencies 1.2. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. 3.2, 3.3

[Nivre et al.2015b] Joakim Nivre, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Yoav Goldberg, Jan Hajič, Jenna Kanerva, Veronika Laippala, Alessandro Lenci, Teresa Lynn, Christopher Manning, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Maria Simi, Aaron

Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015b. Universal dependencies 1.0. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague. 3.2, 3.3.1

[Nivre2004] Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*. 3.2.2

[Och and Ney2003] F. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. In *Proc. of Computational Linguistics*. 5.3.2, 5.3.2

[Och1999] Franz Joseph Och. 1999. An efficient method for determining bilingual word classes. In *EACL*. 4.5

[Östling2015] Robert Östling. 2015. Word order typology through multilingual word alignment. In *Proc. of ACL-IJCNLP*.

[O'Connor et al.2010] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From tweets to polls: Linking text sentiment to public opinion time series. In *Proc. of ICWSM*.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proc. of ACL*. 5.3.2, 5.3.2

[Petrov et al.2012] Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*. 3.2, 3.2.3, 12, 5.3.1

[Poon and Domingos2008] Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Proc. of EMNLP*. 5.2.2

[Rasooli and Collins2015] Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proc. of EMNLP*.

[Ravi and Knight2009] Sujith Ravi and Kevin Knight. 2009. Minimized models for unsupervised part-of-speech tagging. In *Proc. of ACL*. 5.2.2

[Reddy and Waxmonsky2009] Sravana Reddy and Sonjia Waxmonsky. 2009. Substring-based transliteration with conditional random fields. In *Proc. of the Named Entities Workshop*. 5.2.2

[Rosenberg and Hirschberg2007] Andrew Rosenberg and Julia Hirschberg. 2007. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*. 5.3.1, 5.3.1

[Ryding2005] Karin C Ryding. 2005. *A reference grammar of modern standard Arabic*. Cambridge university press.

[Salton et al.1997] Gerard Salton, Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Automatic text structuring and summarization. *Information Processing & Management*, 33(2):193–207.

[Sankoff1998] David Sankoff. 1998. A formal production-based explanation of the facts of code-switching. *Bilingualism: language and cognition*, 1(01):39–50. 5.3.3

[Schneider et al.2013] Nathan Schneider, Behrang Mohit, Chris Dyer, Kemal Olflazer, and Noah A Smith. 2013. Supersense tagging for arabic: the mt-in-the-middle attack. In *Proc. of NAACL*. 3.5

[Smith and Eisner2005] Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proc. of ACL*. 5.2.2, 5.2.2, 5.5

[Socher et al.2010] Richard Socher, Christopher Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *NIPS workshop*. 5.5

[Søgaard et al.2015] Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *Proc. of ACL-IJCNLP 2015*.

[Solorio et al.2014] Thamar Solorio, Elizabeth Blair, Suraj Maharjan, Steve Bethard, Mona Diab, Mahmoud Gonheim, Abdelati Hawwari, Fahad AlGhamdi, Julia Hirshberg, Alison Chang, and Pascale Fung. 2014. Overview for the first shared task on language identification in code-switched data. In *Proceedings of the First Workshop on Computational Approaches to Code-Switching*. 5.3.3

[Spiegler et al.2010] Sebastian Spiegler, Andrew van der Spuy, and Peter A. Flach. 2010. Ukwabelana: An open-source morphological Zulu corpus. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 1020–1028, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Srivastava et al.2014] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. In *Proc. of JMLR*. 3.2.4, 3.2.4

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. of NIPS*.

[Swier and Stevenson2004] Robert Swier and Suzanne Stevenson. 2004. Unsupervised semantic role labelling. In *Proc. of EMNLP*. 5.2.2

[Szegedy et al.2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proc. of ICLR*. 4.3.3

[Täckström et al.2012a] Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012a. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proc. of NAACL-HLT*. 3.2.3

[Täckström et al.2012b] Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012b. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proc. of NAACL*, pages 477–487. 4.5

[Täckström et al.2013] Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. In *Proc. of TACL*.

[Tiedemann et al.2014] Jorg Tiedemann, Zeljko Agic, and Joakim Nivre. 2014. Treebank translation for cross-lingual parser induction. In *Proc. of CoNLL*. 3.5

[Tiedemann2014] Jorg Tiedemann. 2014. Rediscovering annotation projection for cross-lingual parser induction. In *Proc. of COLING*.

[Tiedemann2015] Jorg Tiedemann. 2015. Cross-lingual dependency parsing with universal dependencies and predicted pos labels. In *Proc. of Depling*. 3.2.3

[Titov and Henderson2007] Ivan Titov and James Henderson. 2007. Constituent parsing with incremental sigmoid belief networks. In *Proc. of ACL*.

[Toutanova et al.2015] K. Toutanova, W. Ammar, P. Chourdhury, and H. Poon. 2015. Model selection for type-supervised learning with application to pos tagging. In *Proc. of CoNLL*.

[Tsvetkov et al.2014] Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proc. of ACL*. 4.1

[Tsvetkov et al.2015a] Yulia Tsvetkov, Waleed Ammar, and Chris Dyer. 2015a. Constraint-based models of lexical borrowing. In *Proc. of NAACL*.

[Tsvetkov et al.2015b] Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015b. Evaluation of word vector representations by subspace alignment. In *Proc. of EMNLP*. 4.3.3

[Tsvetkov et al.2016] Yulia Tsvetkov, Sunayana Sitaram, Manaal Faruqui, Guillaume Lample, Patrick Littell, David Mortensen, Alan W. Black, Lori Levin, and Chris Dyer. 2016. Polyglot neural language models: A case study in cross-lingual phonetic representation learning. In *Proc. of NAACL*.

[Turian et al.2010] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proc. of ACL*. 2.2

[Vickrey et al.2010] David Vickrey, Cliff C Lin, and Daphne Koller. 2010. Non-local contrastive objectives. In *Proc. of ICML*. 5.5

[Vilares et al.2015] David Vilares, Miguel A Alonso, and Carlos Gómez-Rodríguez. 2015. One model, two languages: training bilingual parsers with harmonized treebanks. *arXiv preprint arXiv:1507.08449*.

[Vincent et al.2008] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proc. of ICML*. 5.5

[Vogel et al.1996] S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *Proc. of COLING*. 5.2.2

[Weber et al.2000] Markus Weber, Max Welling, and Pietro Perona. 2000. *Unsupervised learning of models for recognition*. Proc. of ECCV. 5.2.2

[Xiao and Guo2014] Min Xiao and Yuhong Guo. 2014. Distributed word representation learning for cross-lingual dependency parsing. In *Proc. of CoNLL*.

[Yamato et al.1992] Junji Yamato, Jun Ohya, and Kenichiro Ishii. 1992. Recognizing human action in time-sequential images using hidden Markov model. In *Proc. of CVPR*. 5.2.2

[Yarowsky et al.2001] David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proc. of HLT*. 3.5

[Zeman and Resnik2008] Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *Proc. of IJCNLP*. 4.1

[Zhang and Barzilay2015] Yuan Zhang and Regina Barzilay. 2015. Hierarchical low-rank tensors for multilingual transfer parsing. In *Proc. of EMNLP*. 3.3

[Zou et al.2013] Will Zou, Richard Socher, Daniel M Cer, and Christopher Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proc. of EMNLP*. 4.5