Machine Translation 4 Microblogs

Wang Ling

Ph.D. Thesis

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Dep. of Computer Science and Engineering
Instituto Superior Técnico
University of Lisbon
Av. Rovisco Pais, 1, 1049-001 Lisbon

Thesis Committee:
Alan W Black, Carnegie Mellon University
Isabel Trancoso, Instituto Superior Técnico
Nuno Mamede, Instituto Superior Técnico
Noah Smith, Carnegie Mellon University and University of Washington
Mário Figueiredo, Instituto Superior Técnico
Chris Callison-Burch, University of Pennsylvania

Thesis Advisors:
Alan W Black, Carnegie Mellon University
Isabel Trancoso, Instituto Superior Técnico
Chris Dyer, Carnegie Mellon University
Luísa Coheur, Instituto Superior Técnico

This work is supported by the Fundação de Ciência e Tecnologia through the Carnegie Mellon|Portugal Program, a joint program between the Portuguese Government and Carnegie Mellon University.

Abstract

The emergence of social media caused a drastic change in the way information is published. In contrast to previous eras in which the written word was more dominated by formal registers, the possibility for people with different backgrounds to publish information has caused non-standard style, formality, content, genre and topic to be present in written documents. One source of such data are posts in microblogs and social networks, such as Twitter, Facebook and Sina Weibo. The people that publish these documents are not all professionals, yet the information published can be leveraged for many ends [Han and Baldwin, 2011, Hawn, 2009, Kwak et al., 2010, Sakaki et al., 2010]. However, current NLP tasks perform poorly in the presence of this type of data, since they are modelled using traditional assumptions and trained on existing edited data. One problem is the lack of annotated datasets in this domain. One such assumption is of spelling homogeneity, where we assume that there is only one way to spell tomorrow, whereas in microblogs, this word can be abbreviated to tmrw (among many other options) or spelled erroneously as tomorow. It is shown in [Gimpel et al., 2011] that using in-domain data and defining more domain specific features can help address this problem for Part-of-Speech Tagging.

In this thesis, we address the challenge of NLP on the domain of informal online texts, with emphasis on Machine Translation. This thesis makes the following contributions in this respect. (1) We present an automatic method to extract such data automatically from microblog posts, by exploring the fact that many bilingual users post translations of their own posts. (2) We propose a compositional model for word understanding based only on the character sequence of those words, breaking the assumption that different word types are independent. This allows the model to generalize better on morphologically rich languages and the orthographically creative language used in microblogs. (3) Finally, we show improvements on several NLP tasks, both syntactically and semantically oriented, using both the crawled data and proposed character-based models. Ultimately, these are combined into a state-of-the-art MT system in this domain.

Abstract

A criação das redes sociais causou uma alteração drástica na forma que a informação épublicada. Em contraste com as eras passadas onde o texto publicado era dominado por registros formais, a possibilidade de publicação de informação por pessoas de differentes backgrounds criou a existência de estilo, formalidade, conteúdo, género e tópicos fora de padrão em documentos escritos. Uma fonte de dados deste tipo são os posts em microblogues e redes sociais, como o Twitter, Facebook e Sina Weibo. As pessoas que publicam estes documentos não são profissionais, mas esta informação pode ser usado para differentes fins [Han and Baldwin, 2011, Hawn, 2009, Kwak et al., 2010, Sakaki et al., 2010]. No entanto, sistemas de PLN existentes funcionam mal na presença deste tipo de dados neste tipo de dados, porque são modelados usando métodos tradicionais e treinados em dados anotados. Um dos problemas éa deficiência de dados anotados neste domínio. Uma das tais assunções éa homogeneidade da ortografia, onde assumimos que existe uma forma de escrever a palavra tomorrow, no entanto, em microblogues, esta palavra pode ser abreviado para tmrw (entre muitas outras opções) ou mal escrito como tomorow. Como foi demonstrado em [Gimpel et al., 2011], dados do mesmo domínio e features adicionais podem ser usados para tratar deste problema.

Nesta tese, tratamos do problema de PLN no domínio de textos onlines informais, com emfase em Tradução Automática. Esta tese faz as seguintes contribuições. (1) Apresentação de um método automático para minar este tipo de dados explorando o facto que muitos utentes traduzem os seus próprios tweets. (2) Propomos um modelo composicional capaz de entender palavras ao nível dos characteres, quebrando a assunção que differentes tipos são independentes. Isto permite ao modelo generalizar melhor para línguas ricas morfologicamente e domínios com criatividade lexical como nos microblogues. (3) Finalmente, mostramos que estes modelos podem melhorar várias tarefas em PLN, ambos em tarefas principalmente semânticas ou sintáticas, usando os dados minados e os modelos de characteres propostos. Ultimamente, estes são combinados em sistemas state-of-the-art neste domínio.

Contents

Chapter 1

Introduction

Microblogs, such as Twitter, Facebook and other non-American microblogs, like Sina Weibo, have gained tremendous popularity in the past 10 years. In addition to being an important form of communication for many people, they often contain extremely current, even breaking, information about world events and are reshaping the way information is published and analyzed [Han and Baldwin, 2011, Hawn, 2009, Kwak et al., 2010, Sakaki et al., 2010]. However, the writing style of microblogs tends to be quite colloquial, with frequent orthographic innovation (*R U still with me or what?*) and nonstandard abbreviations (*idk! smh*)—quite unlike the style found more traditional, edited genres. This poses considerable problems for traditional NLP tools, which were developed with other domains in mind, which often make strong assumptions about the type of the language we are processing, such as orthographic homogeneity (i.e., that there is just one way to spell *you*).

In this thesis, we will address the challenges posed by informal online text genres, with emphasis on machine translation. Machine translation is a fundamental task in NLP, and is composed by a large range of different components and concepts that must be addressed when we wish to adapt our models for this new domain. While many adaptation models [Koehn and Schroeder, 2007, Sankaran et al., 2012, Haddow, 2013, Foster and Kuhn, 2007] have been proposed to adapt MT to different domains, we believe that due to the popularity of microblogs and social media, this domain deserves a more focused effort from the research point of view. Translation in this domain will allow the information that is published to be readable to a larger audience. For instance, a translation system in this domain would allow non-Mandarin speakers to read posts in Mandarin from Chinese microblogs, such as Sina Weibo, or allowing non-English speakers to read English posts from Twitter.

1.1 Motivation

Improvements in NLP and MT systems are obtained using two approaches. Firstly, training the model with more labelled data, especially in-domain data, generally yields better overall results. Thus, methods that automatically obtain in-domain data can substantially boost the performance of existing systems. Secondly, learning a model that generalizes better for the specific problem can also yield improvements in translation quality.

1.1.1 Parallel Data Extraction from Microblogs

While much work has been done in the automatic extraction of parallel data from online domains [Resnik and Smith, 2003, Fukushima et al., 2006, Li and Liu, 2008, Uszkoreit et al., 2010a, Ture and Lin, 2012], most of these extract from carefully edited domains. Thus, the training data that is extracted seldom resembles microblog text. We start by introducing a method for finding naturally occurring parallel microblog text, which helps address the domain-mismatch problem. Our method is inspired by the perhaps surprising observation that a reasonable number of microblog users tweet "in parallel" in two or more languages. For instance, the American entertainer Snoop Dogg regularly posts parallel messages on Sina Weibo (Mainland China's equivalent of Twitter), for example, *watup Kenny Mayne!! - Kenny Mayne*，最近这么样啊！！, where an English message and its translation in Mandarin are in the same post, separated by a dash. Our method is able to identify and extract such translations. Briefly, this requires determining if a tweet contains more than one language, if these multilingual utterances contain translated material (or are due to something else, such as code switching), and what the translated spans are. We show that a considerable amount of parallel data can be extracted from Twitter and Sina Weibo.

1.1.2 Character-based Machine Translation and Natural Language Processing

A considerable factor for the inappropriateness of existing models to address the orthographic diversity observed in microblogs is the fact that existing models treat word types as independent of each other. That is, even though words *cool* and *coool* are lexically similar, this information is discarded by most models, and so

the models are not capable of generalizing that if *cool* is translated to *X*, the same applies to *coool*. We propose two ways to approach this problem. (1) One way is pre-process the input text, standardizing it so that it is simpler to process. The normalization of informal online messages is a task that has been a topic of interest in social media [Han et al., 2013, 2012, Kaufmann, 2010, Han and Baldwin, 2011]. One key application we developed during this thesis, is a microblog normalizer that learns to normalize microblog messages using the parallel data we crawled from microblogs. The approach is based on paraphrasing [Bannard and Callison-Burch, 2005, Ganitkevitch et al., 2013, Zhao et al., 2010, Madnani, 2010, Madnani and Dorr, 2010]. Paraphrasing is the process of rewriting sentences that convey the same meaning but with a different textual form [Barzilay and Lee, 2003]. In our work, we not do make many assumptions about the language, which allows us to build normalization systems for non-English languages. Furthermore, we incorporate character-level information during the process. Finally, applications of normalization systems are not restricted to MT, but also applicable to other NLP tasks. (2) The recent advances in word representation learning have shown that contextual similarity [Collobert et al., 2011, Mikolov et al., 2013] is a strong indicator for semantic similarity. That is, words such as *dog* and *cat* (or cool and cooool) can be found to be similar due to the fact that they occur in similar contexts. More concretely, words that are similar are mapped into a continuous space where similar words possess similar embeddings (similar values in their respective vector). However, in microblogs, new orthographic variants are created daily. For instance, the word *cool* can be spelled with an arbitrary number of *o*'s (e.g. *cooooooool*), and current methods attribute each variant with a different vector. Thus, while it is possible to learn similarities between frequent word types (e.g. *u* and *you*), this is not the case for variants that only occur once or twice in the corpora. Furthermore, it is impossible to store every word type in microblogs, as there are an unbounded number of word types that can be created. Thus, existing methods generally resort to approximations that prune less frequent word types [Mikolov et al., 2013]. In this thesis, we address these problems by introducing a method for learning word embeddings solely from their character sequence. That is, instead of storing each word type, we only store embeddings for characters, and generate the word representations using their character sequence. This allows us to build models that are orders of magnitude smaller than existing models. Furthermore, as lexical traits in words are directly explored in our compositional model, our models directly address lexical properties in words, such as morphology and orthographic innovation. More concretely, our model learns to understand the process where adding the suffix *ly* converts an adjective into an adverb, and also that regardless

of the number of *o*'s that are added to cool, it remains the same word. Finally, we show that our model can also learn non-compositional effects in language, that is, even though the words *butter* and *batter* are lexically similar, their meanings are different.

Aside from understanding words from characters, many models must also be able to generate words. Our main task, machine translation generates the target sentence word by word from the input sentence. Once again, the broadly used methods simply map this problem into predicting the correct word type from a list of existing types. Similar problems arise using these methods that are lexically oriented. For instance, word-based prediction models cannot generate unseen words. For instance, the model cannot generate the unseen word *performed* even though it has seen the word *perform*, and many present-past English verb constructions. Another common example in machine translation, is that many named entities (e.g. *Manders*) are not translated at all across European languages. Yet, most models cannot learn to simply copy the entity name to the output. These are generally implemented as special conditions during decoding or using post processing scripts to clean the translation output.

In this thesis, we shall propose models for the representation and generation of words at the character level, and show improvements in many NLP tasks in terms of quality of the end-to-end task, their performance and their compactness. Finally, we will show that in machine translation, this model can improve existing methods by allowing them to generalize better to lexically diverse languages (e.g. Turkish) and domains (e.g. Twitter).

1.2 Contributions and Thesis Statement

From the algorithmic point of view, we enrich the fields of NLP and MT fields with the following contributions:

Contribution 1: Parallel Data Extraction from Self-Contained Messages We present a parallel data extraction algorithm that allows parallel data extraction from self-contained documents [Ling et al., 2013b]. That is, unlike existing methods that extract data from different documents, our method can extract parallel sentences within the same message if it contains two or more languages.

Contribution 2: Character-based Microblog Normalization We present a paraphrasing model that can be trained on parallel data in order to generate paraphrases of input sentences [Ling et al., 2013a]. Trained on the extracted parallel data, it acts as a microblog normalizer, which can be used to standardize microblog data prior to translation.

Contribution 3: Character-based Word Representation We introduce a model for constructing vector representations of words by composing characters using bidirectional long short term memory (LSTM) recurrent neural networks [Ling et al., 2015c]. Relative to traditional word representation models that have independent vectors for each word type, our model requires only a single vector per character type and a fixed set of parameters for the compositional model. Despite the compactness of this model and, more importantly, the arbitrary nature of the form–function relationship in language. In most NLP components, our model out-performs existing models that consider word types as independent units.

Contribution 4: Character-based Word Generation We present a model for generating words at the character level. Relative to the more broadly used word softmax, our method is faster to compute and requires less parameters. Furthermore, it also allows the model to generate word forms that have not been seen in the training data.

Contribution 5: An Character-based Machine Translation Model We use the character-based representation and generation models to introduce a neural machine translation model that operates using characters as atomic units. We show that our model excels at morphologically rich languages and noisier data, such as microblogs, as it is sensitive to lexical traits in words. Furthermore, our model is an order of magnitude faster and more compact as it only the translation task is performed using characters as atomic unit.

In terms of applications, we improve the existing state-of-the-art systems by:

Contribution 1: Learning up-to-date translation units for machine translation units Our automatic parallel data extraction algorithm collects translations from up-to-date messages, which allows newer terms to be learnt. Thus, aside from learning lexical variants, we also learn translations for newly introduced entities,

such as the movie *Jurassic World*. Thus, we show that the collected data can also be used to improve translation in more standard domains, although the improvements are not as significant as when these are used to translate sentences in the same domain.

Contribution 2: Improvements in NLP for morphologically rich languages We show that our character-based word representation model can also be used for modeling words in morphologically rich languages. In languages such as Turkish, our models tend to obtain significant improvements over existing models using word lookups, in POS tagging and language modeling. Furthermore, it has been shown improvements parsing in Ballesteros et al. [2015].

Contribution 3: Significantly smaller models for NLP A second effect for using the character-based model is the fact that we model words using characters only. Thus, we no longer need to keep parameters for every word in the vocabulary, and replace a model for characters and their composition. This leads to dramatic reductions in the size of the models (less than 1% of the original model). Thus, it would be feasible for models trained on large datasets to be stored and used in mobile devices with limited memory.

Thesis Statement Our claim is that the way content is published is changes consistently and that existing MT and NLP methods, which were devised for processing standard text are not adequate and must be adapted to accomodate these changes. Firstly, existing parallel data mining methods that are structured, such as translated parliament data, or translated webpages, are no longer directly applicable in microblogs, as users tend to post self-contained translations. Secondly, most NLP and MT applications use words as atomic units of information, which is not well suited for addressing the lexical complexity underlying this domain. In this regard, we propose multiple character-based methods that model words as their sequence of characters.

1.3 Contributions from Satellite Work

Some of the work on the doctoral research leading to this thesis document shall be omitted for sake of coherence, as we are not directly related to the topic of this

thesis. Thus, we briefly describe the contributions and more information about them can be accessed in their referenced work.

Learning Reordering Models using Posterior Alignments We learn reordering models used in phrase-based models using soft alignment probabilities, rather than fixed alignments obtained from the Viterbi algorithm [Ling et al., 2011c,b].

Phrase Table Pruning based on Relative Entropy We propose a novel algorithm for filtering phrase tables for translations that are redundant (translations units that can be obtained by combining smaller units). This allows a significant reduction in the model size, with a small reduction in terms of translation quality [Ling et al., 2012a,b].

A Game Application for Learning a Second Language based on translation An application of MT to build a language learning game, where a human player plays a translation game against an automatic agent, which uses a MT system to play the game [Ling et al., 2011d].

A Named Entity Translation Extraction System using Anchor Links We learn named entity translations by exploring the fact that anchor links that link to the same entity generally contain the same meaning. If these are in different languages, it is a strong indicator that these are translations of the same entity [Ling et al., 2011a].

Improved Word Representation Learning We propose adaptations to the word representation models defined in [Mikolov et al., 2013], which can be applied to multiple neural NLP models. These relax the bag of words assumption defined in previous work, by defining position dependent objective functions [Ling et al., 2015b,a]. Improvements have been reported in both syntax-based tasks [Ling et al., 2015c, Tsvetkov et al., 2015], as well as in semantically related tasks [Astudillo et al., 2015b,a].

1.4 Thesis Structure

This thesis is divided in three parts as we will describe below:

Background Chapter 2 systemizes prior work relevant to the main topics addressed in this thesis.

- Section 2.1 describes the prior work on parallel data extraction.
- Section 2.2 summarizes the work on microblog normalization.
- Section 2.3 provides some insight on word representation learning and deep architectures for NLP.

Parallel Data Mining From Microblogs We describe our work on extracting parallel data from corpora from microblogs. This part is divided in the following chapters:

- Chapter 3 describes our automatic extraction algorithm to obtain large amounts of translated material from microblogs.
- Chapter 4 describes an application of the extracted data for microblog normalization, where parallel data is used as a pivot for learning standardized versions of existing words.

Character-based Models for NLP We propose a model for learning word representations and generation based on characters and apply these models different to NLP tasks, as detailed in the following chapters:

- Chapter 5 Describes the character-based word representation model and applies this model to improve language modeling and part-of-speech tagging.
- Chapter 6 Introduces a character-based word generation model, which first tested for character-based language modeling. Then, it combines the efforts in the previous chapters, building a machine translation system that operates on the character level, and showing that these models can potentially excel on morphologically rich languages, such as Portuguese, and lexically rich domains, such as Twitter.

Conclusion We conclude this thesis with an analysis of our work and delineating directions for future work.

Chapter 2

Background

2.1 Parallel Data Extraction

2.1.1 What are Parallel Corpora?

Bitext or parallel data consists of sentences that are translated in multiple languages. Generally, these are the result of translations produced by human translators on an original sentence. When translating from the source language s to a target language t, sentences in the s language can be thought of as input sentences, and the translations in language t as labels. Thus, the majority of machine translation systems [Koehn et al., 2007, Chiang, 2005, Bahdanau et al., 2015] are trained on such data, making it extremely valuable in the field of MT. Exceptions include rule-based MT systems that are built with hand-crafted rules, and work that learns translations from monolingual data [Ravi and Knight, 2011, Dou and Knight, 2012, Nuhn et al., 2012].

2.1.2 Automatic Parallel Corpora Mining

The automatic retrieval of parallel data is a well-studied problem. In particular, the automatic retrieval of parallel web documents has been a mainstream line of research [Resnik and Smith, 2003, Fukushima et al., 2006, Li and Liu, 2008, Uszkoreit et al., 2010b, Ture and Lin, 2012]. In general, these approaches extract bitext in two steps.

In the first step, these methods find multiple pairs of websites that are likely to be parallel. This step is necessary to narrow down the potential candidates for parallel websites as considering all possible pairs in a given set of webpages would be intractable. A simple approach named URL filtering [Resnik and Smith, 2003] builds this set of pairs, by checking for websites with patterns that may indicate the presence of a parallel website. For instance, the pattern *lang=en*, generally indicates that by changing *en* to another language code, such as *pt*, we may find another website, which is the translation of the original one. However, this method has a low recall, since it ignores the content within the web document. More robust approaches, such as the work described in [Uszkoreit et al., 2010b], translates all documents into English, and then finds rare ngrams that are common across web documents with different original languages. Then, all pairs between documents that share a rare ngram are extracted.

The second step determines whether a pair of websites is actually parallel. A language independent approach, named structural filtering [Resnik and Smith, 2003], is to compare the HTML structure of the pair of documents, which is done by measuring how well their HTML tags align. However, this method does not work well when webpages are only partially parallel. Content-based matching [Resnik and Smith, 2003] approaches attempt to measure how well the content of the two documents are translations of each other. One possible implementation is to word align [Brown et al., 1993] the documents, and then compute the following score:

$$S = \frac{\#\text{alignments}}{\#\text{alignments} + \#\text{unaligned words}} \tag{2.1}$$

Finally, a threshold is set and pairs of webpages are considered parallel if their score is higher than the threshold.

One approach to obtain word alignments is IBM Model 1, which we shall now review. Given a source sentence $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$ and a target translation $\mathbf{y} = \langle y_1, y_2, \ldots, y_m \rangle$, where n and m denote the number of tokens in $\mathbf{x}$ and $\mathbf{y}$, IBM Model 1 defines the probability of a lexical alignment $\mathbf{a}$ to translation $\mathbf{y}$ as

$$P_{M1}(\mathbf{a}, \mathbf{y} \mid \mathbf{x}) = \frac{1}{(n+1)^m} \prod_{i=1}^{m} t(y_i \mid x_{a_i}) \tag{2.2}$$

where $\mathbf{a}$ are the alignments between $\mathbf{x}$ and $\mathbf{y}$, and $t(y_i \mid x_{a_i})$ denotes the lexical translation probability that word y_i is the translation of x_{a_i}. More robust models consider other factors such as distortion and fertility, and generally find better

alignments, albeit at the cost of higher complexity. In our work, we use Model 1 as the basis, since its simplicity allows for a particularly tractable solution.[1]

Some work has also focused on a specific domain within the Web, such as the work in [Smith et al., 2010], which attempts to find parallel segments from Wikipedia. This work leverages the fact that Wikipedia contains pages that are written in multiple languages, and even though these are not translations, they are generally similar in content. Thus, parallel segments can be extracted using methods that retrieve parallel sentences from comparable corpora [Smith et al., 2010, Munteanu et al., 2004]. Likewise, the work in [Jehl et al., 2012] attempts to extract bitext from Twitter. This method relies on Cross-Lingual Information Retrieval techniques (CLIR) to extract English-Arabic translation candidates by using the Arabic tweets as search queries in a CLIR system. Afterwards, the model described in [Xu et al., 2001] is applied to retrieve a set of ranked translation candidates for each Arabic tweet, which are then used as parallel segments.

It is also possible to focus the extraction in one particular type of phenomena. The work on mining parenthetical translations [Lin et al., 2008], which attempts to find translations within the same document, has some similarities with our work, since parenthetical translations are within the same document. However, parenthetical translations are generally used to translate names or terms, which is more limited than our work targeting the extraction of whole sentence translations.

2.2 Microblog Normalization

2.2.1 What is Normalization?

Normalization is the task of standardizing the non-standard text found in non-edited domains such as Twitter. Consider the English tweet shown in the first row of Table 2.1 which contains several elements that NLP systems trained on edited domains may not handle well. First, it contains several nonstandard abbreviations, such as, *yea*, *iknw* and *imma* (abbreviations of *yes*, *I know* and *I am going to*). Second, there is no punctuation in the text although standard convention would dictate that it should be used.

While normalization to a form like *To Daniel Veuleman: Yes, I know. I am going*

[1]Furthermore, despite its simplicity, IBM Model 1 has previously shown good performance for sentence alignment systems [Xu et al., 2005, Braune and Fraser, 2010].

Table 2.1: Translations of an English microblog message into Mandarin, using three web translation services.

orig.	*To DanielVeuleman yea iknw imma work on that*
MT[1]	啊iknw DanielVeuleman伊马工作，
MT[2]	DanielVeuleman 是iknw 凋谢关于工作，
MT[3]	到DanielVeuleman是的iknw imma这方面的工作

to work on that. does indeed lose some information (information important for an analysis of sociolinguistic or phonological variation clearly goes missing), it expresses the propositional content of the original in a form that is more amenable to processing by traditional tools. Translating the normalized form with Google Translate produces 要丹尼尔*Veuleman*：是的，我知道。我打算在那工作。, which is a substantial improvement over all translations in Table 2.1.

2.2.2 Lexical Normalization

Most of the work in microblog normalization is focused on finding the standard forms of lexical variants [Yang and Eisenstein, 2013, Han et al., 2013, 2012, Kaufmann, 2010, Han and Baldwin, 2011, Gouws et al., 2011, Aw et al., 2006]. A lexical variant is a variation of a standard word in a different lexical form. This ranges from minor or major spelling changes, such as *jst*, *juxt* and *jus* that are lexical variants of *just*, to abbreviations, such as *tmi* and *wanna*, which stand for *too much information* and *want to*, respectively. Jargon can also be treated as variants, for instance *cday* is a slang word for *birthday*, in some groups.

There are many rules that govern the process lexical variants are generated. Some variants are generated from orthographic errors, caused by some mistake from the user when writing, or also purposely introduced for stylistic effects. For instance, the variants *representin*, *representting*, or *reprecenting* can be generated by a spurious letter swap, insertion or substitution by the user. One way to normalize these types of errors is to attempt to insert, remove and swap characters in a lexical variant until a word in a dictionary of standard words is found [Kaufmann, 2010]. Contextual features are another way to find lexical variants, since variants generally occur in the same context as their standard form. This includes orthographic errors, abbreviations and slang. However, this is generally not enough to detect lexical variants, as many words share similar contexts, such as *already*, *recently*

and *normally*. Consequently, contextual features are generally used to generate a confusion set of possible normalizations of a lexical variant, and then more features are used to find the correct normalization [Han et al., 2012]. One simple approach is to compute the Levenshtein distance to find lexical similarities between words, which would effectively capture the mappings between *representting*, *reprecenting* and *representin* to *representing*. However, a pronunciation model [Tang et al., 2012] would be needed to find the mapping between *g8*, *2day* and *4ever* to *great*, *today* and *forever*, respectively. Moreover, visual character similarity features would be required to find the mapping between *g00d* to *good*.

2.2.3 Sentence Normalization

Wang and Ng [2013] argue that the task of microblog normalization is not simply to map lexical variants into standard forms, but that other tasks, such as punctuation correction and missing word recovery should be performed. Consider the example tweet *you free?*, while there are no lexical variants in this message, the authors consider that the normalizer should recover the missing article *are* and normalize this tweet to *are you free?*. To do this, the authors train a series of models to detect and correct specific errors. While effective for narrow domains, training models to address each specific type of normalization is not scalable over all types of normalizations that need to be performed within the language, and the fact that a set of new models must be implemented for another language limits the applicability of this work.

Another strong point of the work above is that a decoder is presented, while the work on building dictionaries only normalize out of vocabulary (OOV) words. The work of [Han et al., 2012] trains a classifier to decide whether to normalize a word or not, but is still preconditioned on the fact that the word in question is OOV. Thus, lexical variants, such as, *4* and *u*, with the standard forms *for* and *you*, are left untreated, since they occur in other contexts, such as *u* in *u s a*. Inspired by the work above, we also propose a decoder based on the existing off-the-self tool Moses [Koehn et al., 2007], that “translates" from non-standard forms to normal forms.

Character-based MT methods have also been applied to normalization [Pennell and Liu, 2011, 2014], in these approaches, the phrase-based machine translation system is trained at the character level, as opposed as in the word level. These are trained in pairs of sentences where the non-standard sentence is aligned with the standardized/normalized sentence, resulting in a normalization system that learns

this mapping at the character level. There are also methods to find paraphrases automatically from Twitter [Xu et al., 2013], by finding tweets that contain common entities, such as *Obama*, that occur during the same period by matching temporal expressions. The resulting paraphrase corpora can also be used to train a normalizer using similar approaches.

2.3 Word Representations in Neural Networks

2.3.1 Word Lookup Tables

Most NLP methods convert words into a sparse representation, where word types are treated independently. For instance, multinomial distributions that compute the probability of word w being labelled as class c are generally modelled with a table containing a row for each word type and a column for each possible class. Similarly, in logistic regression a word is converted into one-hot representation $\mathrm{onehot}(w)$, which is a vector with the size of the vocabulary V and contains the value 1 in index w and zero in all other indexes. As a softmax is performed on this vector, each position of the vector is given its own set of parameters. One disadvantage of this model is that it assumes independence between words as each word type is given its unique set of weights.

On the other hand, vector space models capture the intuition that words may be different or similar along a variety of dimensions. Learning vector representations of words by treating them as optimizable parameters in various kinds of language models has been found to be a remarkably effective means for generating vector representations that perform well in other tasks [Collobert et al., 2011, Kalchbrenner and Blunsom, 2013, Liu et al., 2014, Chen and Manning, 2014]. Formally, such models define a matrix $\mathbf{P} \in \mathbb{R}^{d \times |V|}$, which contains d parameters for each word in the vocabulary V. For a given word type $w \in V$, a column is selected by right-multiplying $\mathbf{P}$ by a one-hot vector of length $|V|$, which we write $\mathbf{1}_w$, that is zero in every dimension except for the element corresponding to w. Thus, $\mathbf{P}$ is often referred to as word lookup table and we shall denote by $\mathbf{e}_w^W \in \mathbb{R}^d$ the embedding obtained from a word lookup table for w as $\mathbf{e}_w^W = \mathbf{P} \cdot \mathrm{onehot}(w)$. This allows tasks with low amounts of annotated data to be trained jointly with other tasks with large amounts of data and leverage the similarities in these tasks.

2.3.2 Word Representation Learning

A common practice to learn good representations for words is to initialize the word lookup table with the parameters trained on a proxy task for which a lot of data is available, such as, language modeling. Some examples of these include the skip-n-gram and continuous bag-of-words (CBOW) models of [Mikolov et al., 2013], but also extensions of these models Ling et al. [2015b].

Chapter 3

Automatic Microblog Parallel Data Extraction

Data is a major component in any statistical NLP system setup. It is needed both to train models and evaluate the quality of MT systems.

In statistical machine translation, we are particularly interested in **parallel data**, which are segments of text in multiple languages aligned in the sentence level. The existence of multiple parallel corpora supports the work done in this field. However, there is no available parallel corpora in the domain of microblogs and social networks. This is especially problematic, since using out-of-domain training data generally leads to worse results than using in-domain data, which stresses the importance of devising a system to crawl data automatically.

In this chapter, we shall describe our method for extracting parallel data from microblogs, such as Twitter and Sina Weibo (This work has been published in [Ling et al., 2013b]). It is inspired by the fact that a reasonable number of microblog users tweet "in parallel" in two or more languages. For instance, the American entertainer Snoop Dogg regularly posts parallel messages on Sina Weibo (Mainland China's equivalent of Twitter). Figures 3.1 and 3.2 show real parallel posts found in Sina Weibo. In both examples, posts are written first in English and then the same content is added in Mandarin, separated by a dash(-) character. While tweets that are translated are not exactly representative of microblogs as a whole, as not all messages are translated, these are better at representing the general language used in this domain than existing datasets.

Our goal is to find these posts and extract the Mandarin and English segments

from the tweet. Briefly, this requires determining if a tweet contains more than one language, if these multilingual utterances contain translated material (or are due to something else, such as code switching), and what the translated spans are.

Figure 3.1: Examples of parallel posts from Sina Weibo by Snoop Dogg.

3.1 The Intra-Document Alignment (IDA) Model

As previously stated, content-based filtering is a method for parallel data detection that relies on translation/alignment models [Brown et al., 1993, Vogel et al., 1996, *inter alia*]. However, such approaches reason about the probability that a pair of documents—in different languages—is parallel. In contrast, in our problem,

Figure 3.2: Examples of parallel posts from Sina Weibo by Psy.

translated material might be embedded in a single document, with different regions in different languages. In this section, we describe a model for identifying translated material in these documents.

The IDA model takes as input a document $\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle$, where n denotes the number of tokens in the document. Our goal is to find the location of the parallel segments, the languages of those segments and their word alignments. These are defined by the tuple $([p, q], l, [u, v], r, \mathbf{a})$, where the token indexes $[p, q]$ and $[u, v]$ are used to identify the left segment (from p to q) and right segment (from u to v), which are parallel. We shall refer by $[p, q]$ and $[u, v]$ as the **spans** of the left and right segments. To avoid overlaps, we set the constraint $p \leq q < u \leq v$. For simplicity, we also assume that there are at most 2 continuous segments that are parallel. Then, we use l and r to identify the language of the left and right segments, respectively. Finally, $\mathbf{a}$ represents the word alignment between the words in the left and the right segments. An example is provided in Figure 3.3 where, for the given tweet, the left and right segment spans $[1, 5]$ and $[11, 13]$ are identified in (1), the languages for these spans zh and en in (2) and the alignments $(2, 12), (3, 13)$,

indicating that words in indexes 2 and 3 are aligned to words at indexes 12 and 13. The main problem we address is to find the parallel data when the boundaries of the parallel segments are not explicitly defined. That is, to find the optimal segments $[1, 5]$ and $[11, 13]$, we must search through all other possible spans, such as $[1, 4]$ and $[11, 13]$. If the optimal indexes $[p, q]$ and $[u, v]$ were known a priori, it would reduce this problem into a regular word alignment problem, where we simply perform language detection on those segments, and align the segments and use Equation 2.1 to determine the likelihood that the segments are parallel.

The approach we devised to solve this problem finds the optimal segments $[p, q][u, v]$, languages l, r and word alignments **a** jointly. However, there are two problems with this approach that must be addressed. Firstly, word alignment models generally attribute higher scores to smaller segments, since these are the result of a smaller product chain of probabilities. In fact, since the model can freely choose the size of the segments $[p, q]$ and $[u, v]$, using a word alignment model (Equation 2.2) or content-based matching (Equation 2.1) almost always yields word-to-word translations. Secondly, inference must be performed over the combination of all latent variables $([p, q], l, [u, v], r, \mathbf{a})$, which can easily become intractable, even for short documents, using brute force approaches. These problems are described in detail and addressed in Section 3.1.1, 3.1.2 and 3.1.3. The extension of this method to more than a single language pair is presented in Section 3.1.4. Finally, we describe the metrics used to evaluate the quality of the extracted parallel segments in Section 3.1.5.

3.1.1 Model Components

As previously stated, an alignment model itself is not sufficient, since as more possible source alignments are available the probability of any alignment would decrease. Thus, there is a bias for selecting shorter spans.

To understand the problem, consider the example in Figure 3.3, and assume that we know the language pair l, r and the word alignments a. Suppose that the Chinese characters 起 and 努 are aligned to the words *fighting* and *together*, respectively. We now consider three possible segment spans $[p, q], [u, v]$: (1) The desired translation from 一起努力吧 to *We fighting together*; (2) the segment of all aligned words translating from 起努 to *fighting together*; (3) and the segment with a word-to-word translation from 起 to *together*. According to Equation 2.2, the translation probabilities would be calculated as:

- (1) $\frac{1}{4^6} P_{\text{M1}}(\text{together} \mid \text{起}) P_{\text{M1}}(\text{fighting} \mid \text{努})$
- (2) $\frac{1}{2^3} P_{\text{M1}}(\text{together} \mid \text{起}) P_{\text{M1}}(\text{fighting} \mid \text{努})$
- (3) $\frac{1}{1^3} P_{\text{M1}}(\text{together} \mid \text{起})$

By observing these probabilities, we notice two problems with this model. Firstly, the model is unlikely to choose segments with unaligned words. This can be seen by comparing the first and second cases, where it is evident that the first case's translation probability is bounded by the second one regardless of the word translation probabilities, as the normalization factor $\frac{1}{n^{m+1}}$ is inversely proportional to the number of words in the segments. A more aggravating issue is the fact that the model always picks word-to-word translation segments. This can be seen by comparing case (2) and (3), where we can observe that the second case is bounded by the third one, as the second case is a product of more translation probabilities.

Similar results are obtained if we consider Equation 2.1, where the example (1) obtains a lower score than (2) and (3), as it contains unaligned words.

To address this problem, we propose a three-feature model that takes into account the spans of the parallel segments, their languages, and word alignments. This model is defined as follows:

$$\textit{score}([u,v], r, [p,q], l, \mathbf{a} \mid \mathbf{x}) = \underbrace{\phi_S([p,q],[u,v] \mid \mathbf{x})}_{\text{span score}} \times \underbrace{\phi_L(l, r \mid [p,q],[u,v], \mathbf{x})}_{\text{language score}} \times \underbrace{\phi_T(\mathbf{a} \mid [p,q], l, [u,v], r, \mathbf{x})}_{\text{alignment score}}$$

Each of the component scores (ϕ_S, ϕ_L and ϕ_T) returns a score in the interval $[0, 1]$ and each score is conditioned on different latent variables. We shall describe the role each score plays in the model.

Translation Score

The translation score $\phi_T(\mathbf{a} \mid [p,q], l, [u,v], r)$ indicates whether $[p,q]$ is a reasonable translation of $[u,v]$, according to the alignment $\mathbf{a}$. Previously, we relied on pre-trained IBM Model 1 probabilities for this score:

$$\phi_{M1}(\mathbf{a} \mid [p,q], l, [u,v], r, \mathbf{x}) = \frac{1}{(q-p+2)^{v-u+1}} \prod_{i=u}^{v} t_{\text{M1}}^{l \to r}(x_i \mid x_{a_i}).$$

The equation above is a reformulation of the Model 1 presented in Equation 2.2 for a single document with the segment p, q as source, and u, v as target. The lexical tables $t_{\text{M1}}^{l \to r}$ for the various language pairs are trained a priori using available parallel corpora. The null translation probability is set to ϵ, which is set to a negligible probability, so that these are only used if x_i cannot be aligned to any word. Note that the translation score by itself also allows the identification of the language pair l, r, as using a lexical table from an incorrect language pair is likely to yield lower scores.

However, when considering multiple language pairs, lexical translation probabilities for pairs of words in some language pairs tend to be higher, as there are less equivalent translations for the same word. This would bias the model to pick those language pairs more often, which is not desirable. Thus, we redefine this score by adapting the content-based matching metric, which assumes lexical translation uniformity:

$$\phi_{match}(\mathbf{a} \mid [p,q], l, [u,v], r, \mathbf{x}) = \frac{\#\mathbf{a}}{\sum_{i \in [p,q] \cup [u,v]} \delta(i \notin a)}$$

However, the computation of the alignments **a** for a given language pair l, r is still performed under IBM model 1.

Finally, as performing word alignments for different directions can generate a different set of alignments, we also compute the alignments from r to l. These alignments are generated for both directions, which are denoted as $\mathbf{a}_{l,r}$ and $\mathbf{a}_{r,l}$. The translation score is defined as:

$$\phi_T = max(\phi_{match}(\mathbf{a}_{l,r} \mid [p,q], l, [u,v], r, \mathbf{x}), \phi_{match}(\mathbf{a}_{r,l} \mid [u,v], r, [p,q], l, \mathbf{x}))$$

Span Score

The span score $\phi_S([p,q],[u,v] \mid l, r, \mathbf{x})$ of hypothesized pair of segment spans $[p,q],[u,v]$ is defined as:

$$\phi_S(l, r \mid [p,q],[u,v] \mid \mathbf{x}) = \frac{(q-p+1)+(v-u+1)}{\sum_{0<p' \leq q' < u' \leq v' \leq n}(q'-p'+1)+(v'-u'+1)} \times \delta([p,q],[u,v] \notin \omega(l,r,\mathbf{x}))$$

The span score is a distribution over all spans that assigns higher probability to segmentations that cover more words in the document. It is highest for segmentations that cover all the words in the document, which allows the model to consider larger segments.

It also defines the function δ that takes the values 1 or 0 depending on whether all the segment boundaries are valid according to the constraints list ω. The only exception occurs when there are no valid boundaries, in which case we set δ to always yield 1. These constraints are used to define hard constraints regarding a segment's validity based on prior knowledge.

The first intuition that we wish to include is that consecutive words in the same unicode range tend to belong to the same sentence. For instance, in the example in Figure 3.3, segments *fighting together* or *fighting* would violate the constraint, since they do not include the whole sequence of Latin words *We fighting together*. However, the segment *tag: We fighting together* or any other segment that contains the whole sequence of Latin words is acceptable. The same applies to the Mandarin segment 一起努力吧.

The second constraint ensures that a segment containing a parenthesis starter also contains the corresponding parenthesis ender. For instance, for the tweet *Yoona taking the '身体健康' (be healthy)* ^ ^, the segment *(be healthy* would not be valid, since it does not contain the parenthesis closer *)*. However, both *(be healthy)* and *be healthy* are acceptable. The exception to this rule occurs when either the parenthesis starter or ender is missing in the tweet, in which case this constraint is ignored. We consider the following universal starter and ender pairs: *()*,*[]* and {}; as well as the Mandarin and Japanese pairs: （）, 【】, 〔〕 and 「」.

With the inclusion of the span score, the model can successfully address the example in Figure 3.3, as the span score gives preference to longer segments and constraints force the model to use complete sentences. Consider the French-English example tweet *Qui est le véritable avare ? Who is the real miser ?*, and assume that the alignment model is able to align *est* to *is*, *le* to *the* and the left question mark *?* is aligned to the right one. In this case, the current model would be able to find the correct segments as the constraints would enforce that the left segment contains *Qui est le véritable avare* and the right segment contains *Who is the real miser*. Furthermore, as the question marks are aligned, the model is capable of correctly extracting the translation from *Qui est le véritable avare?* to *Who is the real miser?*.

While the alignment and span scores can address most tweets containing trans-

lations, they still suffer from the lack of language detection. For instance, consider that the previous example was tweeted as *Who is the real miser Qui est le véritable avare*. We can see that correctly translated segments from *Who is the real miser* to *Qui est le véritable avare* and the incorrect segments that translate from *Who is the real* to *miser Qui est le véritable avare* would receive the same score according to the span score, since they cover all words in the tweet. This is because the model does not know whether *miser* belongs to the English or French segment. One solution to solve this would be to augment the lexical table so that the alignment model can align the word *miser* to *avare*, so that the alignment score is higher in the first case. However, this is an unreasonable requirement as it would require large amounts of parallel data. An easier solution would include a language detector in order to identify that it is likely that *miser* is an English word, so it is unlikely that it would be placed in the French segment. This information is encoded in the language score.

Language Score

The language score $\phi_L(l, r \mid [p, q], [u, v], \mathbf{x})$ indicates the degree that the language labels l, r are appropriate to the document contents:

$$\phi_L(l, r \mid [p, q], [u, v], \mathbf{x}) = \frac{\sum_{i=p}^{q} P_L(l, x_i) + \sum_{i=u}^{v} P_L(r, x_i)}{(q - p + 1) + (v - u + 1)}$$

where $P_L(l, x_i)$ is a language detection function that yields the probability that word x_i is in language l. In our previous work [Ling et al., 2013b], this was a binary function based on unicode ranges, which assumed that all Latin words were English and all Han Characters were Mandarin. This was not a problem, since we were only extracting Arabic–English and Mandarin–English pairs. To cope with any language pair, including pairs where both languages have the same unicode range, we estimate $P_L(l, x)$ by training a character-based language detector[1], and calculating the posterior probability of a language l given a word x.

The score benefits the model in two aspects. Firstly, it addresses the problem mentioned above, so that we can address tweets with languages in the same unicode range. Secondly, it allows a better identification of the language pair l, r than simply using the alignment model. The reason for this is the fact that some languages, such as Portuguese and Spanish or Mandarin and Japanese contain overlapping words. For instance, in the tweet よい末を！ *Have a nice weekend! #ohayo*, the

[1]http://code.google.com/p/language-detection/

characters 末 mean *weekend* for both Mandarin and Japanese. Thus, if these are the only characters that are aligned during the computation of the translation score, both languages are equality likely to be correct. However, the Hiragana characters よ,い and を only exist in Japanese, so it is likely that the segment よい末を！ is in Japanese and not in Mandarin. These details are considered by the language score.

3.1.2 Inference

In our inference problem, we need to efficiently search over all pairs of spans, languages, and word alignments. We show that, assuming a Model 1 alignment model, an optimal dynamic programing inference algorithm is available. Our search problem is formalized as follows:

$$([p,q],l,[u,v],r)^* = \arg \max_{[p,q],l,[u,v],r} \max_{\mathbf{a}} \textit{score}([p,q],l,[u,v],r,\mathbf{a} \mid \mathbf{x}). \tag{3.1}$$

A high model score indicates that the predicted bispan $[p,q],[u,v]$ is likely to correspond to a valid parallel span.

A naïve approach to solving this problem would require each of the scores ϕ_S, ϕ_L and ϕ_T to be computed for each possible variation of l,r,p,q,u,v. Calculating the span score ϕ_S for all combinations of u,v,p,q values only requires a sum of boundary positions. As for the set of constraints, the set of invalid boundaries ω only needs to be computed once for each document.

As for the language score ϕ_L, the language probabilities for each word $P_L(l,x)$ only need to be computed once. The detector can compute the language probabilities for a word x for all considered languages in linear time over the document length $|\mathbf{x}|$. Furthermore, the number of character ngrams that need to be considered is relatively small, as we are applying it on words rather than documents. Once $P_L(l,x)$ is computed for all possible values of l and x, calculating the language score can be trivially computed.

However, computing the translation score ϕ_T requires the computation of the word alignments in ϕ_T over all possible segmentations, which requires $O(|\mathbf{x}|^6)$ operations using a naive approach ($O(|\mathbf{x}|^4)$ possible segmentations, and $O(|\mathbf{x}|^2)$ operations for aligning each pair of segments). Even if only one language pair l,r is considered, a document with 40 tokens would require approximately 7 million operations in the worst case for ϕ_T to be computed for all segmentations. To process millions of documents, this process would need to be optimized.

We show that we can reduce the order of magnitude of this algorithm to the optimal $O(|\mathbf{x}|^4)$ without resorting to approximations by using a dynamic programming algorithm. This is done by showing that, under Model 1, Viterbi alignments do not need to be recomputed every time the segment spans are changed, and can be obtained by updating previously computed ones. Thus, we propose an iterative approach to compute the Viterbi word alignments for IBM Model 1 using dynamic programming in Section 3.1.3.

Furthermore, computing the alignments using all possible values of l, r is unnecessarily expensive, and we show that we can limit the number of pairs without approximations by using the language identification approach described in Section 3.1.4.

3.1.3 Dynamic programming search

We leverage the fact that the Viterbi word alignment of a bispan (or pair of spans) can be reused to calculate the Viterbi word alignments of larger bispans. The algorithm considers a 4-dimensional chart of bispans and computes the Viterbi alignment for the minimal valid span (i.e., $[0, 0], [1, 1]$). Then, it progressively builds larger spans from smaller ones. Let $A_{p,q,u,v}$ represent the Viterbi alignment of the bispan $[p, q], [u, v]$. Each of the four dimensions p, q, u, v of the chart can be manipulated using λ recursions, which guarantees that the new alignment would be optimal according to IBM Model 1. The following update functions are defined:

- $A_{p,q,u,v+1} = \lambda_{+v}(A_{p,q,u,v})$ inserts one token to the end of the right span with index $v + 1$. Only the optimal alignment for that token is needed, which requires iterating over all the tokens in the left span $[p, q]$, and possibly updating their alignments. See Figure 3.4 for an illustration.

- $A_{p,q,u+1,v} = \lambda_{+u}(A_{p,q,u,v})$ removes the first token of the right span with index u. The Viterbi alignment remains the same with the exception that alignments to the removed token at u must be removed. This can be done in time $O(1)$.

- $A_{p,q+1,u,v} = \lambda_{+q}(A_{p,q,u,v})$ inserts one token to the end of the left span with index $q + 1$. All words in the right span must be rechecked, since aligning them to the inserted token at $q + 1$ may yield a better alignment score. This update requires $n - q + 1$ operations.

- $A_{p+1,q,u,v} = \lambda_{+p}(A_{p,q,u,v})$ removes the first token of the left span with index p. After removing the token, new alignments must be found for all tokens

that were aligned to p. Thus, the number of operations for this update is $K \times (q - p + 1)$, where K is the number of words that were originally aligned to p. In the best case, no words are aligned to the token in p, so nothing needs to be done. However, in the worst case, when all target words were originally aligned to p, this update would result in the recalculation of all alignments.

The algorithm proceeds until all valid cells have been computed, and retrieves the value of the best cell. The most important aspect to consider when updating existing cells is that the update functions differ in complexity, and the sequence of updates used defines the performance of the system. Most spans are reachable using any of the four update functions. For instance, the span $A_{2,3,4,5}$ can be reached using $\lambda_{+v}(A_{2,3,4,4})$, $\lambda_{+u}(A_{2,3,3,5})$, $\lambda_{+q}(A_{2,2,4,5})$ or $\lambda_{+p}(A_{1,3,4,5})$. However, it is desirable to apply λ_{+u} whenever possible, since it only requires one operation. Yet, this is not always feasible. For instance, the state $A_{2,2,2,4}$ cannot be reached using λ_{+u}, as the state $A_{2,2,1,4}$ does not exist, since it breaks the condition $p \leq q < u \leq v$. In this situation, incrementally more expensive updates must be considered, such as λ_{+v} or λ_{+q}, which are in the same order of complexity. Finally, we want to minimize the use of λ_{+p}, which may require the recomputation of the Viterbi Alignemnts. Formally, we define the following recursive formulation that guarantees an optimal outcome:

$$A_{p,q,u,v} = \begin{cases} \lambda_{+u}(A_{p,q,u-1,v}) & \text{if } u > q+1 \\ \lambda_{+v}(A_{p,q,u,v-1}) & \text{else if } v > q+1 \\ \lambda_{+p}(A_{p-1,q,u,v}) & \text{else if } q = p+1 \\ \lambda_{+q}(A_{p,q-1,u,v}) & \text{otherwise} \end{cases}$$

This transition function applies the cheapest possible update to reach state $A_{p,q,u,v}$.

If we consider that we always choose the least complex update function, we reach the conclusion that this algorithm runs at $O(n^4)$ time. Starting by considering the worst update λ_{+u}, we observe that this is only needed in the following cases $[0,1][2,2], [1,2][3,3], \cdots, [n-2,n-1][n,n]$, which amounts to $O(n)$ cases. Since this update is quadratic in the worst case, the complexity of these operations is $O(n^3)$. The update λ_{+q} is applied to the bispans $[*,1][2,2], [*,2][3,3], \cdots, [*,n-1], [n,n]$, where $*$ denotes an arbitrary number within the span constraints, but not present in previous updates. Given that this update is linear, and we need to iterate through all tokens twice, this update also takes $O(n^3)$ operations in total. The update λ_{+v} is applied for the cells $[*,1][2,*], [*,2][3,*], \cdots, [*,n-1], [n,*]$. Thus, with three

degrees of freedom and a linear update, it runs in $O(n^4)$ time. Finally, update λ_{+u} runs in constant time, but is needed for all remaining cases, so it also requires $O(n^4)$ operations. Hence, by summing the executions of all updates, we observe that the order of magnitude of our exact inference process is $O(n^4)$. Note that for exact inference, a lower order would be unfeasible, since simply iterating all possible bispans once requires $O(n^4)$ time.

3.1.4 Language Pair Filtering

While running alignments for all language pairs does not raise the order of the algorithm itself, since the number of existing languages is limited, it still involves a considerable additional run time that can be avoided. For instance, if we consider 20 language pairs, we must run the alignment algorithm 20 times. However, in most cases, many language pairs can be trivially ruled out. For instance, if there are no Han characters, any pair involving Japanese or Mandarin can be excluded. Thus, one can efficiently filter out unlikely language pairs without losses.

The combined score (IDA score) is composed by the product of the span, language and translation scores. The computation of the translation score requires checking a lexical translation table in each operation, which is computationally expensive, so dynamic programming is required. However, the span and language scores can be more efficiently computed, so this operation is only expensive if we compute the translation score.

Thus, we define the *incomplete* IDA score as:

$$S_{inc}([u,v], r, [p,q], l \mid \mathbf{x}) = \phi_S([p,q], [u,v] \mid l, r, \mathbf{x}) \times \phi_L(l, r \mid [p,q], [u,v], \mathbf{x}).$$

We can trivially show that *score*$([u,v], r, [p,q], l, \mathbf{a} \mid \mathbf{x})$ is bounded by $S_{inc}([u,v], r, [p,q], l \mid \mathbf{x})$, as the incomplete score does not include the product of the translation score, in the $[0,1]$ interval. This means that if we know that the highest $score$ value for Mandarin-English is 0.4, and wish to check if we can obtain a better score for Arabic-English, we can first compute the S_{inc} score for Arabic-English, and check if it is higher than 0.4. If it is not, we can skip the computation of the alignment score for this language pair, as it would not be higher than 0.4. Thus, if we find that the highest S_{inc} score among all possible segmentations of p, q, u, v score for a given language pair l_1, r_1 is lower than *score* for any other language pair, it follows that l_1, r_1 is never the highest scoring language pair. More formally, we can discard a

language pair l_1, r_1 if the following condition applies:

$$\exists_{l \neq l_1, r \neq r_1} \max_{[p,q],[u,v]} S_{inc}([p,q], l_1, [u,v], r_1 \mid \mathbf{x})$$
$$\leq \max_{[p,q],[u,v]} \max_{\mathbf{a}} \textit{score}([p,q], l, [u,v], r, \mathbf{a} \mid \mathbf{x})$$

Our method starts by computing the incomplete IDA scores S_{inc} for all values of r, l. Then, starting from the highest scoring language pairs l, r, we compute their complete IDA scores, while keeping track of the highest IDA score. The algorithm can stop once it reaches a language pair whose incomplete IDA score is lower than the highest complete IDA score, as we know that they never achieve the highest complete IDA score.

3.1.5 Evaluation Metrics

The goal of the IDA model is to find the optimal parallel segments $[p,q][u,v]$, their languages l, r and the alignments **a**. The evaluation metric we define tests whether the predicted values of $[p_h, q_h], l_h, [u_h, v_h], r_h$ are determined accurately with their reference values $[p_r, q_r], l_r, [u_r, v_r], r_r$. We refrain from testing the optimal alignments **a**, as gold standards with word alignments are generally complex and expensive to build. The reference values are obtained manually from human annotations, and we present methods to crowdsource these annotations in [Ling et al., 2014].

We start by defining the intersection and union between two segments $[a,b] \cap [a',b']$ and $[a,b] \cup [a',b']$. The intersection between two segments $[a,b] \cap [a',b']$, namely $[a,b]$ and $[a',b']$ computes the number of tokens within the intersection of the intervals, as given by $[max(a,a'), min(b,b')]$. Similarly, the union between two segments $([a,b] \cup [a',b'])$ computes the number of tokens within the union of the intervals, given by $[min(a,a'), max(b,b')]$. One important aspect to consider is that the segments can span half words, which can happen if there is a missing space between a sentence pair boundary, such as *uneasyBom*, which is contains the English word *uneasy* and the Portuguese word *Bom*. To cope with this, we use the simple method where we add the fractional count as the ratio between the number of characters that is included in the interval and total number of characters in the token. Thus, the segment corresponding to *uneasy* in *uneasyBom*, would correspond to two-thirds of a single word.

A hypothetical segment $[a_h, b_h]$ with language l_h is scored against the reference $[a_h, b_h]$ with language l_r as:

$$S_{seg}([a_h, b_h], l_h, [a_r, b_r], l_r) = \frac{\textbf{intersect}([a_h, b_h], [a_r, b_r])}{\textbf{union}([a_h, b_h], [a_r, b_r])} \delta(l_h = l_r) \tag{3.2}$$

This score penalizes the hypothesis segment for each extra token not in the reference, as well as each token in the reference that is not in the hypothesis. Furthermore, segments that differ in language would have a zero score. Unlike our previous work, we decided not to evaluate the language pair detection as a separate task, as only a negligible number of spurious parallel sentences (less than 0.1%) are caused by a incorrect detection of the language pair.

The final score S_{IDA} is computed as the harmonic mean between the segment scores of the parallel segments:

$$S_{IDA}([p_h, q_h], l_h, [u_h, v_h], r_h, [p_r, q_r], l_r, [u_r, v_r], r_r) = \tag{3.3}$$

$$\frac{2S_{seg}([p_h, q_h], l_h, [p_r, q_r], l_r) S_{seg}([u_h, v_h], r_h, [u_r, v_r], r_r)}{S_{seg}([p_h, q_h], l_h, [p_r, q_r], l_r) + S_{seg}([u_h, v_h], r_h, [u_r, v_r], r_r)} \tag{3.4}$$

We opt to compute the harmonic mean (as opposed to a arithmetic mean), since it emphasizes that both parallel segments must be accurate to obtain a high score. This is because parallel segments are only useful when both sides are accurate on the span and language.

3.2 Parallel Data Extraction

The previous section describes a method for finding the optimal segments p, q, u, v and the language pair l, r and the alignments **a**, according to the IDA model score, for any document x. However, extracting parallel sentences using this model requires addressing other issues, such as identifying the tweets that contain translations from those that do not. This section describes how the parallel data extraction process is performed. We start by assuming that we have access to a sample of tweets (e.g. from Twitter), which we denote as T.

The process is divided into three steps. Firstly, we filter the set T in order to remove all monolingual tweets, which results in a set T_{mult} composed solely by multilingual tweets, which substantially reduces the number of tweets that need to be processed by the following steps. Secondly, assuming that all tweets T_{mults} contain parallel data, we extract the parallel segments using the IDA model, which

are placed in one of the sets $D_{s,t}$, where s and t denote the language pair of each parallel segment. The fact that we are applying the IDA model to tweets that may not contain translations means that many instances in $D_{s,t}$ would not be parallel. Thus, as the last step, we filter these instances, using a classifier that is trained to predict whether each sample in $D_{s,t}$ is actually parallel. These steps are described in detail in Sections 3.2.1, 3.2.2 and 3.2.3.

3.2.1 Filtering

The first step is to filter the set of tweets T, so that only multilingual tweets are kept. A tweet is multilingual if there is more than one language within it. Thus, these can contain not only translated material, but also case switching, use of foreign words, or simply sentences in multiple languages. Obviously, if a tweet is not multilingual, it can automatically be disregarded as a potential container of bitext.

Our previous approach for extracting Mandarin-English sentence pairs considered only tweets that simultaneously contained a trigram of Mandarin Characters and a trigram of Latin words. However, this approach is only effective at finding multilingual tweets with languages that do not share the same unicode range. Thus, to allow the extraction of parallel data for language pairs, such as English-Spanish, a more general method is needed.

Multilingual Document Detection

Once again, a challenge in this problem is that language detectors do not identify multiple languages within a document. There are some recently proposed methods for this [Lui et al., 2014], they are not tuned for efficiency, which is the main requirement for this step.

We use an approach similar to the one devised in Section 3.1.1, where we adapted a word-based language detection approach. The approach is based on estimating the probability that two tokens a and b are in different languages, which is given as:

$$P_{mult}(a,b) = 1 - \sum_{i \in L} P_L(a,i) P_L(b,i) \tag{3.5}$$

where $P_L(x,i)$ is once again the probability that token x is in language i, according to a character based model and L is the set of all languages considered. Thus, given a tweet, our model attempts to find a pair of words where $P_{mult}(a,b)$ is higher than

0.95[2]. For instance, in the tweet *eu quero ver este cartoon*, where the message is mainly in Portuguese except for the word *cartoon*, which is in English, the model can use the high probability in $P_{mult}(ver, cartoon)$ to identify this tweet as multilingual. Once again, we are not interested in the accuracy provided by considering contextual information in language detectors for the following reasons. Firstly, the language ambiguity is not a problem, since we are attempting to detect whether a pair is multilingual and not exactly the languages of the pair of words. Notice that the language of the word *ver* by itself is ambiguous, since both Portuguese and Spanish contain this word. However, the model is only interested in knowing if the language of *ver* is different from the language of *cartoon*. Thus, as long as the probability of *ver* is low for English, the multilingual language detector would successfully identify this pair as multilingual. Secondly, even if a pair fails to be detected as multilingual, the model would still test all remaining pairs on whether these are parallel.

Word Pair Indexing

Traversing the whole Twitter corpus and computing Equation 3.5 for all pairs of words can be expensive, given that there are billions of tweets to process. To cope with this, we use an approach based on the work of [Uszkoreit et al., 2010b], where we first index the whole list of tweets. This index maps a word pair to a list of documents containing that word pair. Next, we traverse the index of word pairs and perform the detection using Equation 3.5, tagging the list of documents with that word pair as multilingual. Using this approach, we can avoid recomputing the same word pair more than once, if they occur in multiple tweets. Furthermore, we traverse the index so that word pairs that occur in more documents are processed first, and skip word pairs whose associated documents have already been classified as multilingual.

3.2.2 Location

The set of multilingual tweets T_{mult}, achieved after the filtering step, is then processed using the IDA model in order to find the optimal values for the variables $[p, q], [u, v], l, r$ that define the parallel segments and their languages. Then, each parallel sentence is extracted and placed in the set $D_{s,t} = (s_1, t_1), \ldots, (s_k, t_k)$, containing all parallel sentences extracted in the language pair s, t. It is important to

[2]Threshold determined empirically to obtain high recall.

mention that, while the variables l, r denote the languages of the left and right segments, respectively, in the set $D_{s,t}$, we place all parallel sentences that contain the language pair s, t, regardless of their order. Thus, we define the following insertion function:

$$\textbf{Ins}(d, [p, q], [u, v], l, r, D_{s,t}) = \begin{cases} D_{s,t} \cup (d_p^q, d_u^v) & l = s \wedge r = t \\ D_{s,t} \cup (d_u^v, d_p^q) & l = t \wedge r = s \end{cases} \tag{3.6}$$

where d_a^b denotes the segment corresponding to the indexes from a to b in the original tweet d. This function simply checks if the s and t correspond to the left l or right r segments in the detected parallel data, and places the appropriately aligned parallel segments.

Obviously, all sets D contain a considerable amount of non-parallel segments, as multilingual messages are not guaranteed to contain translated material. Furthermore, we must also consider errors from misalignments of the IDA model and misclassifications of the multilingual message detector. Thus, in order to identify messages that are actually parallel, an identification step is necessary.

3.2.3 Identification

Once the latent variables are converted into a sentence pair (s, t) in the previous step, many existing machine learning methods for detecting parallel data [Resnik and Smith, 2003, Munteanu and Marcu, 2005] can be applied, as this problem becomes a regular unstructured bitext identification problem. In our previous work [Ling et al., 2013b], we simply defined a threshold τ on the IDA model score, which was determined empirically. However, this approach is not optimal and not generalizable for other domains and language pairs. Thus, in our current work, we train a max entropy classifier model for each language pair, similar to that presented in [Munteanu and Marcu, 2005], that detects whether two segments are parallel in a given language pair. Training is performed on annotated data from tweets, where each tweet is annotated on whether there is parallel data in that language pair.

The automatic training involved the following set of features:

- **IDA Model Features** - These features correspond to the scores corresponding to each of the factors S_S, S_L and S_T. Each score is added as a separate feature so that they can be weighted separately.

- **User features** - While the background of the users that parallel post cannot be determined with absolute certainty, it is safe to assume that they are either bilingual speakers and translate their own messages, or hire translators to translate their posts. Thus, users that do not belong to these categories rarely post parallel messages, since they do not have the means, and likewise, users that are in these categories are likely to post a considerable amount of parallel posts. For instance, Sina Weibo users for Snoop Dogg and Paris Hilton mostly post in English-Mandarin. While the IDA model can align the parallel segments in most cases, some shorter more informal messages, such as *Ready to rock NYC.* - 准备好让纽约嗨起来。, tend to be harder to align and receive a lower score and not get classified as parallel. Yet, these messages tend to be more valuable, as they contain artifacts that our translation models cannot translate. Thus, it is desirable to consider the aggregate scores of the user posts as additional information for the classification problem. This is implemented by adding the average IDA model score from all posts from a given user as a feature.

- **Repetition features** - In informal domains, there are many terms that are not translated, such as hashtags (e.g. *#twitter*), at mentions (e.g. *NYC*), numbers and people's names. The presence of such repeated terms in the same tweet can be a strong cue for detecting translations. Hence, we define features that trigger if a given word type occurs in a pair within a tweet. The word types considered are hashtags, at mentions, numbers and words beginning with Capital letters.

- **Length feature** - It has been known that the length differences between parallel sentences can be modelled by a normal distribution [Gale and Church, 1991]. Thus, we used a training parallel data (used to train the alignment model) in the respective language pair to determine $(\tilde{\mu}, \tilde{\sigma}^2)$, which lets us calculate the likelihood of two hypothesized segments being parallel.

For each language pair s, t, we train a max-entropy classifier using these features using an annotated parallel set $D_{gold}(s, t)$. The method used to obtain these annotations is described in [Ling et al., 2014].

The quality of the classifier can be evaluated in terms of precision and recall. We count as a true positive (tp) if we correctly identify a parallel tweet, and as a false positive (fp) if we spuriously detect a parallel tweet. Finally, a true negative (tn) occurs when we correctly detect a non-parallel tweet, and a false negative (fn) if we miss a parallel tweet. Then, we set the precision as $\frac{tp}{tp+fp}$ and recall as $\frac{tp}{tp+fn}$.

Afterwards, F-measure is used to test the overall accuracy of the system in terms of precision and recall.

Depending on the nature of the task the extracted corpora is applied to, recall may be more important than precision or viceversa. For instance, as training data for MT models, recall tends to matter more, as models are robust to errors in the training data.

3.3 Experiments

In this section, we shall describe the experiments performed in this work. There are three sets of experiments that are be performed. We evaluate the parallel data extraction process intrinsically by testing each of the three steps described in Section 3.2, and extrinsically by testing its application to MT.

3.3.1 Setup

We consider the following languages: Arabic, German, English, French, Japanese, Korean, Mandarin, Portuguese, Russian and Spanish. From Sina Weibo, we focus on extracting sentence pairs involving Mandarin as one of the elements in the pair, as Mandarin is the main language in Sina Weibo. As for Twitter, we focus on finding sentence pairs involving English as one of the languages.

Tokenization

Tokenization converts a tweet into a set of tokens. Our tokenizer must consider a variety of languages and some common artifacts in the microblog domain. Below are general properties of our tokenizer:

- Sequences of Latin, Arabic, Cyrillic characters are separated into tokens using white spaces.
- Each Han, Hangul and Kana character is considered an independent token.
- Numbers are considered a token. Quantifiers, such as *$* and *kg*, are separated in a different token.

- Http links, hashtags and emoticons are standardized into _HTTP_, _HASH_ and _EMO_ tokens.
- Punctuation marks are considered separate tokens.
- Traditional characters are standardized into Simplified Mandarin characters[3].

One particular aspect in our tokenizer is that it stores the starting and ending offsets of the tweet from which each token was extracted. This is done so that after finding the parallel segments relative to the tokens, we can also use these offsets to recover the parallel segments in the non tokenized tweet.

Language Detection

A character-based language detector is required for the calculation of the language score in Section 3.1.1 and for the multilingual tweet detection in Section 3.2.1. This detector is trained on 112 languages, with the monolingual data extracted from Wikipedia dumps. While we do not extract parallel sentences for all the 112 languages, more information regarding existing languages allows the detector to estimate the language probabilities more accurately. As we are using a character trigram model, a large amount of data is not required to saturate the model probabilities. Thus, for each language, we extract all data from Wikipedia up to a limit of 100K documents in order to keep the model compact.

Translation Lexicons

The IDA model uses translation lexicons to determine the translation score, as described in Section 3.1.1, which are estimated using parallel corpora. More specifically, we use the aligner described in [Dyer et al., 2013] to obtain the bidirectional alignments from the parallel sentences. Afterwards, we intercept the bidirectional alignments to obtain sure alignment points. Finally, we prune the lexicon using significance pruning [Johnson and Martin, 2007] with the threshold $\alpha + \epsilon$ (as defined in that work). The intersection and the pruning are performed to reduce the size of the lexicon to make the look up faster. The breakdown of the different lexicons built is shown in Table 3.1.

[3]In general, Traditional and Simplified characters convey the simple meaning and normalizing them improves alignments by reducing sparcity.

Language Pair	Corpus	# sentence pairs
German–English	EUROPARL	1920K
Spanish–English	EUROPARL	1966K
French–English	EUROPARL	2007K
Portuguese–English	EUROPARL	1687K
Mandarin–English	FBIS	300K
Arabic–English	NIST	970K
Russian–English	Yandex	1000K
Korean–English	KAIST	60K
Korean–Mandarin	KAIST	60K
Japanese–English	Tatoeba	150K

Table 3.1: Lexicons built using parallel corpora.

3.3.2 Targeted Crawling

While the documents from Twitter were obtained using an existing dataset consisting of 1.6 billion tweets sampled from 2008 to 2013, originally from the *Gardenhose* stream[4].

As for the Weibo data, we describe the crawling method we used to crawl 3M parallel Mandarin-English sentence pairs from Sina Weibo within 6 months. The main problem we had to address is related to the fact that a uniform crawl over 1.6 billion tweets from Twitter only yields approximately 300K English-Mandarin sentence pairs. However, due to the rate limiting[5] established by Sina Weibo's API, we were only able to send 150 requests per hour. Each request could fetch up to 100 posts from a user, and subsequent sets of 100 posts request additional API calls. This means that crawling 1.6 billion tweets is beyond our capabilities. In fact, we were only able to crawl approximately 90 million tweets in 6 months.

Thus, we wished to optimize the parallel tweets we obtained from each API call. This was done from the observation that users that frequently, post in parallel are likely to post more parallel tweets in the future. Thus, we spent only one request per user to obtain his/her first 100 messages, and ran the IDA model on those messages and estimate the number of parallel messages within those 100 messages[6]. If the

[4]obtained from http://www.ark.cs.cmu.edu/tweets/

[5]http://open.weibo.com/wiki/API文档/en

[6]The identification of parallel messages was performed by setting a threshold on the IDA score, since we did not have a max entropy classifier at the time. The details of the old model is defined in [Ling et al., 2013b] It is not possible to repeat this method under the same conditions due to the dynamic nature of microblogs.

number of automatically identified parallel messages within those 100 tweets was higher than 10, we set that user as active. We obtain all messages from active users and periodically check if new messages had been posted. An active user remains active while the number of posts he/she possesses that are identified as parallel constitute at least 10% of all his/her posts.

In summary, the crawler operates as follows:

1. Pick a random user and crawl 100 posts all from followers and followees.
2. For all active users that have not been updated within the last week, check and crawl their new posts.
3. Repeat from 1.

During the one hour downtime from exhausting the 150 requests in any step, we ran the following operations:

1. Run the IDA model for users with unprocessed tweets.
2. Set the user as active if more than 10% of his/her tweets are parallel, or inactive otherwise.

This is done separately in order to spend the 150 requests when available, as unspent requests are lost after an hour.

3.3.3 Building Gold Standards

To train and test the classifier described in Section 3.2.3, and perform MT experiments, a corpus of annotated tweets is needed for different language pairs. Table 3.2 summarizes the obtained corpora for different domains (Twitter or Sina Weibo) and language pairs. The number of tweets that were annotated are quantified in the *#Annotated Tweets* column. The number of tweets that contained parallel data is provided in the *#Parallel Sentences* column. For Sina Weibo, the annotations were performed manually by an expert Mandarin speaker, whereas the Twitter datasets were crowdsourced in Mechanical Turk. Our method to crowdsource such data is described in [Ling et al., 2014]. Finally, we also illustrate the average size of the English and non-English sides of the extracted data based on the number of tokens

Source	Language Pair	Method	#Annotated Tweets	#Parallel Sentences	Average Size (English)	Average Size (Foreign)
Weibo	English–Mandarin	Expert	4347	2581	18.09	28.45
Twitter	English–Mandarin	Crowdsourcing	2688	1302	8.76	14.03
Twitter	English–Arabic	Crowdsourcing	2520	1758	8.32	6.84
Twitter	English–Russian	Crowdsourcing	5082	1631	7.11	6.19
Twitter	English–Korean	Crowdsourcing	3906	1116	7.10	15.67
Twitter	English–Japanese	Crowdsourcing	2583	1155	10.22	19.07
Twitter	English–Portuguese	Crowdsourcing	2583	542	8.50	9.02
Twitter	English–Spanish	Crowdsourcing	2541	722	7.09	7.28
Twitter	English–French	Crowdsourcing	2856	594	8.46	8.95
Twitter	English–German	Crowdsourcing	2772	909	8.94	7.31

Table 3.2: Description of the annotated data. Columns correspond to the the number of annotated tweets, the number of parallel sentences obtained, the method used to obtain the annotations and the average number of words in the English and non-English sides of the parallel sentences.

generated by our tokenizer in the last two columns. We observe that the number of words in Twitter datasets are smaller than those in Weibo, which can be explained by the fact that posts in Twitter are limited to 140 characters as opposed to Sina Weibo, where posts are not bound by such restriction.

It is important to keep in mind that these numbers are not fully representative of the Twitter data as a whole, as we are filtering out monolingual tweets prior to the annotation. Thus, we cannot draw conclusions about the ratio between parallel and non-parallel data in Twitter. For instance, the ratio between parallel and non-parallel tweets for Arabic-English in Twitter is 2:1 in the annotated datasets, which is definitely not the case for uniformly extracted datasets[7]. However, performing the annotation for an uniformly extracted dataset is problematic for two reasons. Firstly, a huge annotation effort would be required to find a significant amount of tweets containing translations, since most tweets do not contain translations. Secondly, training the classifier on such an unbalanced dataset would bias the classifier towards the negative case, as the majority of the samples belong in this category, which is definitely not desired.

[7]We obtain this ratio as we are filtering out monolingual tweets, and we are removing samples that do not contain alignment points

3.3.4 Parallel Data Extraction Experiments

We report on the experiments performed in each of the stages of the parallel data extraction process described in Section 3.2.

Filtering

The filtering step (described in Section 3.2.1) attempts to filter out monolingual tweets, since these definitely do not contain parallel data. Ideally, we would uniformly sample tweets and annotate them on whether they are multilingual. However, this would require an extraordinary amount of effort to obtain a sample with a large number of multilingual tweets, as most tweets are monolingual. Thus, we use an manually annotated dataset on tweets from Twitter, where each word in the tweet was annotated with its language. On this dataset, there are 773 annotated samples from Twitter. We filter these so that all tweets contain words from at least two different languages, resulting in 554 multilingual tweets. We also build two other datasets consisting of tweets that contain the languages we are considering for parallel data extraction with 291 tweets, and another consisting of only languages that use the Latin alphabet[8] with 222 tweets. To find monolingual tweets, we sample tweets uniformly until we find 2000 tweets that are monolingual from Twitter, which is feasible since most tweets are in this category. The goal of this experiment is to measure how many multilingual tweets are lost due to the filtering process. Thus, these are then injected into the Twitter dataset. Then, we compute the accuracy on the multilingual datasets based on the percentage of the injected tweets that were retrieved and the accuracy on monolingual datasets by computing the percentage of monolingual tweets that were filtered using different thresholds for Equation 3.5.

Results are shown in Figure 3.5, where we can observe that by simply by removing the top 90% of word pairs, we can remove 67.8% of the monolingual tweets at the cost of losing 10-15% of the multilingual tweets at threshold 0. Obviously, when we start considering the probabilities of the word pairs, we observe a substantial improvement at the detection of monolingual tweets, at the cost of multilingual ones. As we raise the threshold, more multilingual tweets are discarded in order to remove more monolingual tweets. We also observe that we obtain similar results for tweets containing only Latin languages, such as English and Portuguese, as they share many orthographic similarities. Finally, results for the language we use

[8]English, Spanish, French, German and Czech.

are slightly higher, as they contain many language pairs, such as Mandarin-English, where a different set of characters are used.

In our work, we set the threshold to 95%, in order to maximize the number of monolingual tweets that are removed. While this filters out more potentially multilingual tweets, it substantially reduces the number of tweets that need to be processed. Furthermore, a large portion of the misclassifications for multilingual tweets are the result of tweets that only contain one or two words in a different language, such as a person's name, and these tweets are generally devoid of bitext.

Location Results

To test the quality of the location of the parallel segments, we compare the automatic segment boundaries with the human annotations using the S_{IDA} metric defined in 3.1.5, which measures the overlap between the proposed and the reference boundaries. Results for different language pairs and domains can be found in Table 3.3, where average overlap score S_{seg} for the English and Foreign segments are shown in Columns *English Overlap* and *Foreign Overlap*, respectively. The S_{IDA} score obtained as a harmonic mean between the previous scores is shown in the S_{IDA} column.

From these results we can see that for Sina Weibo's English Mandarin corpus, the results are significantly higher than those in Twitter. One explanation for this is the fact that parallel segments found in Weibo are longer. This allows the alignment model to find more word alignments, which can be used to find better boundaries for the parallel spans.

We can also observe that results tend to be higher on the language-pairs, where we used more and higher quality training corpora to build the lexical translation table. For instance, the English-Korean and English-Japanese, where the parallel corpora used consisted of 60K and 150K sentence pairs, the results are evidently worse compared to the English-Arabic and English-Russian results, where approximately 1 million sentence pairs were used.

Identification Results

The aim of the identification task is to determine if a given tweet contains parallel data. We used the datasets described in Section 3.3.3, and these were evenly divided into training and test sets. The maximum entropy classifier was trained

Source	Language Pair	English Overlap	Foreign Overlap	S_{IDA}
Weibo	English–Mandarin	0.848	0.891	0.859
Twitter	English–Mandarin	0.743	0.779	0.760
Twitter	English–Arabic	0.801	0.794	0.771
Twitter	English–Russian	0.793	0.818	0.778
Twitter	English–Korean	0.737	0.744	0.706
Twitter	English–Japanese	0.695	0.786	0.704
Twitter	English–Portuguese	0.759	0.781	0.770
Twitter	English–Spanish	0.798	0.795	0.796
Twitter	English–French	0.836	0.809	0.822
Twitter	English–German	0.734	0.718	0.726

Table 3.3: Results for the location of the parallel segments over different datasets. The *English Overlap* and *Foreign Overlap* columns illustrate the average of the overlaps of the automatically extracted segments for the English and Foreign segments, respectively. The final score is computed as the harmonic mean between the two previous overlaps, which is shown in the S_{IDA} column.

using Weka [Hall et al., 2009], which maximizes the weighted F-measure of the positive and negative samples. We calculate the precision, recall (on positive labels) and accuracy at increasing intervals of 10% of the top scoring samples. This score is calculated as the probability of the positive label given.

Results for the English-Mandarin language pair for both the Twitter and Sina Weibo domains using the full set of features are presented in Figure 3.6. We can observe that Weibo contains a larger amount of parallel tweets as the precision for the x-axis at 1 (where all the samples are considered as parallel) is only 46% for Twitter, compared to 61% in Weibo. This is because the Twitter dataset we used to extract the parallel data from was extracted uniformly, while the Sina Weibo tweets were crawled using the algorithm described in Section 3.3.2, which attempts to maximize the amount of tweets containing translations using heuristics. We can also observe that results are significantly better for Weibo, as the precision curve for the Weibo dataset is always higher than the one for the Twitter dataset. This shows that while the method is the same, our method's performance is affected by the dataset itself. One reason is the fact that messages are longer in Sina Weibo, so they contain more information that can be used to determine whether translations exist within. The other reason is the fact that orthographic innovation for English tends to occur more often in Twitter, as English is not the mainstream language in Weibo.

While it is not the goal of this work to exhaustively engineer features to maximize

Source	Language Pair	IDA	+User	+Length	+Repetition
Weibo	English–Mandarin	0.781	0.814	0.839	**0.849**
Twitter	English–Mandarin	0.599	0.598	0.603	**0.652**
Twitter	English–Arabic	0.721	0.721	0.725	**0.763**
Twitter	English–Russian	0.692	0.705	0.706	**0.729**
Twitter	English–Korean	0.635	0.650	0.650	**0.655**
Twitter	English–Japanese	0.570	0.566	0.569	**0.579**
Twitter	English–Portuguese	0.859	**0.860**	0.857	0.858
Twitter	English–Spanish	0.841	0.849	0.850	**0.850**
Twitter	English–French	0.886	0.886	0.885	**0.888**
Twitter	English–German	0.789	0.787	0.794	**0.798**

Table 3.4: Results for the parallel data identification task over different datasets. The columns present the identification results using a incremental set of features. Each cell contains the F-measure using a given dataset and set of features.

the results for this task, we wish to show that additional features can be used to improve the quality of the classifier, some even containing Twitter or Weibo specific attributes, such as the meta-information regarding the user that posted each tweet. To do this, we trained the max entropy classifier using an increasingly larger set of features and present the weighted average of the F-measure for positive and negative labels.

Results for different language pairs are shown in Table 3.4, where we can observe that results can be improved by adding more features, similar to previous work [Resnik and Smith, 2003, Munteanu and Marcu, 2005]. The microblog specific *User features* seem to work best in Sina Weibo, as the crawling methodology we used obtains a large amount of posts from the same user. On the other hand, the Twitter dataset was obtained from a uniform crawl where we had less tweets from each user, which is why smaller improvements were observed. We can also observe that other more general features (Length, Repetition and Language) also improve the classifier slightly.

The quality of the results of the identification task is strongly correlated to the quality of the location task, as the identification task depends on the quality of the automatically detected parallel segments. For instance, we can observe better results for Arabic-English, which also obtained a high S_{IDA} in Table 3.3. Furthermore, similar language pairs, such as English-Spanish and English-French, tend to be better aligned, and obtain higher scores. In these cases, additional features are less beneficial for the classifier.

Topic	Most probable words in topic
1 (Dating)	love time girl live mv back word night rt wanna
2 (Entertainment)	news video follow pong image text great day today fans
3 (Music)	cr day tour cn url amazon music full concert alive
4 (Religion)	man god good love life heart would give make lord
5 (Nightlife)	cn url beijing shanqi party adj club dj beijiner vt
6 (Chinese News)	china chinese year people world beijing years passion country government
7 (Fashion)	street fashion fall style photo men model vogue spring magazine

Table 3.5: Most probable words inferred using LDA in several topics from the parallel data extracted from Weibo. Topic labels (in parentheses) were created manually for illustration purposes.

Data Representation

To give an intuition about the contents of the parallel data we are finding, we look at the distribution over topics of the parallel dataset inferred by LDA [Blei et al., 2003]. To do so, we grouped the Weibo filtered tweets by user, and ran LDA over the predicted English segments, with 12 topics. The 7 most interpretable topics are shown in Table 3.5. We see that the data contains a variety of topics, both formal (Chinese news, religion) and informal (entertainment, music). We do not provide a breakdown of topics for the Twitter datasets, as the datasets were significantly more reduced and only 1 or 2 posts per user are available as the twitter dataset was crawled uniformly.

3.3.5 Machine Translation Experiments

In order to measure the impact of the extracted corpora, we performed an extrinsic experiment where we used the extracted data as training parallel sentences for existing MT systems. We first performed an extensive test on the English-Mandarin language pair, where we show that the extracted corpus contributes to improve the state-of-the-art results in Twitter, Sina Weibo and also more formal domains. Then, we showed that these results generalize for other language pairs.

To accurately quantify the effect of our corpora in translation, all experiments were conducted using fixed development and test sets for each domain as well as the same system setup but with different training sets. That is, we wish to show that even with a small sample of in-domain parallel data for tuning and testing, and

a large amount of monolingual data, the automatically extracted parallel data can improve the system significantly.

Datasets

We report on machine translation experiments using our harvested data in three domains: edited news, Twitter and Sina Weibo.

- **News translation** - For the news test, we created a new test set from a crawl of the Mandarin-English documents on the Project Syndicate website[9], which contains news commentary articles. We chose to use this data set, rather than more standard NIST test sets to ensure that we had recent documents in the test set (the most recent NIST test sets contain documents published in 2007, well before our microblog data was created). We extracted 1386 parallel sentences for tuning and another 1386 sentences for testing, from the manually alignment segments. For this test set, we used 8 million sentences from the full NIST parallel dataset as the language model training data. We shall call this test set **Syndicate**.

- **Sina Weibo translation** - The Sina Weibo corpus was created by using the gold standard annotations described in Section 3.3.3. This contained 2581 parallel sentences, where 1290 pairs were used for tuning and the other 1291 pairs were used for testing. Naturally, we removed all these instances from the training data. We refer to this test set as **Weibo**[10]. The language model used in this work was built using 10 million tweets from Sina Weibo for Mandarin. As for English, we used 10 million tweets from Twitter.

- **Twitter translation** - The Twitter datasets were built using a similar methodology, where we use the gold standard annotations to create a held-out dataset of parallel sentences. In all cases, we split the held-out dataset evenly to obtain the tuning and testing datasets. To build the English and Mandarin language models, we used the same data as the Sina Weibo dataset.

[9] http://www.project-syndicate.org/ during the year 2012

[10] We acknowledge that self-translated messages are probably not a typically representative sample of all microblog messages. However, we do not have the resources to produce a carefully curated test set with a more broadly representative distribution. Still, we believe these results are informative as long as this is kept in mind.

Training Data

We report results on these test sets using different training data. First, we use the FBIS dataset which contains 300K high quality sentence pairs, mostly in the broadcast news domain. Second, we use the full 2012 NIST Mandarin-English dataset (approximately 8M sentence pairs, including FBIS). Finally, we use our crawled data from Sina Weibo (referred as Weibo) and those extracted from Twitter (referred as Twitter) by themselves but also combined with the two previous training sets. The max-entropy classifier for detecting parallel data was tuned for a 50% precision, where 118 parallel sentences were extracted.

Setup

We use the Moses phrase-based MT system with standard features [Koehn et al., 2003]. For reordering, we use the MSD reordering model [Axelrod et al., 2005]. As the language model, we use a 5-gram model with Kneser-Ney smoothing. The weights were tuned using MERT [Och, 2003]. Results are presented with BLEU-4 [Papineni et al., 2002].

Results

The BLEU scores for the different parallel corpora are shown in Table 3.6 and the top 10 out-of-vocabulary (OOV) words for each dataset are shown in Table 3.7. Results for the microblog test sets (Weibo and Twitter) suggest that considerable improvements can be obtained by using the crawled corpora for this domain.

The most notable result can be observed by contrasting the obtained BLEU scores on the Weibo test set with the NIST and Weibo training corpus (NIST and Weibo rows), where relative improvements of over 200% in terms of BLEU can be observed. While this is a promising result, it is important to consider the fact that we are drawing training and test samples from the same domain, which naturally leads to better results than using training corpora that was drawn from other domains. However, a strong indication that this is not the case is the fact that similar results can be obtained for the Twitter test set, where we can observe a BLEU improvement from 9.55, using the NIST corpus, to 23.57, using the Weibo corpus. This shows that the extracted corpus contains many translated elements that are representative of the microblogging domain that are not found in publicly available

	Syndicate		Weibo		Twitter	
	ZH-EN	EN-ZH	ZH-EN	EN-ZH	ZH-EN	EN-ZH
FBIS	8.86	19.69	9.24	13.60	8.44	13.04
NIST	**10.87**	**22.32**	10.76	14.89	9.55	13.97
Twitter	2.90	6.59	10.53	11.42	12.67	15.43
Weibo	9.78	20.72	**33.32**	**34.48**	**23.57**	**25.01**
NIST+Twitter	12.43	23.81	15.29	16.72	15.83	19.29
NIST+Weibo	**13.11**	**24.60**	**33.01**	**34.33**	**23.35**	**27.07**

Table 3.6: BLEU scores for different datasets (**Syndicate**, **Weibo** and **Twitter**) in different translation directions (EN-ZH and ZH-EN), broken with different training corpora (top to bottom). The top four rows depict experiments using a single datasets, while the bottom two rows combine both in-domain and out-of-domain datasets.

corpora, such as NIST[11]. Examples include translations for newer terms, such as *iTunes*, as we can observe in Table 3.7. While there is no guarantee that all the terms are translated correctly, it is a promising result that using the Weibo corpus, we can find translations for all words that occur more than once in the Twitter dataset, which can be seen in Table 3.7. On the other hand, the NIST and FBIS dataset do not possess frequent terms, such as *wanna*, *lol* and *omg* and also newer terms like *kpop* (Korean pop music) in their translation inventory. Furthermore, some important terms that are not captured by looking for OOV words, such as abbreviations, like *u* and *4* are found in the NIST dataset, but can be used with different meanings in microblogs. We also found examples of microblog terms that are not addressed by the NIST dataset, and are learnt from the Weibo training corpus. First, we observe the character *jiǒng*, which generally means *embarrassed* in informal speech. The term 粉丝(*fěn sī*), is a phonetic translation of the English term *fans*. This is translated incorrectly using the FBIS dataset as *powder silk*, which is its literal character level translation. Finally, the term *diǎo sī* is very common in Chinese microblogs. While there is no direct translation for this term, it is generally translated based on context into *loser* or *sucker* in our extracted parallel sentence pairs from Weibo.

If we observe the results using the Twitter training set (Twitter row), we can observe that improvements are much more reduced. In fact, for the Weibo test set, the obtained results are lower than using the NIST dataset. This can be explained by the lexical gaps caused by the small size of the Twitter training set (117K sentence

[11]as results different radically between datasets we did not run significance tests

pairs). However, we can see that a considerable improvement can be obtained by combining these datasets (NIST+Twitter row), which suggests that a meaningful translation inventory can be obtained from this training set for the translation of microblog data.

As for the Syndicate test set (Syndicate row), the NIST and FBIS datasets perform better than our extracted parallel data as they are much more representative of this domain. Yet, we can observe that training with the Weibo corpus still obtains a similar result compared to the NIST and FBIS datasets, due to the fact that many news events are posted on Weibo, and contain the formal language present in NIST and FBIS. Furthermore combining datasets leads to improvements in BLEU. Error analysis indicates that one major factor is that names from current events, such as *Romney* and *Wikileaks* do not occur in the older NIST and FBIS datasets, but are represented in the Weibo dataset.

We did not show results for combining the three datasets (NIST, Weibo and Twitter) as the Twitter dataset is substantially smaller and results are not significantly different than combining the NIST and Weibo datasets.

Experiments for Other Language Pairs

To further validate the usefulness of this method, we perform a similar test for other language pairs. We perform the same test on the other test data we obtained from Twitter, as the Weibo dataset mainly contained English-Mandarin sentence pairs, as the crawler was parameterized to find users that post in this language pair. The Twitter dataset was crawled uniformly, which lowered the amount of training data that can be found within this dataset. We did not perform experiments for language pairs where the number of extracted training sentences is particularly small (3K for Korean-English). For other language pairs, we checked whether we can improve translation results by adding the extracted corpora into formal datasets.

Once again, the crowdsourced corpus in Table 3.2 is divided evenly in development and test sets for MT. The in-domain training corpus for Twitter is extracted automatically by optimizing the threshold for 50% precision. The size of these corpora varies from 10K to 150K. As out-of-domain training data, we use the data from which the lexicons (Table 3.1) were built. As monolingual data for English, we use the same 10M tweets from Twitter as in the previous experiment, and we always translate from the foreign language into English.

Results are shown in Table 3.8, where we can observe that on average, the usage

of the in-domain data results in better translation quality as expected. Furthermore, combining both the in-domain and out-of-domain data improves the results further, similarly to the Mandarin-English experiments.

In most cases, the significant improvements are led by the inexistence of frequently used variations for fundamental function words in each language, which do not occur in formal domains. For instance, in the Portuguese-English language pair, the Portuguese terms *muito* and *para*, which mean *very* and *for* in English, are generally written as *mto* and *pra* in Twitter. As the in-domain corpus that is used are relatively small, adding a large amount of out-of-domain bitext yields large improvements in BLEU.

Translation of Online Terms

In order to provide insight of the distintive translations of online terms that are learned in our extracted data, we manually selected some examples of parallel sentences containing terms that are generally not found in other media. These are shown in Figure 3.7.

The first row provides a sentence pair with the term *diaosi*, which is a popular buzzword among Chinese communities and is used to describe a class of underprivileged men, lacking certain desirable features (looks, social status, wealth etc...). Generally, these terms are difficult to translate as no counterpart exist in other languages, and must be addressed depending on context. In this case, the term is translated into *loser*, which is an acceptable translation due to the lack of better English terms.

The second row shows an example where multiple lexical variants in English are used. In this example, the Mandarin translation is translated formally without the stylistic features present in the English sentence. However, in some parallel sentences, these properties are preserved during translation. For instance, the term *toooooo* in the third row corresponds to the word *too* with extra *o*'s to further emphasize the following adjective *thick*. Likewise, in the Mandarin sentence, it is translated into 太*XX*, which is composed by the character 太, an equivalent for the word *too* in English, and the string *XX*, which adds a similar connotation of the extra *o*'s in the English sentence.

Finally, in the fourth and fifth rows, we observe some examples of Chinglish, where new English words are constructed from Mandarin terms. In the fourth row, the term 牛逼, which is an adjective to describe that something is great, is translated

into *niubility*, which is constructed by taking the Pinyin representation for the term牛逼(*niubi*) and adding the English suffix *lity*. More examples are given in the firth row, where other similar terms (niubility, zhuangbility, shability and erbility) are also translated. In this case, we can also observe examples of abbreviations in the Mandaring translations, where the character 逼 is replaced by the letter *B*, which is the first letter in the Pinyin representation of the character.

It is also important to mention that mainstream translation systems, such as Google Translate[12] and Bing Translator[13] fail to translate these examples. For instance, the term *diaosi* in the first row, is translated character-by-character, which leads to a translation with a very different meaning. Both abbreviations in English (e.g. *imma*) and in Mandarin (e.g. 逼*B*) also tend to be mistranslated. These are either incorrectly translated phonetically for English words, or character-by-character for Mandarin terms. Finally, we also tried translating Chinglish terms using online systems and found that these are either not translated as OOV words, or translated incorrectly.

The existence of these terms in microblogs and other informal domains justifies the need for the methods to find parallel data where the translations for those terms can be learnt. As such, we believe that the method we propose in this work has the potential to substantially improve the state-of-the-art MT systems in these domains.

[12]https://translate.google.com
[13]http://www.bing.com/translator/

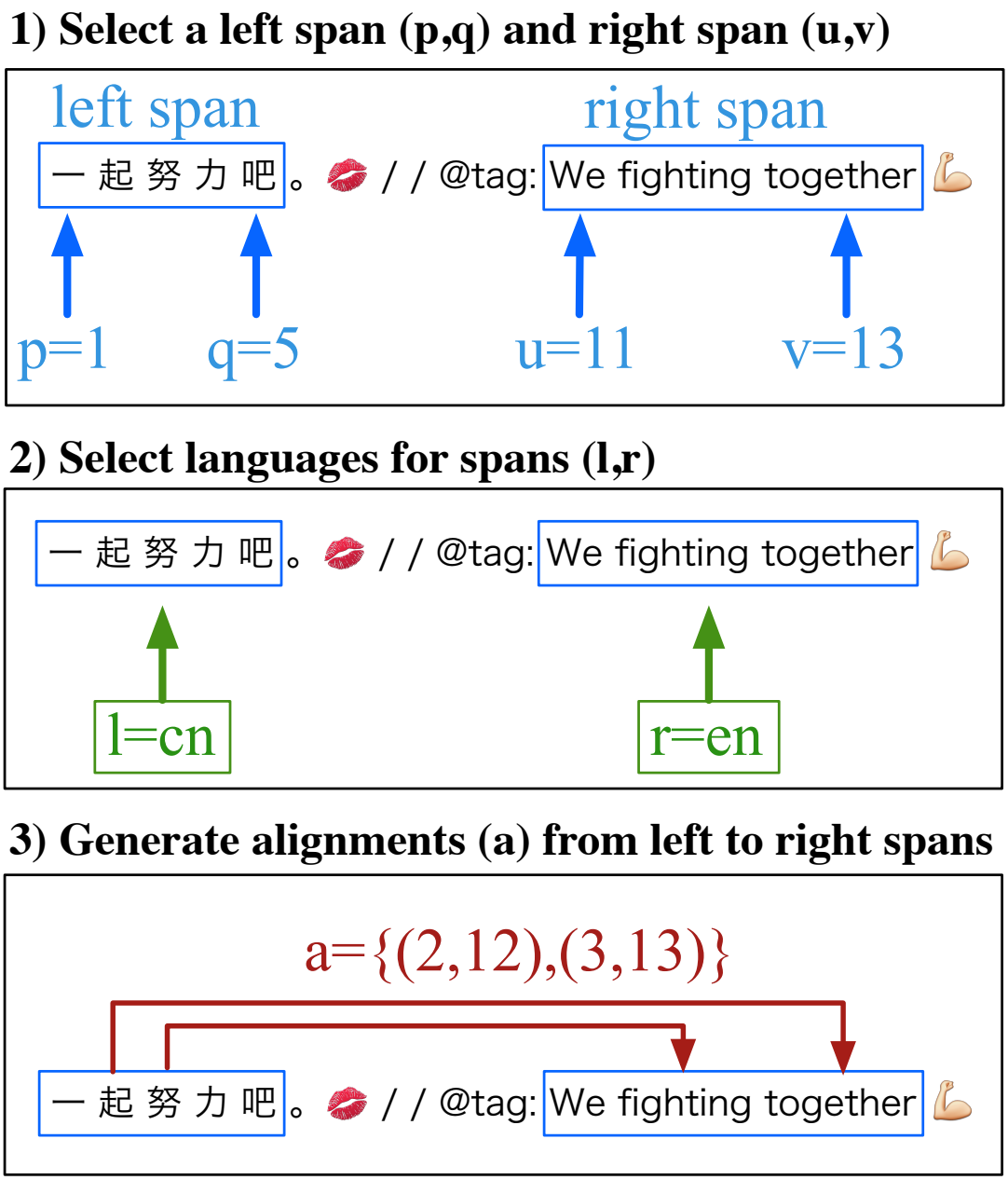

Figure 3.3: Illustration of each of the model parameters. The top box shows a potential pair of parallel segments p, q, u, v. The box in the middle represents a possible pair of languages l, r and the bottom box illustrates the word alignments between these segments.

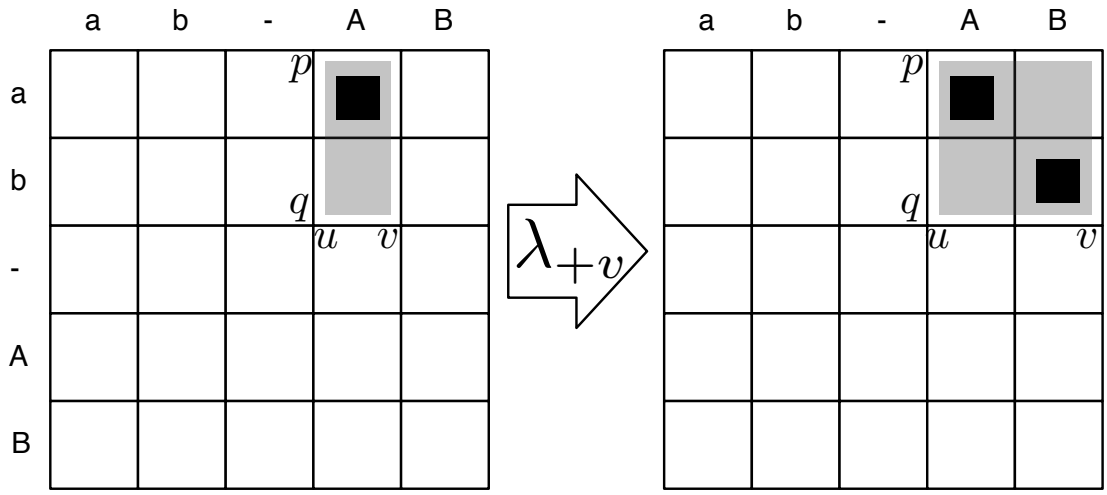

Figure 3.4: Illustration of the λ_{+v} operator. The light gray boxes show the parallel span and the dark boxes show the span's Viterbi alignment. In this example, the parallel message contains a "translation" of a b to A B.

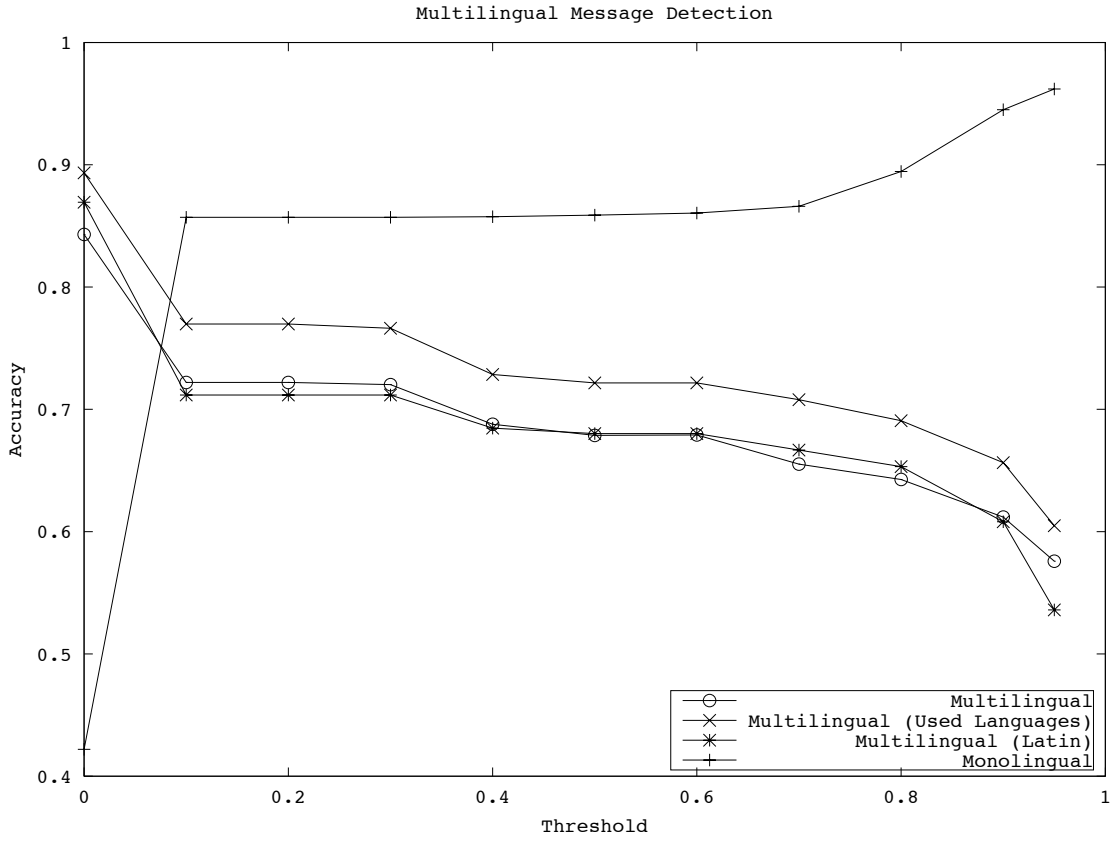

Figure 3.5: Results for the language based filtering task. The X-axis represents the threshold for Equation 3.5. The “Multilingual" line represents the percentage of multilingual tweets that are kept, while the “Monolingual" line represents the percentage of monolingual tweets that are discarded. Thus, we wish to maximize the Monolingual score and minimize the Multilingual score. For contrast, we also show the same scores using the languages that we are extracting parallel data for in the “Multilingual (Used Languages)" line, and those that are in Romance languages in the “Multilingual (Latin)" line.

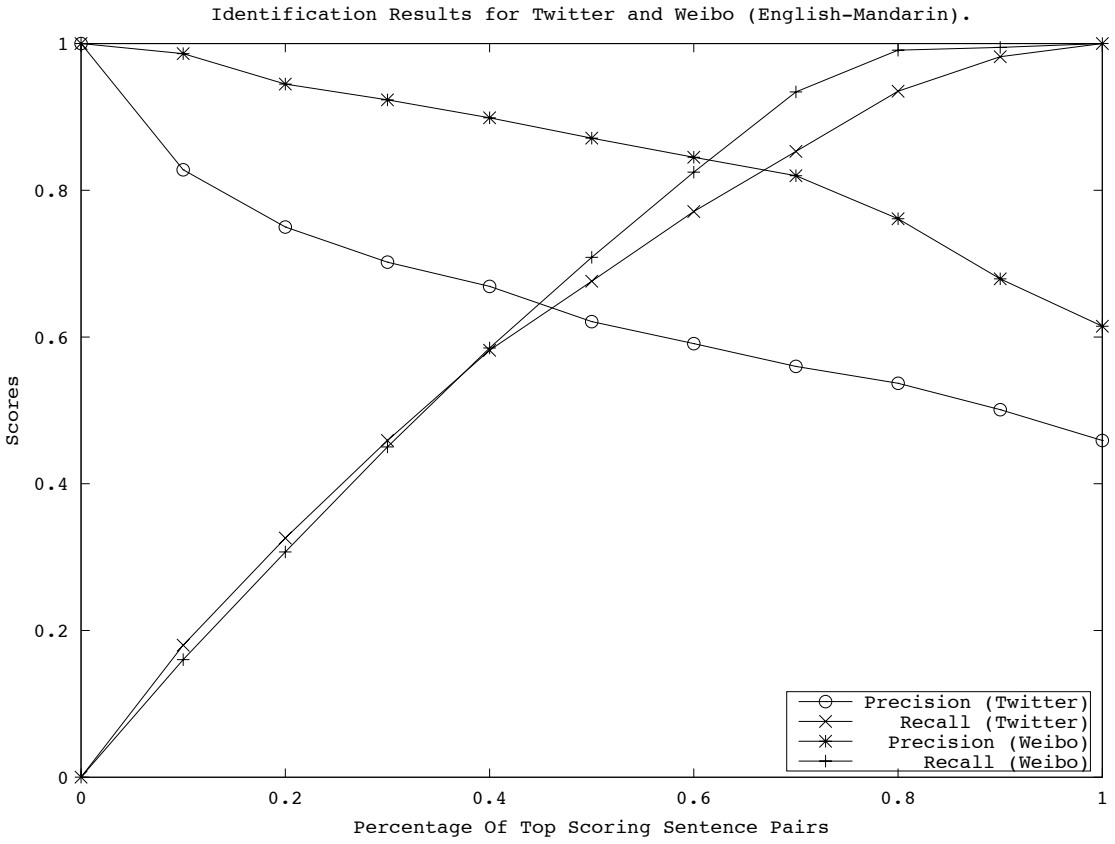

Figure 3.6: Precision and recall curves an increasingly larger set of top scoring sentence pairs for the Weibo and Twitter datasets for the English-Mandarin language pair. The precision and recall scores are quantified in the Y-axis, and the percentage of samples labelled as positive is represented in the X-axis.

Syndicate (test)			
FBIS	NIST	Weibo	Twitter
obama (83)	barack (59)	democracies (15)	cambridge (80)
barack (59)	namo (6)	imbalances (13)	debt (68)
princeton (40)	mitt (6)	mahmoud (12)	2008 (55)
ecb (8)	guant (6)	millennium (9)	monetary (52)
bernanke (8)	fairtrade (6)	regimes (8)	economies (47)
romney (7)	hollande (5)	wolfowitz (7)	paris (45)
gaddafi (7)	wikileaks (4)	revolutions (7)	increasingly (41)
merkel (7)	wilders (3)	qaddafi (7)	princeton (40)
fats (7)	rant (3)	geopolitical (7)	recession (39)
dialogue (7)	esm (3)	genome (7)	fiscal (38)
Weibo (test)			
FBIS	NIST	Weibo	Twitter
2012 (24)	showstudio (9)	submissions (4)	alanis(15)
alanis (15)	crue (9)	ivillage (4)	mexico (14)
crue (9)	overexposed (8)	scola (3)	ego (12)
showstudio (9)	tweetmeian (5)	rbst (3)	kelly (11)
overexposed (8)	tvd (5)	curitiba (3)	showstudio (10)
itunes (8)	iheartradio (5)	zeman (2)	awakening (9)
havoc (8)	xoxo (4)	yaptv (2)	milan (8)
sammy (6)	snoop (4)	witnessing (2)	crue (8)
obama (6)	shinoda (4)	whoohooo (2)	wit (7)
lol (6)	scrapbook (4)	wbr (2)	trespassing (7)
Twitter (test)			
FBIS	NIST	Weibo	Twitter
twitter (5)	twitter (5)	zzzzzzzzz (1)	submit (3)
kyuhyun (5)	kyuhyun (5)	unobstructive (1)	fin (3)
siwon (4)	siwon (4)	totalitarian (1)	behavior (3)
lol (4)	oppa (3)	telus (1)	artificial (3)
oppa (3)	didn (3)	takamine (1)	scribd (2)
haha (3)	scribd (2)	spansh (1)	owes (2)
gd (3)	omg (2)	spaciousness (1)	net (2)
didn (3)	kpop (2)	soshi (1)	kindness (2)
wanna (2)	facebook (2)	snowkyu (1)	february (2)
tid (2)	exo (2)	wukan (1)	ego(2)

Table 3.7: The most frequent out-of-vocabulary (OOV) words and their counts for the three English-source test sets (Syndicate, Weibo and Twitter) with four different training sets (FBIS, NIST, Weibo and Twitter). We did not consider http links, hash tags and at mentions, in this list as these are generally not translated.

Source Language	Arabic	Spanish	French	Japanese	Russian
Out-of-domain Size	970K	1966K	2007K	150K	1000K
In-domain Size	138K	36K	12K	31K	11K
Out-of-domain	12.76	11.11	16.01	6.72	28.05
+In-domain	29.27	27.00	25.89	11.23	31.53

Table 3.8: Translation experiment results for different language pairs on the Twitter data. Each column shows the MT results for a given source language into English. The *Out-of-domain Size* and *In-domain Size* rows represent the number of sentence pairs in the in-domain and out-of-domain training sets. The *Out-of-domain* and *+In-domain* rows show the BLEU scores for a setup where only the out-of-domain data was used and the same setup after adding the in-domain dataset, respectively.

1	It's said all the **losers** around the country have been downloading a 982.77M video over the past two or three days. The latest greetings are: "have you done?" and the answer is: "Not yet! It got stuck at 83%!" Quit playing dumb. You know what I'm talking about	据说这两天全国的**屌丝**都在下载一个大小为982.77MB的文件；最新的见面问候语是："下完了没？"回答："还早呢，卡在83%不动了"别装、你们都懂的
2	To DanielVeuleman **yea** **iknw** **imma** work on that	对DanielVeuleman说，**是的**，我**知道**，**我正在**向那方面努力
3	People say real men can read women like a book. But don't you think this book is way **toooooo** thick?	他们说真男人懂得像读书一样去读懂女人，可是这书也**太XX**厚了吧
4	What's **Niubility**? **Niubility** is being able to get 1 million followers even you post nothing and follow nobody on Weibo. We call it Han-style Niubility!	啥是**牛逼**？**牛逼**就是你在微博上一条消息未发，一个人不关注，却能得到百万粉丝。我们称之为韩式**牛逼**。
5	Many people think they are full of **niubility** and like to play **zhuangbility**, which only reflect their **shability** and **erbility**.	很多人觉得自己很**牛B**特喜欢**装B**，其实就是个**傻B**和**2B**！

Figure 3.7: Examples of parallel sentences extracted from Sina Weibo. These examples were hand extracted because they contain elements that are characteristic of this domain.

Chapter 4

Normalization Using Paraphrases

In this Chapter, we propose an application of the parallel corpus using the dataset we obtained. The task is to build a normalization model capable of standardizing the nonstandard language in microblogs. This can be used as a pre-processing step prior to translation, but also other NLP tasks. More concretely, we introduce a data-driven approach to learning normalization rules by conceiving of normalization as a kind of paraphrasing and taking inspiration from the bilingual pivot approach to paraphrase detection [Bannard and Callison-Burch, 2005] and the observation that translation is an inherently "simplifying" process [Laviosa, 1998, Volansky et al., 2013]. We start from extracted parallel corpus of microblog messages consisting of English paired with several other languages, we use standard web machine translation systems to *re-translate* the non-English segment, producing ⟨English original, English MT⟩ pairs. Then, a normalization model is built to map the original message to the produced MT outputs. [1]

4.1 Obtaining Normalization Examples

We want to treat normalization as a supervised learning problem akin to machine translation, and to do so, we need to obtain pairs of microblog posts and their normalized forms. While it would be possible to ask annotators to create such a corpus, it would be quite expensive to obtain large numbers of examples. In this section, we propose a method for creating normalization examples without any

[1] This chapter contains the work published in Ling et al. [2013a]

Table 4.1: Translations of Chinese original post to English using web-based service.

orig.	To DanielVeuleman yea iknw imma work on that
orig.	对DanielVeuleman说，是的，我知道， 我正在向那方面努力
MT[1]	Right DanielVeuleman say, yes, I know, I'm Xiangna efforts
MT[2]	DanielVeuleman said, Yes, I know, I'm that hard
MT[3]	Said to DanielVeuleman, yes, I know, I'm to that effort

human annotation, by leveraging existing tools and data resources.

The English example sentence in Table 2.1 was selected from the corpus extracted in Chapter 3. Row 2 of Table 4.1 shows the Mandarin self-translation from the corpus. The key observation is what happens when we *automatically* translate the Mandarin version back into English. Rows 3–5 shows automatic translations from three standard web MT engines. While not perfect, the translations contain several correctly normalized subphrases. We used such *re-translations* as a source of (noisy) normalization examples. Since such self-translations are relatively numerous on microblogs, this technique can provide a large amount of data.

We argue that NLP tools — like the very translation systems we propose to use — often fail on unnormalized input. Is this a problem? We argue that it is not for the following two reasons.

Normalization in translation. Work in translation studies has observed that translation tends to be a *generalizing* process that "smooths out" author- and work-specific idiosyncrasies [Laviosa, 1998, Volansky et al., 2013]. Assuming this observation is robust, we expect that dialectal variant forms found in microblogs to be normalized in translation. Therefore, if the parallel segments in our microblog parallel corpus did indeed originate through a translation process (rather than, e.g., being generated as two independent utterances from a bilingual), we may then state the following assumption about the distribution of variant forms in a parallel segment $\langle \mathbf{e}, \mathbf{f} \rangle$: *if* $\boldsymbol{e}$ *contains nonstandard lexical variants, then* $\boldsymbol{f}$ *is likely to be a normalized translation using with fewer nonstandard lexical variants (and vice-versa).*

Table 4.2: Corpora Used for Paraphrasing.

Lang. Pair	Source	Segs.	MT Engines
Mandarin-English	Weibo	800K	Google, Bing, Youdao
Mandarin-English	Twitter	113K	Google, Bing, Youdao
Arabic-English	Twitter	114K	Google, Bing
Russian-English	Twitter	119K	Google, Bing
Korean-English	Twitter	78K	Google, Bing
Japanese-English	Twitter	75K	Google, Bing

Uncorrelated orthographic variants. Any written language has the potential to make creative use of orthography: alphabetic scripts can render approximations of pronunciation variants; logographic scripts can use homophonic substitutions. However, the kinds of innovations used in particular languages would be language specific (depending on details of the phonology, lexicon, and orthography of the language). However, for language pairs that differ substantially in these dimensions, it may not always be possible (or at least easy) to preserve particular kinds of nonstandard orthographic forms in translation. Consider the (relatively common) pronoun-verb compounds like *iknw* and *imma* from our motivating example: since Chinese uses a logographic script without spaces, there is no obvious equivalent.

4.1.1 Variant–Normalized Parallel Corpus

For the two reasons outlined above, we argue that we would be able to translate back into English using MT, even when the underlying English part of the parallel corpus has a great deal of nonstandard content. We leverage this fact to build the normalization corpus, where the original English tweet is treated as the variant form, and the automatic translation obtained from another language is considered a potential normalization.[2]

Our process is as follows. We use all corpora that include English as one of the languages in the pair. The respective non-English side is translated into English using different translation engines. The different sets we used and the engines we used to translate are shown in Table 4.2. Thus, for each original English post $\mathbf{o}$, we obtain n paraphrases $\{\mathbf{p}_i\}_{i=1}^{n}$, from n different translation engines.

[2]We additionally assume that the translation engines are trained to output more standardized data, so there would be additional normalizing effect from the machine translation system.

4.1.2 Alignment and Filtering

Our parallel microblog corpus was crawled automatically and contains many misaligned sentences. To improve precision, we attempt to find the similarity between the (unnormalized) original and each of the normalizations using an alignment based on the one used in METEOR [Denkowski and Lavie, 2011], which computes the best alignment between the original tweet and each of the normalizations but modified to permit domain-specific approximate matches. To address lexical variants, we allow fuzzy word matching, that is, we allow lexically similar, such as *yea* and *yes* to be aligned (similarity is determined by the Levenshtein distance). We also perform phrasal matchings, such as *ikwn* to *i know*. To do so, we extend the alignment algorithm from word to phrasal alignments. More precisely, given the original post **o** and a candidate normalization **n**, we wish to find the optimal segmentation producing a good alignment. A segmentation $\mathbf{s} = \langle s_1, \ldots, s_{|\mathbf{s}|} \rangle$ is a sequence of segments that aligns as a block to a source word. For instance, for the sentence *yea iknw imma work on that*, one possible segmentation could be s_1 =*yea ikwn*, s_2 =*imma* and s_3 =*work on that*.

Model. We define the score of an alignment **a** and segmentation **s** using a model that makes semi-Markov independence assumptions, similar to the work in [Bansal et al., 2011]:

$$u(\mathbf{a}, \mathbf{s} \mid \mathbf{o}, \mathbf{n}) = \prod_{i=1}^{|\mathbf{s}|} \Big[u_e(s_i, a_i \mid \mathbf{n}) \times u_t(a_i \mid a_{i-1}) \times u_\ell(|s_i|) \Big]$$

In this model, the maximal scoring segmentation and alignment can be found using a polynomial time dynamic programming algorithm. Each segment can be aligned to any word or segment in **o**. The aligned segment for s_k is defined as a_k. For the score of a segment correspondence $u_e(s, a \mid \mathbf{n})$, we assume that this can be estimated using the lexical similarity between segments, which we define to be $1 - \frac{L(s_k, a_k)}{\max\{|s_k|, |a_k|\}}$, where $L(x, y)$ denotes the Levenshtein distance between strings x and y, normalized by the highest possible distance between those segments.

For the alignment score u_t, we assume that the relative order of the two sequences would be mostly monotonous. Thus, we approximate u_t with the following density $pos_s(a_k) - pos_e(a_{k-1}) \sim \mathcal{N}(1, 1)$, where the pos_s is the index of the first word in the segment and pos_e the one of the last word.

After finding the Viterbi alignments, we compute the similarity measure $\tau = \frac{|A|}{|A|+|U|}$, used in [Resnik and Smith, 2003], where $|A|$ and $|U|$ are the number of

words that were aligned and unaligned, respectively. In this work, we extract the pair if $\tau > 0.2$.

4.2 Normalization Model

From the normalization corpus, we learn a normalization model that generalizes the normalization process. That is, from the data we observe that *To DanielVeuleman yea iknw imma work on that* is normalized to *To Daniel Veuleman: yes, I know. I am going to work on that.* However, this is not useful, since the chances of the exact sentence *To DanielVeuleman yea iknw imma work on that* occurring in the data is low. We wish to learn a process to convert the original tweet into the normalized form.

There are two mechanisms that we use in our model. The first (§4.2.1) learns word–word and phrase–phrase mappings. That is, we wish to find that *DanielVeuleman* is normalized to *Daniel Veuleman*, that *iknw* is normalized to *I know* and that *imma* is normalized to *I am going*. These mappings are more useful, since whenever *iknw* occurs in the data, we have the option to normalize it to *I know*. The second (§4.2.2) learns character sequence mappings. If we look at the normalization *DanielVeuleman* to *Daniel Veuleman*, we can see that it is only applicable when the exact word *DanielVeuleman* occurs. However, we wish to learn that it is uncommon for the letters *l* and *v* to occur in the same word sequentially, so that be can add missing spaces in words that contain the *lv* character sequence, such as normalizing *phenomenalvoter* to *phenomenal voter*. However, there are also cases where this is not true, for instance, in the word *velvet*, we do not wish to separate the letters *l* and *v*. Thus, we shall describe the process we use to decide when to apply these transformations.

4.2.1 From Sentences To Phrases

The process to find phrases from sentences has been throughly studied in machine translation. This is generally done in two steps, Word Alignment and Phrase Extraction.

Alignment. The first step is to find the word-level alignments between the original post and its normalization. This is a well studied problem in MT, referred as Word

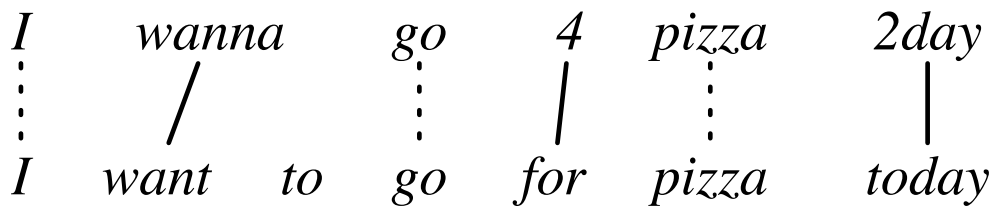

Figure 4.1: Variant–normalized alignment with the variant form above and the normalized form below; solid lines show potential normalizations, while dashed lines represent identical translations.

Alignment [Brown et al., 1993]. In our work, we use the fast aligner proposed in [Dyer et al., 2013] to obtain the word alignments. Figure 4.1 shows an example of a word aligned pair of a tweet and its normalization.

Phrase Extraction. The phrasal extraction step [Ling et al., 2010] uses the word aligned sentences and extracts phrasal mappings between the original tweet and its normalization, named phrase pairs. For instance, in Figure 4.1, we would like to extract the phrasal mapping from *go 4* to *go for*, so that we learn that the word *4* in the context of *go* is normalized to the proposition *for*. To do this, the most common approach is to use the template proposed in [Och and Ney, 2004], which allows phrase pairs to be extracted, if there is at least one word alignment within the pair, and there are no words inside the pair that are aligned to words not in the pair. For instance, in the example above, the phrase pair that normalizes *wanna* to *want to* would be extracted, but the phrase pair normalizing *wanna* to *want to go* would not, because the word *go* in the normalization is aligned to a word not in the pair.

Phrasal Features. After extracting the phrase pairs, a model is produced with features derived from phrase pair occurrences during extraction. This model is equivalent to a phrasal translation model in MT, but we shall refer to it as the normalization model. For a phrase pair $\langle \mathbf{o}, \mathbf{n} \rangle$, where $\mathbf{o}$ is the original phrase, and $\mathbf{n}$ is the normalized phrase, we compute the normalization relative frequency $f(\mathbf{n} \mid \mathbf{o}) = \frac{C(\mathbf{n},\mathbf{o})}{C(\mathbf{o})}$, where $C(\mathbf{n}, \mathbf{o})$ denotes the number of times $\mathbf{o}$ was normalized to $\mathbf{n}$ and $C(\mathbf{o})$ denotes the number of times $\mathbf{o}$ was seen in the extracted phrase pairs.

Table 4.3: Fragment of the phrase normalization model built. For each original phrase **o**, we present the top-3 normalized forms ranked by $f(\mathbf{n} \mid \mathbf{o})$.

Original (**o**)	Normalization (**n**)	$f(\mathbf{n} \mid \mathbf{o})$
wanna	want to	0.4679
wanna	will	0.0274
wanna	going to	0.0114
4	4	0.5641
4	for	0.01795
go 4	go for	1.0000

Table 4.3 gives a fragment of the normalization model. The columns represent the original phrase, its normalization and the probability, respectively.

In Table 4.3, we observe that the abbreviation *wanna* is normalized to *want to* with a relatively high probability, but it can also be normalized to other equivalent expressions, such as *will* and *going to*. The word *4* by itself has a low probability to be normalized to the preposition *for*. This is expected, since this decision cannot be made without context. However, we see that the phrase *go 4* is normalized to *go for* with a high probability, which specifies that within the context of *go*, *4* is generally used as a preposition.

4.2.2 From Phrases to Characters

While we can learn lexical variants that are in the corpora using the phrase model, we can only address word forms that have been observed in the corpora. This is quite limited, since we cannot expect all the word forms to be present, such as all the possible orthographic errors for the word *cat*, such as *catt*, *kat* and *caaaat*. Thus, we will build a character-based model that learns the process lexical variants are generated at the subword level.

Our character-based model is similar to the phrase-based model, except that, rather than learning word-based mappings from the original tweet and the normalization sentences, we learn character-based mappings from the original phrases to the normalizations of those phrases. Thus, we extract the phrase pairs in the phrasal normalization model, and use them as a training corpora. To do this, for each phrase pair, we add a start token, <*start*>, and an end token, <*end*>, at the beginning and ending of the phrase pair. Afterwards, we separate all characters by

Table 4.4: Fragment of the character normalization model where examples representative of the lexical variant generation process are encoded in the model.

Original (**o**)	Normalization (**n**)	$f(\mathbf{n} \mid \mathbf{o})$
o o o	o o	0.0223
o o o	o	0.0439
s	c	0.0331
z	s	0.0741
s h	c h	0.019
2	t o	0.014
4	f o r	0.0013
0	o	0.0657
i n g f o r	i n g <space> f o r	0.4545
g f	g <space> f	0.01028

space and add a space token <*space*> where spaces were originally. For instance, the phrase pair normalizing *DanielVeuleman* to *Daniel Veuleman* would be converted to <*start*> *d a n i e l v e u l e m a n* <*end*> and <*start*> *d a n i e l* <*space*> *v e u l e m a n* <*end*>.

Character-based Normalization Model - To build the character-based model, we proceed using the same approach as in the phrasal normalization model. We first align characters using Word Alignment Models, and then we perform phrase extraction to retrieve the phrasal character segments, and build the character-based model by collecting statistics. Once again, we provide examples of entries in the model in Table 4.4.

We observe that many of the normalizations dealt with in the previous model by memorizing phrases are captured with string transformations. For instance, from phrase pairs such as *tooo* to *too* and *sooo* to *so*, we learn that sequences of *o*'s can be reduced to 2 or 1 *o*. Other examples include orthographic substitutions, such as *2* for *to* and *4* for *for* (as found in *2gether*, *2morrow*, *4ever* and *4get*). Moreover, orthographic errors can be generated from mistaking characters with similar phonetic properties, such as, *s* to *c*, *z* to *s* and *sh* to *ch*, generating lexical variants such as *reprecenting*. Finally, we learn that the number *0* that resembles the letter *o*, can be used as a replacement, as in *g00d*. Finally, we can see that the rule *ingfor* to *ing for* attempts to find segmentation errors, such as *goingfor*, where a

space between *going* and *for* was omitted.[3]

4.3 Normalization Decoder

In section 4.2, we built two models to learn the process of normalization, the phrase-based model and the character-based model. In this section, we describe the decoder we used to normalize the sentences.

The advantage of the phrase-based model is that it can make decisions for normalization based on context. That is, it contains phrasal units, such as, *go 4*, that determine, when the word *4* should be normalized to the preposition *for* and when to leave it as a number. However, it cannot address words that are unseen in the corpora. For instance, if the word form *4ever* is not seen in the training corpora, it is not be able to normalize it, even if it has seen the word *4get* normalized to *forget*. On the other hand, the character-based model learns subword normalizations, for instance, if we see the word *nnnnno* normalized to *no*, we can learn that repetitions of the letter *n* are generally shorted to *n*, which allows it to generate new word forms. This model has strong generalization potential, but the weakness of the character-based model is that it fails to consider the context of the normalization that the phrase-based model uses to make normalization decisions. Thus, our goal in this section is describe a decoder that uses both models to improve the quality of the normalizations.

4.3.1 Phrasal Decoder

We use Moses, an off-the-shelf phrase-based MT system [Koehn et al., 2007], to "translate" the original tweet to its normalized form using the phrasal model (§4.2.1). Aside from the normalization probability, we also include the common features used in MT. These are the reverse normalization probability, the lexical and reverse lexical probabilities and the phrase penalty. We also use the MSD reordering model

[3]Note that this captures the context in which such transformations are likely to occur: there are not many words that contain the sequence *ingfor*, so the probability that these should be normalized by inserting a space is high. On the other hand, we cannot assume that if we observe the sequence *gf*, we can safely separate these with a space. This is because, there are many words that contain this sequence, such as the abbreviation of *gf* (*girlfriend*), *dogfight*, and *bigfoot*.

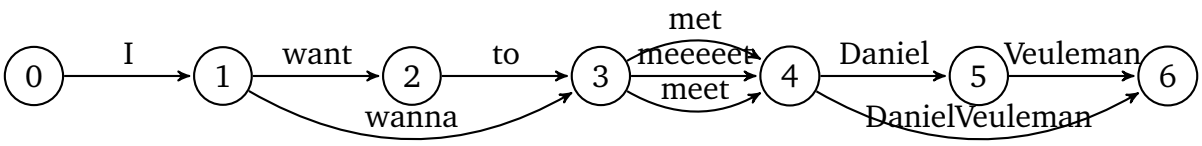

Figure 4.2: Example output lattice of the character-based decoder, for the sentence *I wanna meeeeet DanielVeuleman*.

proposed in [Koehn et al., 2005], which adds reordering features.[4] The final score of each phrase pair is given as a sum of weighted log features. The weights for these features are optimized using MERT [Och, 2003]. In our work, we sampled 150 tweets randomly from Twitter and normalized them manually, and used these samples as development data for MERT. As for the character-based model features, we simply rank the training phrase pairs by their relative frequency the $f(\mathbf{n} \mid \mathbf{o})$, and use the top-1000 phrase pairs as development set. Finally, a language model is required during decoding as a prior, since it defines the type of language that is produced by the output. We wish to normalized to formal language, which is generally better processed by NLP tools. Thus, for the phrase model, we use the English NIST dataset composed of 8M sentences in English from the news domain to build a 5-gram Kneser-Ney smoothed language model.

4.3.2 Character and Phrasal Decoder

We now turn to how to apply the character-based (§4.2.2), together with the phrasal model. For this model, we again use Moses, treating each character as a "word". The simplest way to combine both methods is first to decode the input sentence **o** with the character-based decoder, normalizing each word independently and then normalizing the resulting output using the phrase-based decoder, which enables the phrase model to score the outputs of the character model in context.

Our process is as follows. Given the input sentence **o**, with the words $o_1, \ldots, o_m$, where m is the number of words in the input, we generate for each word o_i a list of n-best normalization candidates $z^1_{o_i}, \ldots, z^n_{o_i}$. We further filter the candidates using two criteria. We start by filtering out each candidate $z^j_{o_i}$ that occurs less frequently than the original word o_i. This is motivated by our observation that lexical variants occur far less than the respective standard form. Second, we build a corpus of

[4]Reordering helps find lexical variants that are generated by transposing characters, such as, *mabye* to *maybe*.

English language Twitter consisting of 70M tweets, extract the unigram counts, and perform Brown clustering [Brown et al., 1992] with $k = 3000$ clusters. Next, we calculate the cluster similarity between o_i and each surviving candidate, $z^j_{o_i}$. We filter the candidate if the similarity is less than 0.8. The similarity between two clusters represented as *bit strings*, $S[c(o_i), c(z^j_{o_i})]$, calculated as:

$$S(x, y) = \frac{2 \cdot |lpm\{x, y)\}|}{|x| + |y|},$$

where lpm computes the longest common prefix of the contexts and $|x|$ is the length of the bit string.[5] If a candidate contains more than one word (because a space was inserted), we set its count as the minimum count among its words. To find the cluster for multiple word units, we concatenate the words together, and find the cluster with the resulting word if it exists. This is motivated by the fact that it is common for missing spaces to exist in microblog corpora, generating new word forms, such as *wantto*, *goingfor*, and given a large enough corpora as the one we used, these errors occur frequently enough to be placed in the correct cluster. For example, the variants such as *wanna* and *tmi*, occur in the same clusters as the words *wantto* and *toomuchinformation*.

Remaining candidates are combined into a word lattice, enabling us to perform lattice-based decoding with the phrasal model [Dyer et al., 2008]. Figure 4.3.2, provides an example of such a lattice for the variant sentence *I wanna meeeet DanielVeuleman*. Lattice inputs permit the decoder to choose on the basis of translation and language model features – contextually appropriate inputs.

4.3.3 Learning Variants from Monolingual Data

Until now, we learned normalizations from pairs of original tweets and their normalizations. We shall now describe a process to leverage monolingual documents to learn new normalizations, since the monolingual data is far easier to obtain than parallel data. This process is similar to the work in [Han et al., 2012], where confusion sets of contextually similar words are built initially as potential normalization candidates. We again use the $k = 3000$ Brown clusters,[6] and this time consider the contents of each cluster as a set of possible normalization variants. For instance, we

[5]Brown clusters are organized such that more words with more similar distributions share common prefixes.

[6]The Brown clustering algorithm groups words together based on contextual similarity.

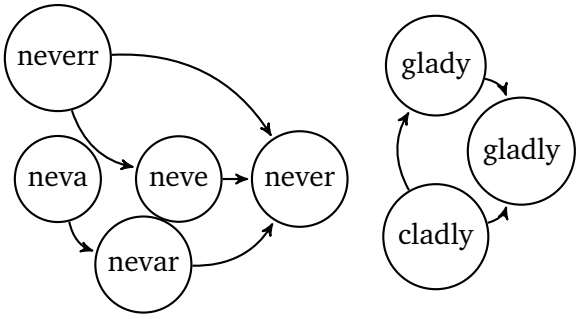

Figure 4.3: Example DAGs, built from the cluster containing the words *never* and *gladly*.

find that the cluster that includes the word *never*, also includes the variant forms *neverrrr*, *neva* and *nevahhh*. However, the cluster also contains non-variant forms, such as *gladly* and *glady*. Thus, we want to find that *neverrrr* maps to *never*, while *glady* maps to *gladly* in the same cluster. Our work differs from previous work in that, rather than defining features manually, we use our character-based decoder to find the mappings between lexical variants and their normalizations.

For every word type w_i in cluster $c(w_i) = \{w_1, \ldots, w_n\}$, we generate a set of possible candidates for each word $w_i^1, \ldots, w_i^m$. Then, we build a directed acyclic graph (DAG), where every word. We add an edge between w_i and w_j, if w_i can be decoded into w_j using the character model from the previous section, and also if w_i occurs less than w_j; the second condition guarantees that the graph will be acyclic. Sample graphs are shown in Figure 4.3.

Afterwards, we find the number of paths between all the nodes in the graph (this can be computed efficiently in $O(|V| + |E|)$ time). Then, for each word w_i, we find the w_j to which it has the highest number of paths to and extract the normalization of w_i to w_j. In case of a tie, we choose the word w_j that occurs more often in the monolingual corpora. This is motivated by the fact that normalizations are transitive. Thus, even if *neva* cannot be decoded directly to *never*, we can use *nevar* as an intermediate step to find the correct normalization. This is performed for all the clusters, and the resulting dictionary of lexical variants mapped to their standard forms is added to the training data of the character-based model.

4.4 Experiments

We evaluate our normalization model intrinsically by testing whether our normalizations more closely resemble standardized data, and then extrinsically by testing whether we can improve the translation quality of in-house as well as online machine translation systems by normalizing the input.

4.4.1 Setup

Once again, we use our parallel dataset, composed by 2581 English-Mandarin microblog sentence pairs obtained in Chapter 3. From this set, we randomly select 1290 pairs for development and 1291 pairs for testing.

The normalizer model is trained on the corpora extracted and filtered in section 4.1, in total, there were 1.3M normalization pairs used during training. The test sentences are normalized using four different setups. The first setup leaves the input sentence unchanged, which we call **No Norm**. The second uses the phrase-based model to normalize the input sentence, which we will denote **Norm+phrase**. The third uses the character-based model to output lattices, and then decodes with the phrase based model, which we will denote **Norm+phrase+char**. Finally, we test the same model after adding the training data extracted using monolingual documents, which we will refer as **Norm+phrase+char+mono**.

To test the normalizations themselves, we used Google Translate to translate the Mandarin side of the 1291 test sentence pairs back to English and use the original English tweet. While, this is by itself does not guarantee that the normalizations are correct, since the normalizations could be syntactically and semantically incorrect, it will allow us to check whether the normalizations are closer to those produced by systems trained on news data. This experiment will be called **Norm**.

As an application and extrinsic evaluation for our normalizer, we test if we can obtain gains on the MT task on microblog data by using our normalizer prior to translation. We build two MT systems using Moses. Firstly, we build an out-of-domain model using the full 2012 NIST Chinese-English dataset (approximately 8M sentence pairs), which is a dataset from the news domain, and we will denote this system as **Inhouse+News**. Secondly, we build a in-domain model using the 800K highest ranked sentence pairs extracted in Chapter 3. We also add the NIST dataset to improve coverage. We call this system **Inhouse+News+Weibo**. To train these systems, we use the Moses phrase-based MT system with standard features [Koehn

Table 4.5: Normalization and MT Results. Rows denote different normalizations, and columns different translation systems, except the first column (Norm), which denotes the normalization experiment. Cells display the BLEU score of that experiment.

Condition	Norm	In house (News)	In house (News+Weibo)	Online A	Online B	Online C
baseline	19.90	15.10	24.37	20.09	17.89	18.79
norm+phrase	21.96	15.69	24.29	20.50	18.13	18.93
norm+phrase+char	22.39	15.87	24.40	20.61	18.22	19.08
norm+phrase+char+mono	**22.91**	**15.94**	**24.46**	**20.78**	**18.37**	**19.21**

et al., 2003]. For reordering, we use the MSD reordering model [Axelrod et al., 2005]. As the language model, we train a 5-gram model with Kneser-Ney smoothing using a 10M tweets from twitter. Finally, the weights were tuned using MERT [Och, 2003]. As for online systems, we consider the systems used to generate the paraphrase corpora in section 4.1, which we will denote as **Online A**, **Online B** and **Online C**[7].

The normalization and MT results are evaluated with BLEU-4 [Papineni et al., 2002], comparing the produced translations or normalizations with the appropriate reference.

4.4.2 Results

Results are shown in Table 4.5. In terms of the normalizations, we observe a much better match between the normalized text with the reference, than the original tweets. In most cases, adding character-based models improves the quality of the normalizations.

We observe that better normalizations tend to lead to better translations. The relative improvements are most significant, when moving from No Norm to **norm+phrase** normalization. This is because we are normalizing words that are not seen in general MT system's training data, but occur frequently in microblog data, such as *wanna* to *want to*, *u* to *you* and *im* to *i'm*. The only exception is in the **Inhouse+News+Weibo** system, where the normalization deteriorates the results. This is to be expected, since this system is trained on the same microblog data used to learn the normal-

[7]The names of the systems are hidden to not violate the privacy issues in the terms and conditions of these online systems.

izations. However, we can observe on **norm+phrase+char** that if we add the character-based model, we can observe improvements for this system as well as for all other ones. This is because the model is actually learning normalizations that are unseen in the data. Some examples of these normalization include, normalizing *lookin* to *looking*, *nutz* to *nuts* and *maimi* to *miami* but also separating *peaceof* to *peace of*. The fact that these improvements are obtained for all systems is strong evidence that we are actually producing good normalizations, and not overfitting to one of the systems that we used to generate our data. The gains are much smaller from **norm+phrase** to **norm+phrase+char**, since the improvements we obtain come from normalizing less frequent words. Finally, we can obtain another small improvement by adding monolingual data to the character-based model in **norm+phrase+char+mono**.

4.4.3 Summary

In this chapter, we introduced a data-driven approach to microblog normalization based on paraphrasing. We build a corpora of tweets and their normalizations using parallel corpora from microblogs using MT techniques. Then, we build two models that learn generalizations of the normalization process, one the phrase level and on the character level. Then, we build a decoder that combines both models during decoding. Improvements on multiple MT systems support the validity of our method.

Chapter 5

Character-based Word Representations for NLP

Generalization is one of the main concerns when building any statistical model. Given two models, the model with better generalization capabilities will perform better at unseen data, even though they are trained on the same datasets. As a simple example, suppose that our training data tells us that *greatly* and *objectively* are adjectives, while *John* and *Peter* are proper nouns. One way to model the data would be simply memorizing all words and their labels in a dictionary. The problem with this approach is that given the unseen example *poorly*, the model would be unable to classify this example. On the other hand, if the model is able to capture that both adjectives end with the suffix *-ly*, and both proper nouns start with a capital letter, many of the unseen examples would be labelled correctly. It is important to note that while both models can achieve a perfect accuracy on the seen data, the quality of the model is solely associated with their performance on unseen examples.

One would imagine that in a domain such as Twitter, where many lexical variants are infrequent, a model that would consider character level information, rather than simply memorizing words is desirable. Unfortunately, phrase tables used in current phrase-based MT systems [Koehn et al., 2007] simply memorize words. That is, each entry maps a source phrase to a target phrase, and any changes to either phrases would result in a different phrase pair. This is problematic on one hand as the model is incapable of learning the translations of unseen words, even if these are simply slight variations of existing words in the phrase table (e.g. cool $\rightarrow$ coooool), but also as we cannot generate for forms that have not been seen. Moreover, phrase

tables generated from phrase-based systems tend to be extremely large due to the fact that all contexts must be encoded as separate entries. Combined with the fact that the creative language in microblogs alone can generate millions of word types, an approach that simply memorizes translations for all words in their different contexts is definitely not scalable to large datasets. While we propose methods to alleviate this problem by standardizing [Ling et al., 2013a] and pruning [Ling et al., 2012a] phrase tables in the work developed in this thesis, most of these methods are used to compensate a missing property in phrase-based systems: compositionality.

Compositionality allows complex structures to be formed by increasingly smaller and simpler structures [Dyer et al., 2015]. For instance, documents are composed of sentences, which are in turn composed by words, which are composed of characters. An example in machine translation is that the translation of a sentence can be decomposed into the translation of smaller phrases. Yet, the phrase extraction process [Ling et al., 2010] extracts many redundant phrase pairs that could be composed by even smaller phrase pairs. Thus, the model is not fully exploiting the compositional properties within language that allows very compact models to be learnt. In prior work, we provide evidence [Ling et al., 2012a] that many of these redundant phrases can be removed for compactness with only a small negative impact in the translation quality.

In this chapter, we will develop models that take major advantage of the compositional properties within language. Our work [Ling et al., 2015c] focuses mainly on the character to word relationship. That is, how to model words solely from their characters. Then, we show their validity in two major NLP tasks, language modeling and part-of-speech tagging.

5.1 Word Vectors and Wordless Word Vectors

Our model is motivated by the recent advances in vector space models used in neural network models. In contrast to naïve models in which all word types in a vocabulary V are equally different from each other, vector space models capture the intuition that words may be different or similar along a variety of dimensions. Learning vector representations of words by treating them as optimizable parameters in various kinds of language models has been found to be a remarkably effective means for generating vector representations that perform well in other tasks [Collobert et al., 2011, Kalchbrenner and Blunsom, 2013, Liu et al., 2014, Chen and Manning, 2014, Dyer et al., 2015]. Formally, such models define a matrix $\mathbf{P} \in \mathbb{R}^{d \times |V|}$, which

contains d parameters for each word in the vocabulary V. For a given word type $w \in V$, a column is selected by right-multiplying $\mathbf{P}$ by a one-hot vector of length $|V|$, which we write $\mathbf{onehot}(w)$, that is zero in every dimension except for the element corresponding to w. Thus, $\mathbf{P}$ is often referred to as word lookup table and we shall denote by $\mathbf{e}_w^W \in \mathbb{R}^d$ the embedding obtained from a word lookup table for w as $\mathbf{e}_w^W = \mathbf{P} \cdot \mathbf{onehot}(w)$. This allows tasks with low amounts of annotated data to be trained jointly with other tasks with large amounts of data and leverage the similarities in these tasks. A common practice to this end is to initialize the word lookup table with the parameters trained on an unsupervised task. Some examples of these include the skip-n-gram and CBOW models of [Mikolov et al., 2013].

5.1.1 Problem: Independent Parameters

There are two practical problems with word lookup tables. Firstly, while they can be pre-trained with large amounts of data to learn semantic and syntactic similarities between words, each vector is independent. That is, even though models based on word lookup tables are often observed to learn that *cats*, *kings* and *queens* exist in roughly the same linear correspondences to each other as *cat*, *king* and *queen* do, the model does not represent the fact that adding an *s* at the end of the word is evidence for this transformation. This means that word lookup tables cannot generate representations for previously unseen words, such as *Frenchification*, even if the components, *French* and *-ification*, are observed in other contexts.

Second, even if copious data is available, it is impractical to actually store vectors for all word types. As each word type gets a set of parameters d, the total number of parameters is $d \times |V|$, where $|V|$ is the size of the vocabulary. Even in relatively morphologically poor languages, such as English, the number of word types tends to scale to the order of hundreds of thousands, and in noisier domains, such as online data, the number of word types raises considerably. For instance, in the English Wikipedia dump with 60 million sentences, there are approximately 20 million different lowercased and tokenized word types, each of which would need its own vector. Intuitively, it is not sensible to use the same number of parameters for each word type.

Finally, it is important to remark that it is uncontroversial among cognitive scientists that our lexicon is structured into related forms—i.e., their parameters are not independent. The well-known "past tense debate" between connectionists and proponents of symbolic accounts concerns disagreements about how humans represent knowledge of inflectional processes (e.g., the formation of the English

past tense), not whether such knowledge exists [Marslen-Wilson and Tyler, 1998].

5.1.2 Solution: Compositional Models

Our solution to these problems is to construct a vector representation of a word by composing smaller pieces into a representation of the larger form. This idea has been explored in prior work by composing *morphemes* into representations of words [Luong et al., 2013, Botha and Blunsom, 2014, Soricut and Och, 2015]. Morphemes are an ideal primitive for such a model since they are—by definition—the minimal meaning-bearing (or syntax-bearing) units of language. The drawback to such approaches is they depend on a morphological analyzer.

In contrast, we would like to compose representations of *characters* into representations of words. However, the relationship between words forms and their meanings is non-trivial [de Saussure, 1916]. While some compositional relationships exist, e.g., morphological processes such as adding *-ing* or *-ly* to a stem have relatively regular effects, many words with lexical similarities convey different meanings, such as, the word pairs *lesson* $\iff$ *lessen* and *coarse* $\iff$ *course*.

5.2 C2W Model

Our compositional character to word (C2W) model is based on bidirectional LSTMs [Graves and Schmidhuber, 2005], which are able to learn complex non-local dependencies in sequence models. An illustration is shown in Figure 5.1. The input of the C2W model (illustrated on bottom) is a single word type w, and we wish to obtain a d-dimensional vector used to represent w. This model shares the same input and output of a word lookup table (illustrated on top), allowing it to easily replace them in any network.

As input, we define an alphabet of characters C. For English, this vocabulary would contain an entry for each uppercase and lowercase letter as well as numbers and punctuation. The input word w is decomposed into a sequence of characters $c_1, \ldots, c_m$, where m is the length of w. Each c_i is defined as a one hot vector $\mathbf{onehot}(c_i)$, with one on the index of c_i in vocabulary M. We define a projection layer $\mathbf{P}_C \in \mathbb{R}^{d_C \times |C|}$, where d_C is the number of parameters for each character in the character set C. This of course just a character lookup table, and is used to capture similarities between characters in a language (e.g., vowels *vs*. consonants). Thus,

we write the projection of each input character c_i as $\mathbf{e}_{c_i} = \mathbf{P}_C \cdot \mathbf{onehot}(c_i)$.

Given the input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$, a LSTM computes the state sequence $\mathbf{h}_1, \ldots, \mathbf{h}_{m+1}$ by iteratively applying the following updates:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \\
&\qquad \mathbf{i}_t \odot \tanh(\mathbf{W}_{cx}\mathbf{x}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned}$$

where σ is the component-wise logistic sigmoid function, and $\odot$ is the component-wise (Hadamard) product. LSTMs define an extra cell memory $\mathbf{c}_t$, which is combined linearly at each timestamp t. The information that is propagated from $\mathbf{c}_{t-1}$ to $\mathbf{c}_t$ is controlled by the three gates $\mathbf{i}_t$, $\mathbf{f}_t$, and $\mathbf{o}_t$, which determine what to include from the input $\mathbf{x}_t$, what to forget from $\mathbf{c}_{t-1}$ and what is relevant to the current state $\mathbf{h}_t$. We write $\mathcal{W}$ to refer to all parameters the LSTM, ($\mathbf{W}_{ix}$, $\mathbf{W}_{fx}$, $\mathbf{b}_f$, ...). Thus, given a sequence of character representations $\mathbf{e}^C_{c_1}, \ldots, \mathbf{e}^C_{c_m}$ as input, the forward LSTM, yields the state sequence $\mathbf{s}^f_0, \ldots, \mathbf{s}^f_m$, while the backward LSTM receives as input the reverse sequence, and yields states $\mathbf{s}^b_m, \ldots, \mathbf{s}^b_0$. Both LSTMs use a different set of parameters $\mathcal{W}^f$ and $\mathcal{W}^b$. The representation of the word w is obtained by combining the forward and backward states:

$$\mathbf{e}^C_w = \mathbf{D}^f\mathbf{s}^f_m + \mathbf{D}^b\mathbf{s}^b_0 + \mathbf{b}_d,$$

where $\mathbf{D}^f$, $\mathbf{D}^b$ and $\mathbf{b}_d$ are parameters that determine how the states are combined.

Caching for Efficiency. Relative to $\mathbf{e}^W_w$, computing $\mathbf{e}^C_w$ is computational expensive, as it requires two LSTMs traversals of length m. However, $\mathbf{e}^C_w$ only depends on the character sequence of that word, which means that unless the parameters are updated, it is possible to cache the value of $\mathbf{e}^C_w$ for each different w that will be used repeatedly. Thus, the model can keep a list of the most frequently occurring word types in memory and run the compositional model only for rare words. Obviously, caching all words would yield the same performance as using a word lookup table $\mathbf{e}^W_w$, but would also use the same amount of memory. Consequently, the number

of word types used in cache can be adjusted to satisfy memory vs. performance requirements of a particular application.

At training time, when parameters are changing, repeated words within the same batch only need to be computed once, and the gradients for each word vector can be accumulated within the batch so that only one update needs to be done per word type. For this reason, it is preferable to define larger batches.

5.3 Experiments: Language Modeling

Our proposed model is similar to models used to compute composed representations of sentences from words [Cho et al., 2014, Li et al., 2015]. However, the relationship between the meanings of individual words and the composite meaning of a phrase or sentence is arguably more regular than the relationship of representations of characters and the meaning of a word. Is our model capable of learning such an irregular relationship? We now explore this question empirically.

Language modeling is a task with many applications in NLP. An effective LM requires syntactic aspects of language to be modeled, such as word orderings (e.g., "John is smart" *vs*. "John smart is"), but also semantic aspects (e.g., "John ate fish" *vs*. "fish ate John"). Thus, if our C2W model only captures regular aspects of words, such as, prefixes and suffixes, the model will yield worse results compared to word lookup tables.

5.3.1 Language Model

Language modeling amounts to learning a function that computes the log probability, $\log p(\boldsymbol{w})$, of a sentence $\boldsymbol{w} = (w_1, \ldots, w_n)$. This quantity can be decomposed according to the chain rule into the sum of the conditional log probabilities $\sum_{i=1}^{n} \log p(w_i \mid w_1, \ldots, w_{i-1})$. Our language model computes $\log p(w_i \mid w_1, \ldots, w_{i-1})$ by composing representations of words $w_1, \ldots, w_{i-1}$ using a recurrent LSTM model [Mikolov et al., 2010, Sundermeyer et al., 2012a].

The model is illustrated in Figure 5.2, where we observe on the first level that each word w_i is projected into its word representation. This can be done by using word lookup tables $\mathbf{e}^W_{w_i}$, in which case we will have a regular recurrent language model. To use our C2W model, we can simply replace the word lookup table with the model $f(w_i) = \mathbf{e}^C_{w_i}$. Each LSTM block $\mathbf{s}_i$ is used to predict word w_{i+1}. This is

performed by projecting the s_i into a vector of the size of the vocabulary V and performing a softmax.

The softmax is still simply a $d \times V$ table, which encodes the likelihood of every word type in a given context. Since it is a closed-vocabulary model, at test time, out-of-vocabulary (OOV) words cannot be addressed. A strategy that is generally applied is to prune the vocabulary V by replacing word types with lower frequencies as an OOV token. At test time, the probability of words not in vocabulary is estimated as the OOV token. Thus, depending on the number of word types that are pruned, the global perplexities may decrease, since there are fewer outcomes in the softmax, which makes the absolute value of perplexity not informative when comparing models of different vocabulary sizes. Yet, the relative perplexity between different models indicates which models can better predict words based on their contexts.

5.3.2 Experiments

Datasets We look at the language model performance on English, Portuguese, Catalan, German and Turkish, which have a broad range of morphological typologies. While all these languages contain inflections, in agglutinative languages affixes tend to be unchanged, while in fusional languages they are not. For each language, Wikipedia articles were randomly extracted until 1 million words are obtained and these were used for training. For development and testing, we extracted an additional set of 20,000 words.

Setup We define the size of the word representation d to 50. The C2W model requires setting the dimensionality of characters d_C and current states d_{CS}. We set $d_C = 50$ and $d_{CS} = 150$. Each LSTM state used in the language model sequence s_i is set to 150 for both states and cell memories. Training is performed with mini-batch gradient descent with 100 sentences. The learning rate and momentum are set to 0.2 and 0.95. The softmax over words is always performed on lowercased words. We restrict the output vocabulary to the most frequent 5,000 words. Remaining word types will be replaced by an unknown token, which must also be predicted. The word representation layer is still performed over all word types (i.e., completely open vocabulary). When using word lookup tables, the input words are also lowercased, as this setup produces the best results. In the C2W, case information is preserved.

To address OOV words in the baseline setup, these are replaced by an unknown

token, and also associated with a set of embeddings. During training, word types that occur once are replaced with the unknown token stochastically with 0.5 probability. The same process is applied at the character level for the C2W model.

Evaluation is performed by computing the perplexities over the test data, and the parameters that yield the highest perplexity over the development data are used.

Perplexities Perplexities over the testset are reported on Table 6.1. From these results, we can see that in general, it is clear that C2W always outperforms word lookup tables (row “Word"), and that improvements are especially pronounced in Turkish, which is a highly morphological language, where word meanings differ radically depending on the suffixes used (*evde* $\rightarrow$ *in the house vs. evden* $\rightarrow$ *from the house*).

Number of Parameters As for the number of parameters (illustrated for block “#Parameters"), the number of parameters in word lookup tables is $V \times d$. If a language contains 80,000 word types (a conservative estimate in morphologically rich languages), 4 million parameters would be necessary. On the other hand, the compositional model consists of 8 matrices of dimensions $d_{CS} \times d_C + 2d_{CS}$. Additionally, there is also the matrix that combines the forward and backward states of size $d \times 2d_{CS}$. Thus, the number of parameters is roughly 150,000 parameters—substantially fewer. This model also needs a character lookup table with d_C parameters for each entry. For English, there are 618 characters (including noise, such as Mandarin and Russian characters), for an additional 30,900 parameters. So the total number of parameters for English is roughly 180,000 parameters (2 to 3 parameters per word type), which is an order of magnitude lower than word lookup tables.

Performance As for efficiency, both representations can score sentences at a rate of approximately 300 words per second during training. While this is surprising, due to the fact that the C2W model requires a composition over characters, the main bottleneck of the system is the softmax over the vocabulary. Furthermore, caching is used to avoid composing the same word type twice in the same batch. This shows that the C2W model is relatively fast compared operations such as a softmax.

	Fusional			Agglutinative	
Perplexity	EN	PT	CA	DE	TR
5-gram KN	70.72	58.73	39.83	59.07	52.87
Word	59.38	46.17	35.34	43.02	44.01
C2W	**57.39**	**40.92**	**34.92**	**41.94**	**32.88**
#Parameters					
Word	4.3M	4.2M	4.3M	6.3M	5.7M
C2W	**180K**	**178K**	**182K**	**183K**	**174K**

Table 5.1: Language Modeling Results.

increased	*John*	*Noahshire*	*phding*
reduced	Richard	Nottinghamshire	mixing
improved	George	Bucharest	modelling
expected	James	Saxony	styling
decreased	Robert	Johannesburg	blaming
targeted	Edward	Gloucestershire	christening

Table 5.2: Most-similar in-vocabular words under the C2W model; the two query words on the left are in the training vocabulary, those on the right are nonce (invented) words.

Representations of (invented) words While it is promising that the model is not simply learning lexical features, what is most interesting is that the model can propose embeddings for nonce/invented words, in stark contrast to the situation observed with lookup table models. To illustrated this, we show the 5-most-similar in-vocabulary words (measured with cosine similarity) as computed by our character model on two in-vocabulary words and two nonce words. This makes our model generalize significantly better than lookup tables that generally use unknown tokens for OOV words. Furthermore, this ability to generalize is much more similar to that of human beings, who are able to infer meanings for new words based on its form.

5.4 Experiments: Part-of-speech Tagging

As a second illustration of the utility of our model, we turn to POS tagging. As morphology is a strong indicator for syntax in many languages, much effort has been spent engineering features [Nakagawa et al., 2001, Mueller et al., 2013]. We now show that some of these features can be learnt automatically using our model.

5.4.1 Bi-LSTM Tagging Model

Our tagging model is likewise novel, but very straightforward. It builds a Bi-LSTM over words as illustrated in Figure 5.3. The input of the model is a sequence of features $f(w_1), \ldots, f(w_n)$. Once again, word vectors can either be generated using the C2W model $f(w_i) = \mathbf{e}^C_{w_i}$, or word lookup tables $f(w_i) = \mathbf{e}^W_{w_i}$. We also test the usage of hand-engineered features, in which case $f_1(w_i), \ldots, f_n(w_i)$. Then, the sequential features $f(w_1), \ldots, f(w_n)$ are fed into a Bi-LSTM model, obtaining the forward states $\mathbf{s}^f_0, \ldots, \mathbf{s}^f_n$ and the backward states $\mathbf{s}^b_{N+1}, \ldots, \mathbf{s}^b_0$. Thus, state $\mathbf{s}^f_i$ contains the information of all words from 1 to i and $\mathbf{s}^b_i$ from n to i. The forward and backward states are combined, for each index from 1 to n, as follows:

$$\mathbf{l}_i = \tanh(\mathbf{L}^f \mathbf{s}^f_i + \mathbf{L}^b \mathbf{s}^b_i + \mathbf{b}_l),$$

where $\mathbf{L}^f$, $\mathbf{L}^b$ and $\mathbf{b}_l$ are parameters defining how the forward and backward states are combined. The size of the forward $\mathbf{s}^f$ and backward states $\mathbf{s}^b$ and the combined state $\mathbf{l}$ are hyperparameters of the model, denoted as d^f_{WS}, d^b_{WS} and d_{WS}, respectively. Finally, the output labels for index i are obtained as a softmax over the POS tagset, by projecting the combined state $\mathbf{l}_i$.

5.4.2 Experiments

Datasets For English, we conduct experiments on the Wall Street Journal of the Penn Treebank dataset [Marcus et al., 1993], using the standard splits (sections 1–18 for train, 19–21 for tuning and 22–24 for testing). We also perform tests on 4 other languages, which we obtained from the CoNLL shared tasks [Martí et al., 2007, Brants et al., 2002, Afonso et al., 2002, Atalay et al., 2003]. As for the other languages, we withdraw the last 100 sentences from the training dataset and use them for tuning.

Setup The POS model requires two sets of hyperparameters. Firstly, words must be converted into continuous representations and the same hyperparametrization as in language modeling (Section 5.3) is used. Additionally, we compare to the convolutional model of [Santos and Zadrozny, 2014], which also requires the dimensionality for characters and the word representation size, which are set to 50 and 150, respectively. Secondly, words representations are combined to encode context. Our POS tagger has three hyperparameters d^f_{WS}, d^b_{WS} and d_{WS}, which

correspond to the sizes of LSTM states, and are all set to 50. As for the learning algorithm, use the same setup (learning rate, momentum and mini-batch sizes) as used in language modeling.

Once again, we replace OOV words with an unknown token, in the setup that uses word lookup tables, and the same with OOV characters in the C2W model. In setups using pre-trained word embeddings, we consider a word an OOV if it was not seen in the labelled training data as well as in the unlabeled data used for pre-training.

Compositional Model Comparison A comparison of different recurrent neural networks for the C2W model is presented in Table 5.3. We used our proposed tagger in all experiments and results are reported for the English Penn Treebank. Results on label accuracy test set is shown in the column “acc". The number of parameters in the word composition model is shown in the column “parameters". Finally, the number of words processed at test time per second are shown in column “words/sec".

We observe that approaches using RNN yield worse results than their LSTM counterparts with a difference of approximately 2%. This suggests that while regular RNNs can learn shorter character sequence dependencies, they are not ideal to learn longer dependencies. LSTMs, on the other hand, seem to effectively obtain relatively higher results, on par with using word look up tables (row “Word Lookup"), even when using forward (row “Forward LSTM") and backward (row “Backward LSTM") LSTMs individually. The best results are obtained using the bidirectional LSTM (row “Bi-LSTM"), which achieves an accuracy of 97.29% on the test set, surpassing the word lookup table. The convolution model [Santos and Zadrozny, 2014] obtained slightly lower results (row “Convolutional (S&Z)"), we think this is because the convolutional model uses a max-pooling layer over series of window convolutions. As order is only perserved within windows, longer distance dependences are unobserved.

There are approximately 40k lowercased word types in the training data in the PTB dataset. Thus, a word lookup table with 50 dimensions per type contains approximately 2 million parameters. In the C2W models, the number of characters types (including uppercase and lowercase) is approximately 80. Thus, the character look up table consists of only 4k parameters, which is negligible compared to the number of parameters in the compositional model, which is once again 150k parameters. One could argue that results in the Bi-LSTM model are higher than

	acc	parameters	words/sec
Word Lookup	96.97	2000k	6K
Convolutional (S&Z)	96.80	42.5k	4K
Forward RNN	95.66	17.5k	4K
Backward RNN	95.52	17.5k	4K
Bi-RNN	95.93	40k	3K
Forward LSTM	97.12	80k	3K
Backward LSTM	97.08	80k	3K
Bi-LSTM $d_{CS} = 50$	97.22	70k	3K
Bi-LSTM	**97.36**	150k	2K

Table 5.3: POS accuracy results for the English PTB using word representation models.

those achieved by other models as it contains more parameters, so we set the state size $d_{CS} = 50$ (row "Bi-LSTM $d_{CS} = 50$") and obtained similar results.

In terms of computational speed, we can observe that there is a more significant slowdown when applying the C2W models compared to language modeling. This is because there is no longer a softmax over the whole word vocabulary as the main bottleneck of the network. However, we can observe that while the Bi-LSTM system is 3 times slower, it is does not significantly hurt the performance of the system.

Results on Multiple Languages Results on 5 languages are shown in Table 6.1. In general, we can observe that the model using word lookup tables (row "Word") performs consistently worse than the C2W model (row "C2W"). We also compare our results with Stanford's POS tagger (Row "Stanford"), with the default set of features, found in Table 6.1. Results using the tagger are comparable or better than state-of-the-art systems. We can observe that in most cases we can slightly outperform the scores obtained using their tagger. This is a promising result, considering that we use the same training data and do not handcraft any features. Furthermore, we can observe that for Turkish, our results are significantly higher ($>$4%).

Comparison with Benchmarks Most state-of-the-art POS tagging systems are obtained by either learning or handcrafting good lexical features [Manning, 2011, Sun, 2014] or using additional raw data to learn features in an unsupervised fashion. Generally, optimal results are obtained by performing both. Table 5.5 shows the

System	Fusional			Agglutinative	
	EN	PT	CA	DE	TR
Word	96.97	95.67	98.09	97.51	83.43
C2W	**97.36**	97.47	**98.92**	**98.08**	**91.59**
Stanford	97.32	**97.54**	98.76	97.92	87.31

Table 5.4: POS accuracies on different languages.

current Benchmarks in this task for the English PTB. Accuracies on the test set are reported on column “acc". Columns “+feat" and “+data" define whether hand-crafted features are used and whether additional data was used. We can see that even without feature engineering or unsupervised pretraining, our C2W model (row “C2W") is on par with the current state-of-the-art system (row “structReg"). However, if we add hand-crafted features, we can obtain further improvements on this dataset (row “C2W + features").

However, there are many words that do not contain morphological cues to their part-of-speech. For instance, the word *snake* does not contain any morphological cues that determine its tag. In these cases, if they are not found labelled in the training data, the model would be dependent on context to determine their tags, which could lead to errors in ambiguous contexts. Unsupervised training methods such as the Skip-n-gram model [Mikolov et al., 2013] can be used to pretrain the word representations on unannotated corpora. If such pretraining places *cat*, *dog* and *snake* near each other in vector space, and the supervised POS data contains evidence that *cat* and *dog* are nouns, our model will be likely to label *snake* with the same tag.

We train embeddings using English Wikipedia with the dataset used in [Ling et al., 2015b], composed of approximately 50 million tweets, and the Structured Skip-n-gram model [Ling et al., 2015b]. Results using pre-trained word lookup tables and the C2W with the pre-trained word lookup tables as additional parameters are shown in rows “word(sskip)" and “C2W + word(sskip)". We can observe that both systems can obtain improvements over their random initializations (rows “word" and (C2W)).

Finally, when using the C2W model in conjunction pre-trained word embeddings we also found that adding a non-linearity to the representations extracted from the C2W model $\mathbf{e}_w^C$ improves the results over using a simple linear transformation (row “C2W(tanh)+word (sskip)"). This setup, obtains 0.28 points over the previous state-of-the-art system(row “SCCN").

	+feat	+data	acc	improvement
word	no	no	96.70	0%
C2W	no	no	**97.36**	**0.68%**
word+features	yes	no	97.34	0.66%
C2W+features	yes	no	**97.57**	**0.90%**
Stanford 2.0 [Manning, 2011]	yes	no	97.32	0.64%
structReg [Sun, 2014]	yes	no	97.36	0.68%
word (sskip)	no	yes	97.42	0.74%
C2W+word (sskip)	no	yes	97.54	0.87%
C2W(tanh)+word (sskip)	no	yes	**97.78**	**1.12%**
Morče [Spoustová et al., 2009]	yes	yes	97.44	0.76%
SCCN [Søgaard, 2011]	yes	yes	97.50	0.82%

Table 5.5: POS accuracy result comparison with state-of-the-art systems for the English PTB.

5.4.3 Summary

In this chapter, we propose a C2W model that builds word embeddings for words without an explicit word lookup table. Thus, it benefits from being sensitive to lexical aspects within words, as it takes characters as atomic units to derive the embeddings for the word. On POS tagging, our models using characters alone can still achieve comparable or better results than state-of-the-art systems, without the need to manually engineer such lexical features. Although both language modeling and POS tagging both benefit strongly from morphological cues, the success of our models in languages with impoverished morphological cues shows that it is able to learn non-compositional aspects of how letters fit together.

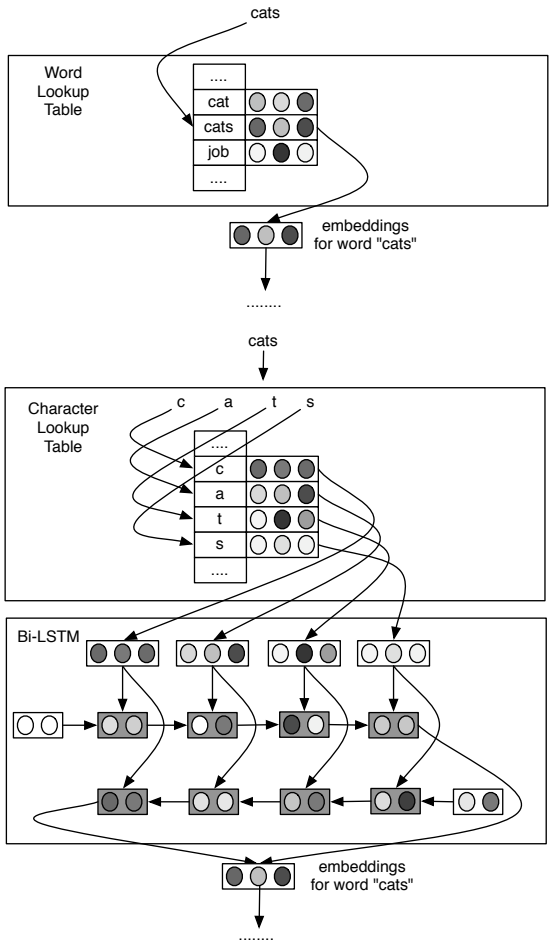

Figure 5.1: Illustration of the word lookup tables (top) and the lexical Composition Model (bottom). Square boxes represent vectors of neuron activations. Shaded boxes indicate a non-linearity.

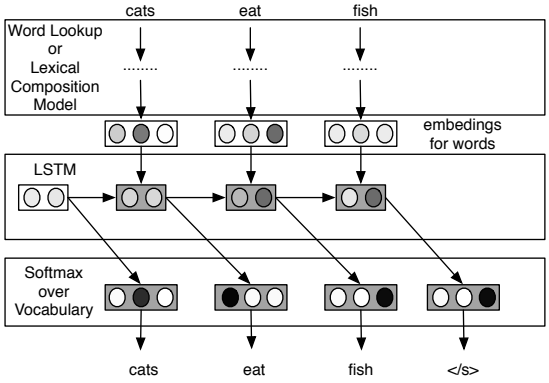

Figure 5.2: Illustration of our neural network for Language Modeling.

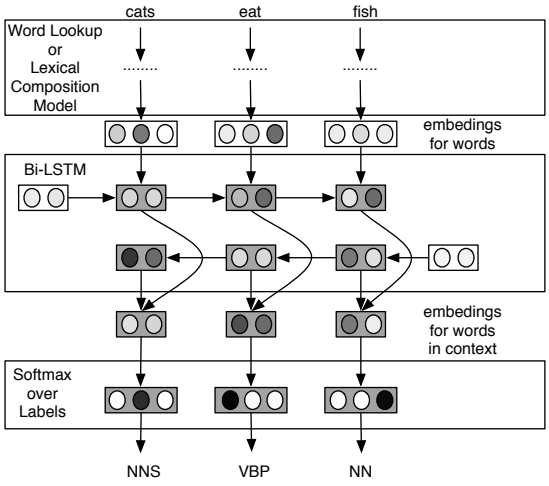

Figure 5.3: Illustration of our neural network for POS tagging.

Chapter 6

Character-based Word Generation for NLP

Chapter 5 describes a model to generate word representations from characters. That is, from a string input that represents a word, we learn a vector representations that encodes the traits for that word that are relevant to solving a given problem. However, the reverse process is also needed for many applications. For instance, in a neural language model, the model predicts a word based on the previous context. This context is generally quantified as a continuous vector obtained by the composition of all previous words in recurrent networks [Mikolov et al., 2010] or using a fixed size window [Mnih and Hinton, 2009]. In neural machine translation models [Kalchbrenner and Blunsom, 2013, Bahdanau et al., 2015], this step is also necessary to generate the translated words. In translation models, the generated word is still dependent on the target sentence context, which are the words that have been translated so far, but also the source sentence, which can also be mapped into a continuous vector.

Thus, the general problem we wish to solve is to produce a string, given an input vector. A widely known solution to this problem is to use a softmax function over words, a class-based logistic regression unit, trained to maximize the probability over all words in the vocabulary. However, a word softmax requires a separate set of parameters for each word type. Thus, a word softmax cannot generate unseen words in the training set, and requires a large amount of parameters due to the fact that each word type must be modelled independently. Furthermore, another well know problem is that the whole target vocabulary T must be traversed for each prediction during both training and testing phases. While at training time

approximations such as noise contrastive estimation [Gutmann and Hyvarinen, 2010] can be applied, traversing T is still required at test time. Other options include tree-based softmax [Morin and Bengio, 2005], which in general lead to suboptimal results compared to the regular word softmax.

In this chapter, we shall propose a character-level word generation model that generates words one character at time. The model is still dependent on the given input vector, but instead of predicting the probability distribution over words in a single matrix projection over the whole vocabulary, our model predicts the sequence of characters of the observed word. Our model benefits from the potential to generate unseen words, so long as all the characters of that unseen word have been seen. Furthermore, as the number of character types in the vocabulary is orders of magnitude smaller than the number of word types, our model benefits from a shorter prediction time, even though the number of predictions is higher. Finally, similarly to character-based word representations, the fact that we do not keep word type parameters allows us to build more compact models, as only character-based parameters are needed.

Then, the model shall be used in conjunction with the C2W model to build a fully character-based system for language modeling and machine translation by composing input words from their character sequence and generating output words character-by-character.

6.1 V2C Model

An illustration of the V2C (vector to characters) is shown in Figure 6.1. We define a character vocabulary for the language T_c. The input of our model is an arbitrary vector $\mathbf{i}$, which is generally the result of another part of the network. We wish to maximize the probability of the observed word w defined as a sequence of characters $w^0, \ldots, w^y$. To represented this, we involve the chain rule for probabilities:

$$P(w|\mathbf{i}) = \prod_{q \in [0,y]} P(w_q \mid w_0, \ldots, w_{q-1}, \mathbf{i})$$

That is, rather than learning to predict single words, our model predicts the character sequence of the output word. Each prediction is dependent on the input of the model, $\mathbf{i}$, and also on the previously generated character context $w_0, \ldots, w_{i-1}$.

Essentially, our model is writing the word character by character from left to right, and as new characters are written, these will condition the prediction of the following characters. We also assume that words end at w_y with a special *EOW* token. At training time, this token is simply added to all observed words. At test time, characters are generated until the *EOW* character is predicted.

The character context is computed by an LSTM. First, we project each character $w_0, \dots, w_{q-1}$ with a character lookup table into a $d_{t,c}$-dimensional vector $\mathbf{t}_{j,0}, \dots, \mathbf{t}_{j,q-1}$. Then, each vector is concatenated to the input vector $\mathbf{i}$, and passed as the input to an LSTM, generating the sequence of states $\mathbf{y}^f_0, \dots, \mathbf{y}^f_{q-1}$. Then, the prediction of the character w_q is obtained as the softmax function:

$$P(w_q|w_0, \dots, w_{q-1}, \mathbf{i}) = \frac{\exp(S_y^{w_q}\mathbf{y}^f_{q-1} + \mathbf{b}_{w_q})}{\sum_{i\in[0,T_c]} \exp(S^i_y\mathbf{y}^f_{q-1}) + \mathbf{b}_i},$$

where S_y are the parameters that convert the state y into a score for each output character, and S^i_y denotes the parameters respective to the character type i.

Finally, the model is also required to produce the end of sentence token *EOS*, similarly to a word softmax. For instance, in translation models, a EOS token is used to define the end of sentence, which is required at decoding time to know when the translation is finished. In our model, we simply consider the *EOS* token as a word whose only character is *EOS*. This way, no special handing is needed for this token.

The hyper parameters of our model are essentially the LSTM parameters (cell and state size) y_{cell} and y_{state}. As for the parameters, the V2W model requires an LSTM $\mathbf{Y}_{LSTM}$ and the parameters for the character softmax S_y.

6.2 Character to Character Language Modeling

Neural language models have shown to obtain state-of-the-art results on many tasks [Bengio et al., 2003, Mnih and Hinton, 2008]. Most of these models operate at the word level, with the option to add additional subword units for morphological awareness, and the ability to handle out-of-vocabulary words [Larson, Kozielski et al., 2013, 2014]. Many of these approaches segment words into smaller units, which are then used to train either an n-gram language model or a recurrent language model. The main benefit for using subword units is the fact that these

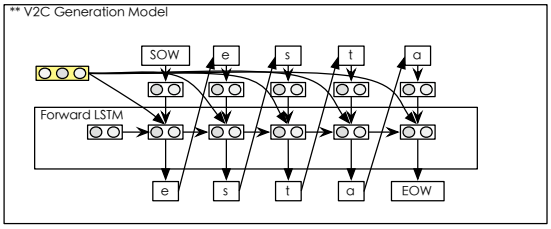

Figure 6.1: Illustration of the V2C model. Square boxes represent vectors of neuron activations.

allow better handling of unknown words, which must be accounted for in open vocabulary systems.

Character-based language models offer an appealing strategy for dealing with the need to model completely open-vocabulary language. In addition to being able to generate completely novel word forms, they do not require the model to keep a list of word types paired with parameters, which can substantially reduce the amount of RAM required to represent the model, potentially allowing models trained on larger datasets to be used in smaller devices. A second benefit of character models is computational tractability: as the execution time of the softmax over the whole vocabulary grows linearly as the vocabulary size increases, character models that predict characters rather than words tend be significantly more efficient at training and test time, as the number of character types tends to be orders of magnitude smaller than word types (with some exceptions, such as Mandarin).

Unfortunately, despite these benefits, character-only models approaches have been unsuccessful in exceeding the quality of the results of word-based approaches when used in stand-alone applications. Hybrid solutions that combine character-based language models with word-based models [Ali et al., 2012] overcomes this problem in terms of the quality of the end-to-end results, but the computational benefits of character-based models (i.e., speed and compactness) are lost when the character-based model is combined with a word-based model as the word-based model would be the bottleneck of the system as a whole.

Thus, we shall propose a **hierarchical neural language model** that uses characters as atomic input and output units. Firstly, we compose character representations into words representations using the character to word (C2W) model proposed in [Ling et al., 2015c]. Then, word representations are combined with a regular

LSTM-based recurrent neural network [Sundermeyer et al., 2012b], thereby encoding cross word (context) information. Finally, output words are predicted using the (V2W) model. As both the word composition and prediction models are performed at the character level, our models tend to be smaller and faster than equivalently performing word-based models. We show that our model achieves better results than both purely word-based and character-based approaches when used as features for automatic speech recognition re-ranking even in closed vocabulary scenarios, where the OOV prediction problem for word-based neural networks does not hold.

6.2.1 Experiments

We evaluate the quality of our method extrinsically by training our language models to score the n-best outputs generated by an ASR system. We chose this evaluation method over an intrinsic evaluation using perplexity, as in many cases, the results on perplexity are inconclusive on the quality of the model in end-to-end tasks [Clarkson and Robinson, 1999].

Dataset

Evaluation is performed using a TED Talk subtitles dataset[1], containing approximately 160 hours of TED Talks audio, with 1.8 million words. Out of the 1,382 talks, 5 were selected for development and 5 for testing. The subtitles were tokenized and lowercased. In the pre-processed training data, we found 36,782 word types and 33 character types, composed by the Latin alphabet and punctuation.

ASR System

The speech recognizer used in our experiments is AUDIMUS, a hybrid HMM-MLP system which uses the WFST approach to integrate knowledge sources, such as the language model and the lexicon with the neural network-based acoustic models. AUDIMUS combines multiple feature streams (RASTA, PLP and MSG) for improved robustness to different acoustic conditions.

The acoustic model was trained using 160 hours of TED Talks audio. The subtitles for each talk were force aligned with the audio to produce the targets used

[1] http://www.ted.com

for training. The language model used in the ASR was built using a 4-gram model language model with Kneser-Ney smoothing. It results from the interpolation of the model trained with the in-domain TED Talk data and another language model built using the out-of-domain data from Europarl [Koehn, 2005], which contains approximately 50 million words.

We also restrict the vocabulary of the ASR to the word types in the in-domain datasets, as an open vocabulary experiment would benefit character-based models, which is not the goal of our work.

Neural Language Models

Neural language models are trained on the in-domain TED Talk data. As baseline, we train a word-based LSTM recurrent neural network [Sundermeyer et al., 2012b]. We set the word vector size K to 50, resulting in a word lookup table with approximately 1.8 million parameters. We set the language model state size S to 150, as well as for the cell size. This leads to an LSTM with 187 thousand parameters. Finally, the softmax over vocabulary projects each state $\mathbf{s}$ to the whole vocabulary V, which requires 5.5 million parameters. Thus, in total, the model contains approximately 7.5 million parameters. We shall denote this model as “Word LM".

Next, we train our hierarchical model, which uses the same parameters for K and S. Instead of the word lookup table, we train the C2W model, with a bidirectional LSTM with 150 dimensions for the state and cell units. As there are only 33 character types, we set the character vector size P to 20 leading to only 660 parameters for the character lookup table. Then the BLSTM trained for the C2W model contains approximately 340 thousand parameters. As we did not change K and S, the LSTM encoding the context state $\mathbf{s}$ still holds 187 thousand parameters. Finally, for the V2C model, we set its LSTM state and cell sizes to 150. Then, the character softmax only requires 4950 parameters, while the LSTM itself requires 417 thousand parameters. Thus, in total the model only requires approximately 1 million parameters and shall be named “Hierarchical LM".

We also attempt to build a pure character-based LSTM model by simply considering each character a different input. As the character-based model requires longer range dependencies, we set the state S to 400 units. This leads a to a model with approximately 1.1 million parameters. We shall refer to this model as “Character LM".

	None	Word LM	Character LM	Hierarchical LM
WER ACC+LM	23.87	23.64	23.61	**23.53**
WER ACC	26.98	26.34	26.15	**26.00**
LM parameters	n/a	7.5M	1.1M	**1.0M**
LM words/sec	n/a	70	938	**1,223**

Table 6.1: Re-ranking results and neural language model statistics.

Training

All models were trained using adaptive gradient descent [Duchi et al., 2011], with the learning rate 0.2, and mini-batches of 10 sentences. As stopping criteria, we employ early stopping by checking on the perplexity on the development data every epoch. Training stops when the perplexity does not improve for 5 epochs.

As for re-ranking the ASR, we generate a list of 200 n-best possible outputs for each utterance. The ASR score $score_{asr}$ is obtained from the combination of the acoustic score and the native language model score. We simply linearly combine it with the score obtained from the neural language model $score_{nlm}$, as $\alpha score_{nlm} + (1 - \alpha) score_{asr}$. That is, we estimate an α parameter that minimizes the WER over the development set. As only one parameter is needed for this combination, we simply performed a grid search over all possible values for this parameter at intervals of 0.001, and chose the value that minimizes the word error rate on the development set.

Results

Results are shown in Table 6.1. We observe that even though our hierarchical model (column “Hierarchical LM") uses less than 15% of the parameters (row “LM parameters") in the word-based model (column “Word LM") it actually achieves a better word error rate (row “WER ACC+LM"). While improvements are small (0.11 over the word-based model), it is important to notice that these results are achieved without explicitly representing words in our model, but only characters. Furthermore, we can observe that our Hierarchical model achieves a significantly faster decoding rate (row “LM words/sec"), as it performs the softmax over character types, which are orders of magnitude smaller than word types.

Compared to a purely character-based language model (column “Character LM"), we observe that our model performs slightly better in terms WER, using slightly less parameters.

However, it is still not certain that the character-based models can model words without a word-based model, as the ASR system contains a native n-gram based language model. This means that we are essentially interpolating a character-based model with a word-based model. Thus, we removed the score of the native language model from the score produced by the ASR and repeated this experiment. While this degrades the overall results by approximately 3% (row “WER ACC"), we can still observe improvements using our hierarchical model (column “Hierarchical LM").

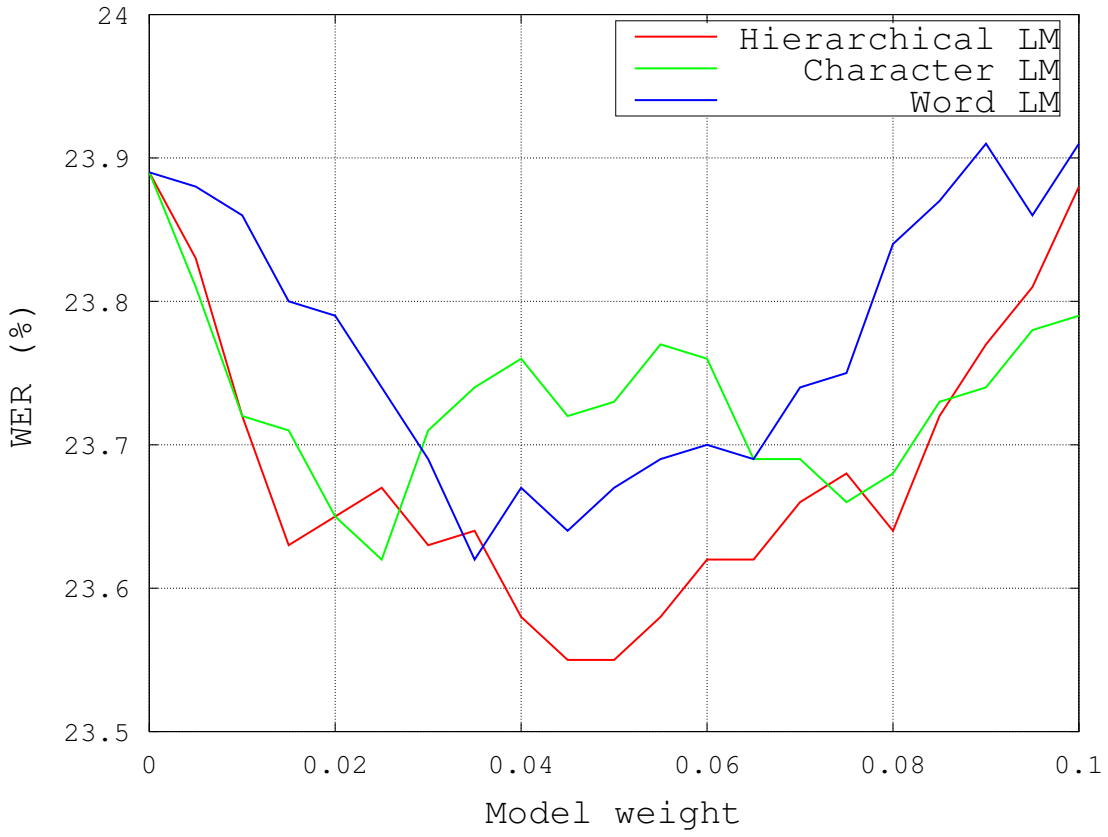

Figure 6.2: WER values for different values of α using the acoustic and native language model scores.

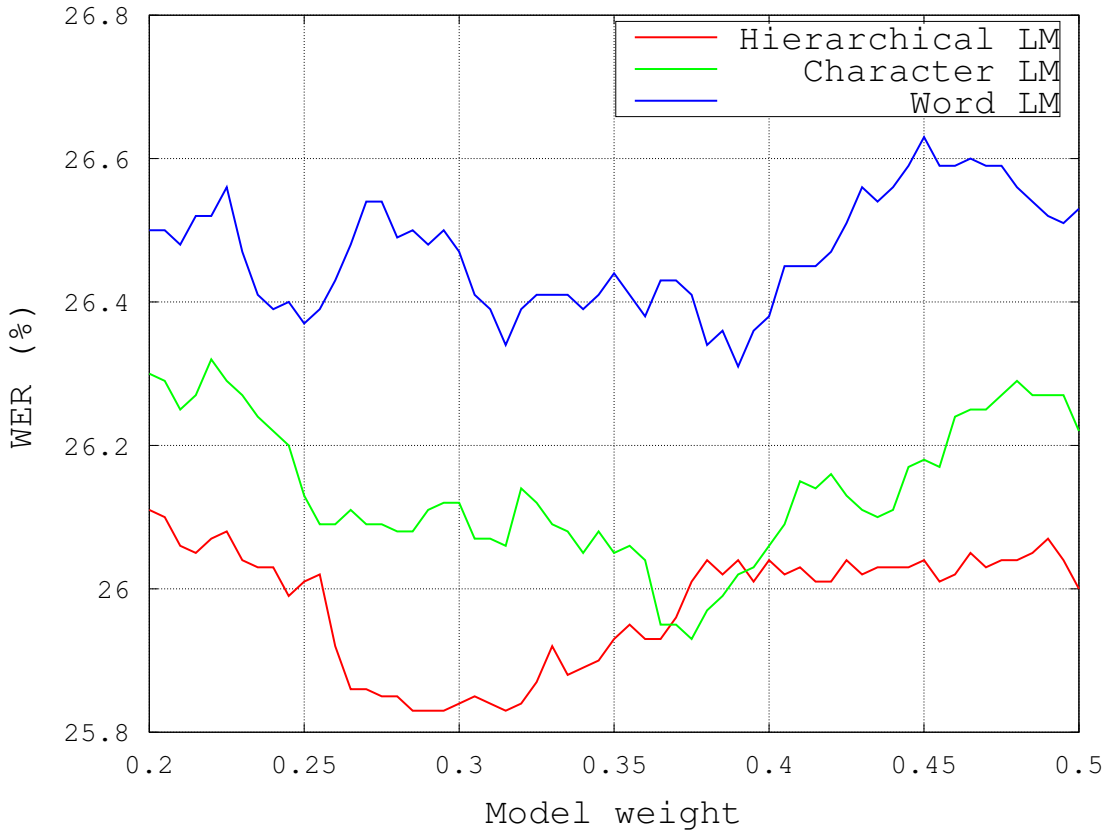

Figure 6.3: WER values for different values of α using the acoustic score.

Figures 6.2 and 6.3 illustrate the WER on the test set for different values of the interpolation weight (higher values indicate that the neural language model is given higher weight). These are computed by re-ranking the full ASR system, and the ASR system without the native language model score, respectively. We can observe that character-based model (“Character LM") performs better than the word-based model (“Word LM") for most values of α when no native language model is used, suggesting that word-based neural language models trained on smaller datasets are more likely to overfit the data compared to character-based models. Yet, we show that when the native language model trained on larger datasets is used, the interpolation of both models can lead to better estimates than the character-based models. On the other hand, we can observe that in general our hierarchical model

("Hierarchical LM") obtains better WER values for most ranges of the interpolation weight compared to other models.

6.3 Character to Character Machine Translation

In Section 4.2 we presented a model to normalize input words, so that variants, such as *cooool* are normalized to their standard forms. However, the main source of this problem is that current MT systems do not use word representations that are not lexically aware. Thus, effort is needed on normalizing the input prior to translation. In this section, we shall describe our character-level machine translation model by applying both the character-based representation model (C2W) and the character-based softmax model (V2C). This allows the model to inherently to learn translations for words based on their orthographic properties. That is, we wish to build models that can represent *coool* and *cool* with similar orthographic representations so that when learning the translation of *cool*, translations for other orthographic variants are learnt as well.

Current machine translation models model words as independent units. Thus, the model is unable to capture lexical cues in the source language and generate unseen words in the target language. For instance, the phrase pairs *mountain*→*montanha* and *mountains*→*montanhas*, encoding the translation of the English words *mountain* and *mountains* to the Portuguese words *montanha* and *montanhas*, are essentially independent from each other in conventional models. Thus, the generalization capabilities of the model are hampered, as it fails to learn the morphological processes governing the two languages. For instance, the model would not be able to translate *rivers* to the perfectly regular word *rios* if it was not seen in the training data, even if a translation of *river* to *rio* is available.

There are clearly benefits in approaching the translation problem from a sub-lexically oriented view. In morphologically rich languages (e.g. German and Turkish), segmenting the source language using morphological analysers can substantially improve translation by improving the coverage of the translation system [Dyer, 2007]. As for the target language, morphological variants can be obtained using models that predict inflected forms [Chahuneau et al., 2013]. In language pairs where there are systematic mappings between source and target characters, new translations can be learnt by learning this mapping process. Some concrete examples include the work in cognate detection [Beinborn et al., 2013] and transliteration [Kang and Choi, 2000]. Finally, in noisy domains, such as Twitter,

it is common to normalize lexical variants in order to improve the coverage of the model as we described in Chapter 4.

In this section, we introduce a character-based neural machine translation model, where the representation and generation of the source and target words are performed solely at the character-level. However, we still incorporate the knowledge of words into the model by defining a hierarchical model that generates the source word representations from characters, maps the word representation into the target space and then proceeds to generate the target word character-by-character. As the composition of the words is based on characters, the model can learn morphological aspects in the source language, allowing it to build representations for unseen words. On the other hand, the character-based generation model allows the model to output words that have not been observed. As both composition and generation of word types is performed at the character level, the model only keeps parameters for source and target characters, which allows it to learn the translation process using an extremely small number of parameters. This allows the model to be run in machines with low memory resources, such as mobile phones.

Previous work [Vilar et al., 2007] shows that a purely character-based system using phrase-based systems that directly maps the source character sequence to the target character sequence would significantly underperform a word-based system. Thus, we propose a hierarchical models that learns to map words from characters, but the mapping between the source and target languages is performed at the word level. That is, we not allow direct dependencies between characters in the source and target character sequences, nor direct dependencies between source words. This makes our model less computationally expensive, as most operations such as alignment are only performed on the word level rather than on the character level. Furthermore, a model using only characters would need substantially larger amounts of training data to generalize, and in many language pairs or domains, this data is scarce. Finally, while our model does not hold any direct dependencies between characters in different words, the words themselves are connected. As the word representations are generated from characters, such dependencies can still be learnt in an indirect manner.

6.3.1 Character-based Machine Translation

In this section, we shall describe our character-based machine translation (CMT) approach. The model translates a source sentence $s = s_0, \ldots, s_n$, where s_i is the source word at index i into the target sentence $t = t_0, \ldots, t_m$, where t_i is the

target word at index i. This can be decomposed into a series of predictions, where the model predicts the next target word t_p, given the source sentence s and the generated target words $t_0, \ldots, t_{p-1}$. In this section, we shall describe the inner workings of our CMT model.

Notation As for notation, we shall represent vectors with lowercase bold letters (e.g. $\mathbf{a}$), matrixes with uppercase bold letters (e.g. $\mathbf{A}$), scalar values as regular lowercase letters (e.g. a) and sets as regular uppercased letters (e.g. A). When referring to whole source and target sentences, we shall use the variables s and t, respectively. Individual source and target words, shall be referred as s_i and t_j, where i and j are the indexes of the words within the sentence. Furthermore, we use variables n and m to refer to the lengths of the source and target sentences. Finally, we shall define that s_0 and t_0 represent a special start of sentence token denoted as *SOS* and that s_n and t_m represent a special end of sentence token denoted as *EOS*. When we wish to refer to individual characters, we shall use the notation $s_{i,u}$ and $t_{j,v}$, which refers to the u-th character in the i-th word in the source sentence and to the v-th character in the j-th word in the target sentence, respectively. We use the variables x and y as the lengths of the source and target words. We also define that the first character within a word is always a start of word token denoted as *SOW* and the last characters $s_{i,x}$ and $s_{j,y}$ are always the end of word character *EOW*.

Joint Alignment and Translation Model

We shall first describe the word based neural approach we use as baseline, which is defined in [Bahdanau et al., 2015]. An illustration of the model is shown in Figure 6.4. In this model, the translation of a new target word t_p, given the source sentence $s_0, \ldots, s_n$ and the current target context $t_0, \ldots, t_{p-1}$ is performed in the following steps.

1. **Source Word Projection** As source words are a series of strings, they must be mapped into continuous vectors in order to be processed by the rest of the model. The most common approach to perform the projection from words to vectors is to use a lookup table, where each word type is attributed an independent set of parameters. These correspond to a vector of $d_{s,w}$ dimensions, where $d_{s,w}$ is the dimensionality of the word vectors, defined as a hyperparameter. This means that the set of parameters in the source lookup

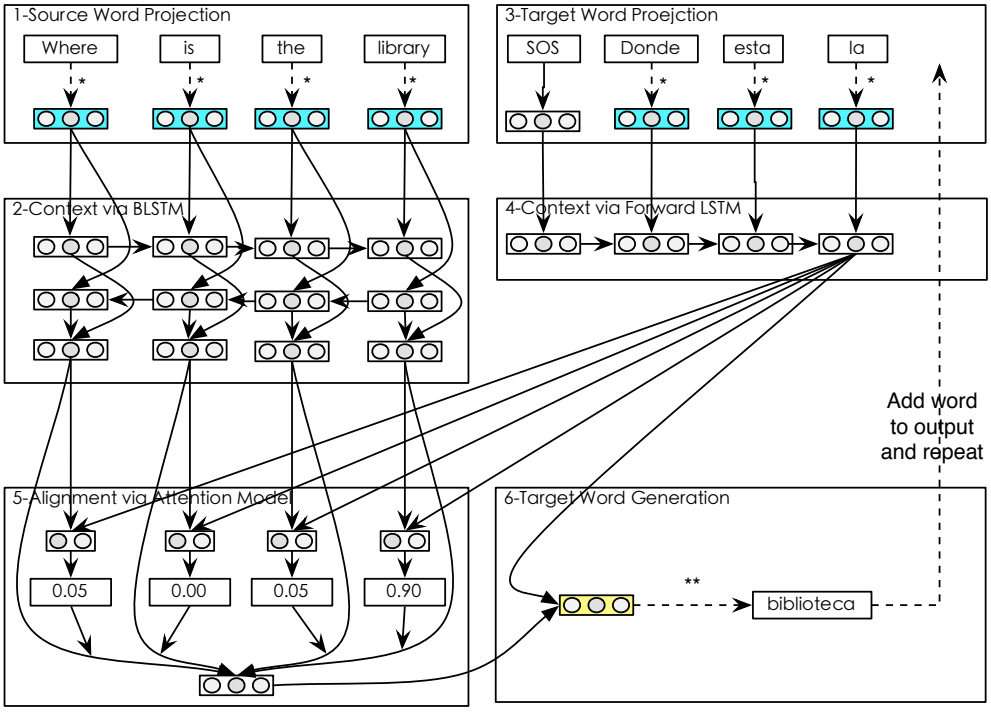

Figure 6.4: Illustration of the joint alignment and translation model. Square boxes represent vectors of neuron activations.

table S_{lookup} will contain $d_{s,w} \times S$ parameters, where S is the number of word types in the source language.

Thus, this step requires the definition of a single hyper-parameter $d_{s,w}$ and the parameters $S_{lookup}^{d_{s,w} \times S}$. As input, it takes a sequence of source words $s_0, \ldots, s_n$ and outputs the projection of the source words $\mathbf{s}_0, \ldots, \mathbf{s}_n$.

As an alternative to word lookup tables, we use the C2W model described in Chapter 5. That is, similarly to word lookup tables, the C2W model takes a word string s_j as input and produces a vector $d_{s,w}$-dimensional vector for that word, but rather than allocating parameters for each individual word type, the word vector $\mathbf{s}_j$ is composed by a series of transformation using its character sequence $s_{j,0}, \ldots, s_{j,x}$.

2. **Context via BLSTMs** The information contained within each projected source

word $\mathbf{s}_0, \ldots, \mathbf{s}_n$, does not hold any information regarding their position in the sentence, which is limitative for many reasons (e.g. determining the sense of ambiguous words). While there are many ways for encoding this information, such as convolutional and window-based approaches [Collobert et al., 2011], a popular approach is to use recurrent neural networks to encode global context. We use long short memory RNNs (LSTMs), which are similar to recurrent neural networks in that given a set of input vectors $\mathbf{i}_0, \ldots, \mathbf{i}_k$, it produces a set of states $\mathbf{h}_0, \ldots, \mathbf{h}_k$, where each state $\mathbf{h}_j$ is the result of the composition between the current input $\mathbf{i}$ and the previous state $\mathbf{h}_{j-1}$. Thus, a context-aware representation of each source word vector $\mathbf{s}_0, \ldots, \mathbf{s}_n$ is obtained by using two LSTMs. We build a forward LSTM by building the state sequence $\mathbf{g}_0^f, \ldots, \mathbf{g}_n^f$ using the input sequence $\mathbf{s}_0, \ldots, \mathbf{s}_n$. Then, we build a backward LSTM, by feeding the input sequence in the reverse order $\mathbf{s}_n, \ldots, \mathbf{s}_0$ generating the backward state sequence $\mathbf{g}_n^b, \ldots, \mathbf{g}_0^b$. Consequently, for each word s_i, the forward state $\mathbf{g}_i^f$ encodes s_i with its left context, while the backward state $\mathbf{g}_i^b$ encodes s_i in its right context. To encode the global context, we combine the forward and backward states

$$\mathbf{b}_i = \mathbf{G}_f \mathbf{g}_i^f + \mathbf{G}_b \mathbf{g}_i^b + \mathbf{g}_b,$$

where $\mathbf{G}^f$, $\mathbf{G}^b$ and $\mathbf{b}_b$ are parameters that determine how the states are combined.

In summary, this module defines the hyper parameters of the BLSTM, which are essentially the state g_{state} and cell g_{cell} sizes. The parameters of the BLSTM are the parameters governing the updates for the cells and states, based on the input and previous cell and state, which we shall denote as G_{blstm} for simplicity. Finally, the combination of the forward and backward states is parametrized by $\mathbf{G}_f$, $\mathbf{G}_b$ and $\mathbf{g}_b$. This module takes as input a sequence of source word vectors $\mathbf{s}_0, \ldots, \mathbf{s}_n$ and produces a set of context-aware vectors $\mathbf{b}_0, \ldots, \mathbf{b}_n$.

3. **Target Word Projection** The projection of target words essentially uses the same model as the projection of source words, with a different set of parameters. Thus, the hyper parameter $d_{t,w}$ refers to the size of the target language word vectors, and the lookup table $T_{lookup}^{d_{t,w} \times T}$, defines the parameters needed to project the words in the target vocabulary T. As input, it takes a sequence of target words $t_0, \ldots, t_n$ and outputs the projection of the source words $\mathbf{t}_0, \ldots, \mathbf{t}_n$.

Once again, the C2W model can be used as alternatives to word lookup tables.

4. **Context via Forward LSTM** Unlike the source sentence s, the target sentence is not known a priori. Thus, only a forward LSTM is built over the translated sentence $t_0, \ldots, t_{p-1}$, generating states $\mathbf{l}^f_0, \ldots, \mathbf{l}^f_{p-1}$. Similarly to a language model, for the prediction of word t_p, we use state $\mathbf{l}^f_{p-1}$, which encodes the left context prior to the predictions of t_p. This model is simply an LSTM, which is hyper-parametrized by the state and cell sizes, l_{state} and l_{size}, and defines the update parameters L_{lstm}. As input, the model receives the currently translated target words $t_0, \ldots, t_{p-1}$ and returns the last state of the model encoding the current target context $\mathbf{l}^f_{p-1}$.

5. **Alignment via Attention**

   For each target context $\mathbf{l}^f_{p-1}$, the attention model learns the attention that is attributed to each of the source vectors $\mathbf{b}_0, \ldots, \mathbf{b}_n$. Essentially, the model computes a score z_i for each source word by applying the following function:

$$z_i = \mathbf{s} \odot \tanh(\mathbf{W}_t \mathbf{l}^f_{p-1} + \mathbf{W}_s \mathbf{b}_i),$$

   where the source vector $\mathbf{b}_i$ and target vector $\mathbf{l}^f_{p-1}$ are combined into a d_s-dimensional vector using the parameters $\mathbf{W}_s$ and $\mathbf{W}_t$, respectively, and a non-linearity is applied. In our case, we apply a hyperbolic tangent function $\tanh$. Then, the score is obtained using the vertical vector $\mathbf{s}$. This operation is performed for each source index obtaining the scores $z_0, \ldots, z_n$. Then, a softmax over all scores as is performed as follows:

$$a_i = \frac{\exp(z_i)}{\sum_{j \in [0,n]} \exp(z_j)}$$

   This yields a set of attention coefficients $a_0, \ldots, a_n$, with the property that each coefficient is a value between 0 and 1, and the sum of all coefficients is 1. In MT, this can be interpreted as a soft alignment for the target word t_p over each of the source words. These attentions are used to obtain a representation of the source sentence for predicting word w_p, which is simply the weighted average $\mathbf{a} = \sum_{i \in [0,n]} a_i \mathbf{b}_i$. In contrast, alignment in standard lexical translations are latent variables rather than just a weighting scheme.

Thus, the attention model defines the hyper parameter d_z, which is the size of the hidden layer that combines $\mathbf{l}^f_{p-1}$ and each $\mathbf{b}_i$. Then, the parameters $\mathbf{W}_s$ and $\mathbf{W}_t$ project the input vectors into this hidden layer and the vertical vector s that scores this hidden layer. The attention model takes as input the target context vector $\mathbf{l}^f_{p-1}$ and a set of source vectors $\mathbf{b}_0, \ldots, \mathbf{b}_n$, and returns a vector $\mathbf{a}$, which is the average of all source vectors $\mathbf{b}_0, \ldots, \mathbf{b}_n$, weighted by their estimated attentions $a_0, \ldots, a_n$.

6. **Target Word Generation** In the previous steps, we obtain two vectors that contain important information for the prediction of the next translated word t_p. The first is the target context vector $\mathbf{l}^f_{p-1}$, which encodes the information of all words preceding t_p, and $\mathbf{a}$ which contains the information of the most likely source word/words to generate w_p. All that is left is to actually generate t_p. This is accomplished using the V2C model we propose in this chapter, or using the standard word softmax for a word-based generation model.

Training

During training the whole set of target words $t_0, \ldots, t_m$ are known, so we simply maximize the log likelihood of the sequence of words $\log P(t_0|\mathbf{a}, \mathbf{SOS}), \ldots, \log P(t_m|\mathbf{a}, \mathbf{l}^f_{m-1})$. More formally, we wish to maximize the log likelihood of the training data defined as:

$$\sum_{(s,t)\in D} \sum_{p\in[0,m]} \log P(t_q|\mathbf{a}, \mathbf{l}^f_{m-1})$$

where D is the set of parallel sentences used as training data.

Using a regular softmax, optimizing such a function simply requires the derivation of the softmax function and the propagation of the gradient values. Using the W2C model, we shift the log probability $\log p(t_q \mid \mathbf{a}, \mathbf{l}^f_{m-1})$ to:

$$logP(t_q|\mathbf{a}, \mathbf{l}^f_{m-1}) = \log P(t_{p,q}|t_{k,0}, \ldots, t_{k,q-1}, \mathbf{a}, \mathbf{l}^f_{m-1})$$

Thus, we now wish to maximize the log likelihood that each character is predicted correctly.

Rather than normalizing over the whole character inventory, we use noise contrastive estimation [Gutmann and Hyvarinen, 2010], which subsamples a set of K negative samples at each prediction.

Decoding

Similarly to previous work [Bahdanau et al., 2015], decoding is performed using beam search. In the word-based approach, we define a stack of translation hypothesis per timestamp $\mathbf{A}$, where each position is a set of translation hypotheses. Starting with $\mathbf{A}_0 = SOS$, at each timestamp j, we condition on each of the previous contexts $t = \mathbf{A}_{j-1}$ and add new hypothesis for all words obtained in the softmax function. For instance, if $\mathbf{A}_1 = I, you$, and the vocabulary $T = I, you, go$, then $\mathbf{A}_2$ would be composed by "I I", "I you", "I go", "you I", "you you", "you go". We set a beam k_w, which defines the number of hypotheses to be expanded prioritizing hypothesis with the highest sentence probability. The search stops once all translation hypothesis generate the end of sentence token, or if the timestamp reaches 200. Then, the highest hypothesis that generated an end of sentence token is returned. If this is not possible, an empty string is returned.

Whereas the word softmax simply returns a list of candidates for the next word by iterating through each of the target word types, the V2C model can generate an infinite number of target words. Thus, in the character-based MT model, a second decoding process is needed to return the list of top scoring words at each timestamp. That is, we define a second beam search decoder with beam k_c, and perform a stack-based coding on the character-level for each word prediction. The decoder defines a stack $\mathbf{B}$, where at each timestamp, a new character is generated for each hypothesis. Once again, the search stops once all hypothesis reach their final state by generating the end of word tag *EOW*. Then, all hypothesis that reach the final state are returned as a list of translation options. If no hypothesis reaches a final state, we simply return a special string "NA" as the only option.

6.3.2 Experiments

We report on machine translation experiments in the English-Portuguese language pair in two domains: parliament data and Twitter data in the English-Portuguese language pair.

Data

The parliament data was drawn from Europarl [Koehn, 2005], where we have drawn 600k sentence pairs for training, 500 sentence pairs for development and 500 sentence pairs for testing. For the Twitter data, we manually translated the dataset provided in [Gimpel et al., 2011] from English into Portuguese. We kept the splits used in their Part-of-speech-tagging experiments (1k training sentences, 327 development sentences and 500 sentences).

Setup

Both languages were tokenized with the tokenizer provided in [Owoputi et al., 2013] as it has rules for Twitter specific artifacts, such as hashtags. Training is always performed using both the Europarl and Twitter datasets. In total, there are 248 character types and 64.976 word types in the English training data, and 257 character types and 97.964 word types in the Portuguese training data. Both word-based and character-based MT neural systems are trained using mini-batch gradient descent with mini-batches of 40 sentence pairs. Updates are performed with Adagrad [Duchi et al., 2011].

As for the hyper parameters of the model, all LSTM states and cells are set to 150 dimensions. Word projection dimensions for the source and target languages $d_{s,w}$ and $d_{t,w}$ were set to 50. Similarly, the character projection dimensions were also set to $d_{s,c}$ and $d_{t,c}$. For the alignment model, d_z was set to 100. Finally, the beam search at the work level k_w was set to 5, and the beam size for the character level search k_c was set to 5. Training is performed until there is no BLEU [Papineni et al., 2002] improvement on the tuning set for 5 epochs, and the epoch with the highest score is used.

As for decoding, we used a word-beam and a character-beam of 20. That is, at each timestamp the top 20 hypothesis are chosen.

We also use the Moses phrase-based MT system with standard features [Koehn et al., 2003]. For reordering, we use the MSD reordering model [Axelrod et al., 2005]. As the language model, we use a 5-gram model with Kneser-Ney smoothing. The weights were tuned using MERT [Och, 2003].

We did not run each experiment multiple times as each experiment using the character-based models required 2 months of training with 24 CPUS.

	Europarl		Twitter	
	DEV	TEST	DEV	TEST
Word-to-Word NN	19.21	19.39	5.94	6.72
Char-To-Char NN	19.30	19.57	6.69	7.23
Moses	20.49	20.80	6.51	7.11

Table 6.2: BLEU scores for different systems.

Results

BLEU results on different datasets are shown in Table 6.2. We can observe that the character-based model can obtain BLEU scores that are slightly higher than the word-based models.

One of the main factors for this improvement is due to the fact that neural network MT approaches tend struggle with the translation of OOV words. A popular approach used to address these problems is to simply transfer the OOV word form into the target sentence. As most OOV words tend to be numbers and named entities, these are generally not translated at across most languages. For instance, in Twitter, all the unknown hashtags and at mentions are not translated correctly by the word-based neural translation system, as the word softmax does not allow unknown words to be generated. On the other hand, character-based models can learn to simply transpose numbers, hashtags, web links and most named entities. In comparison to the phrase-based system, we can observe that our results are lower in the Europarl dataset, as most of the unknown words are handled correctly by transferring them directly, which is part of the decoding process in Moses. Yet, we can observe slight improvements in the Twitter dataset. This is because terms, such as *strongg*, are generally treated as unknown words in Moses, and translated as they are, while in the character-based system, this word has a similar vector representation to the standard word *strong*, which allows the model to translate this word correctly.

The number of parameters and computational speed of each model is represented in Figure 6.3. As for the number of parameters, we can observe that even though the number of parameters in alignment model (steps 2 and 5) are the same, the number of parameters needed for the word representation for the source and target languages differs radically. As we are using approximately half million sentence pairs, the large number of word types (64.976 English words and 97.964 Portuguese words), requiring 50 parameters each will lead to a word lookup table

	Parameters (Word Representation)	Parameters (Word Generation)	Parameters (Alignment)	Word/Sec
Word-to-Word NN	8M	19M	500k	113
Char-To-Char NN	800k	350k	500k	41

Table 6.3: Number of parameters and translation speed for different systems.

of approximately 8 million parameters. Similarly, for the word softmax, each of the 97.964 Portuguese word types will require 200 parameters. Thus, it would require 19 million parameters to implement this softmax. On the other hand, the C2W and V2C models require far less parameters as only a character lookup table is required. In this case, the C2W model, needs to model 248 English and 257 Portuguese characters with 50 parameters each, leading to 25k parameters. Then, the compositional model (2 LSTMs) require approximately 375k parameters for each language. Finally, the output word vector requires a 300×50 projection matrix. These amount to approximately, 800k parameters. Similarly, for the V2C model, only a character based lookup table in the target language is used, requiring 13k parameters. Then, the LSTM used to encode target character context requires 337k parameters to encode. This leads to a total of 350k parameters. Thus, in total, the word-based model requires 17.5M parameters while the character-based model only requires 1.65M parameters, which is only 10% of the word-based model.

In terms of computational speed, we can observe that the word-based model performs significantly slower than the character-based model, which replaces the word softmax with the character softmax. The difference is not as wide as in the language modeling experiments in Section 6.2.1, as a character-level decoding process is required to produce each word.

6.4 Summary

We presented a character-based alternative to predict and generate words, by predicting the characters of the word sequentially. As an alternative to a word softmax, it allows prediction and decoding to be performed more efficiently and with less parameters. Furthermore, it can generate unseen words in the training set. In this sense, our model behaves similarly to that found in [Sennrich et al., 2015], which defines a translation model for rare words using characters or subword units. We presented a character-based machine translation model that represents and gener-

ates sentences at the character level. Thus, our model can learn representations that take into transformation at the subword level during translation. This allows our model to excel at domains such as Twitter, where lexical variants are commonly used, as well as in morphological rich languages, where words are composed from multiple subunits. Experiments on the English-Portuguese language pair, in both parliament and Twitter data indicate that our model can perform better than its a word-based counterpart with only a fraction parameters. Furthermore, our work has been applied to dependency parsing, successfully in [Ballesteros et al., 2015].

Chapter 7

Conclusion and Future Work

In this chapter, we summarize our contributions and highlight some open problems, suggesting possible new directions for future research.

In this thesis, we address the problem of modeling language with considerable lexical variation, such as found in Twitter and Sina Weibo, focusing in particular on machine translation but also looking at applications in several other related subtasks, such as language modeling. For considerable improvements to be obtained in any task, two conditions must be met.

Firstly, large amounts of labelled data must be available as more data is crucial for obtaining better results. In this regard, while much parallel data can be obtained in machine translation, these are generally obtained from edited sources, such as Web documents [Resnik and Smith, 2003]. As the language in microblogs differs substantially from formal text, these are not optimal candidates for learning the translation process. Yet, most existing work that can obtain a considerable amount of parallel data [Resnik and Smith, 2003] does so by identifying parallel documents, which is not applicable for microblogs, as tweets tend to be relatively small and the concept of parallel web pages does not translate into microblogs. Instead, users tend to post self-translated messages, where users post their message in multiple languages in the same post. We show that existing automatic parallel data methods can be augmented to work in this new environment in Chapter 3, and that arbitrary large amounts of in-domain parallel data can be obtained in these domains. While our work in this area benefits existing algorithms for parallel data extraction, we believe that our main contribution is to show that much effort is put into translating microblog posts, which arise from the natural need of users to translate their messages for a broader audience. We show that not only substantial improvements

in machine translation can be obtained using these methods in Chapter 3, but also that other tasks, such as normalization, can benefit from such data in Chapter 4.2. Possible directions for future work include the improvement of the techniques used for parallel data retrieval in a self translated message setup and the development of more applications of the parallel data that have been extracted.

Secondly, we present a model that can better exploit the training data to generalize to unseen samples. As most machine translation systems are not sensitive to lexical traits in words. This leads to sparsity problems in lexically rich domains, such as microblogs. In this regard, we propose character-based word representations that allow words to be composed solely from characters. In Chapter 5, we show that these representations can be beneficial to many NLP applications, as they allow lexical aspects in words, such as morphology, to be learnt automatically. Furthermore, only characters are modelled, which allows models to be extremely compact and scale better to the large amounts of data that can be found in microblogs. Then, in Chapter 6, we describe an alternative to the word softmax using characters and proceed to describe a character-based machine translation system, by reading and generating words at the character level. This allows us to build models that better generalize to the lexically variable language used in microblogs, with a much smaller number of parameters. Our main contribution is to show that it is possible to find a much more compact representation for words both at reading and generation time. It should be clear that while Twitter can contain millions of different word types in English, most of the words are simply reformulations of existing words (*go vs gooooo*), and as such, given that we know the meaning of *go* learning *gooooo* should not be expensive as we only need to learn that *gooooo* is a lexical variant of *go*. However, as existing models operate on the word level, these models are forced to memorize each of the different variants as separate units. This is not only memory inefficient due to the large number of parameters originated from modeling words types separately, but it also limits the generalization ability of the model, as it is simply memorizing words, and not learning the process these words are generated. For instance, if the word *gooo* has not been seen, our model would not be translate it, even if other variations of *go* have been seen.

The main contributions of this these are as follows:

- **Parallel Data Extraction from Self-Contained Messages** - We present a method for extracting parallel data within the same document. Applied to microblogs, this method allows us to extract a large amount of translations that naturally occur in this media.

- **Character-based Word Representations** - We propose an alternative to word lookup tables, the C2W model, which considers the orthographic information within words, allowing it to excel in morphologically rich languages. It also builds a more compact representation of words making the neural networks built using our representations more compact. Furthermore, as the model is composing word from characters, the lexically oriented aspects in words, such as morphology, can be learnt automatically.

- **Character-based Word Generation** - We also generate words at the character level rather than at the word level. This allows the model to generate unseen words in the training set. Combined with the C2W model, we show that the MT problem can be modelled as an open vocabulary problem.

The work in this thesis only provides a first look into many problems underlying NLP and MT. In many cases, the ideas we propose, while novel, make very simplistic assumptions about the problem we address. Some examples include, the simple alignment model (IBM Model 1) and language detection models used for segmenting tweets into parallel segments in Chapter 3, and the normalization method using off-the-shelf MT systems in Chapter 4.2. Even the character-based models used to represent and generate words simply define an encoder and decoder using LSTMs, which suggest that better models can be introduced in the future. We now list some possible future directions that could be considered for the work defined in this thesis.

- **Improved models for parallel data detection in monolingual documents** - The alignment model presented in chapter 3 can be extended to be used to extract parallel data from any documents. For instance, we can use on wikipedia documents to find examples of parenthetical translations [Lin et al., 2008]. However, our model uses IBM model 1 to compute the alignment scores. The advantage of this model is that the decisions made to align each target word to each source word is local each target word, making it easy to use dynamic programming to reuse Viterbi alignments computed from different segmentations of each document. However, IBM Model 1 also ignores distorting, which is problematic when reordering very long documents, each target word is likely to find some source word to align to, especially for function words. Distortion models, such as HMM models [Vogel et al., 1996] and IBM Model 2 and 4, address this problem, as they discourage alignments that are coherent. For instance, it is unlikely that the first and last words in the target sentences are aligned to the first and seconds in the source

language. This intuition is captured by distortion models, which in general prefer alignments between words that are positioned closely in both languages. A key challenge in these approaches would adapting these algorithms to scale with the number of segmentations for each documents. HMM-based models are the best candidates, as the decision making at each target word is still local (in this case to the respective target word and the previous one). Thus, Viterbi alignments can be reused for faster computation.

- **Further analysis of the translation processes in microblogs** - While we show in chapter 3, that improvements can be obtained by using the crawled parallel data, and some examples of lexical gaps that can be filled using these datasets. A more detailed analysis of the data that can be obtained would proof useful to evaluate the possible contribution of the dataset and extract method. For instance, as newer terms arise (e.g. celebraties names or alias), can we obtain translations of these entities in microblogs prior to finding these translation in other domains, such as webpages, Wikipedia or knowledge bases. Answering such open questions would allow a better overview of the value of microblogs in terms of providing parallel data to machine translation systems.

- **Combining word lookups with character-based models** - The main advantage of word lookup tables compared to the C2W model presented in chapter 5 is the speed of the lookup as it simply requires a table lookup. Thus, training and testing the C2W model is significantly slower than the same neural network using a word lookup table. Thus, it is worth extending the model in order to obtain the advantages of the C2W model in terms of lexical awareness and compactness, while perserving the computational speed of the word lookup tables. A simple approach would use a word lookup table for the most frequent word units, and only use the C2W model for less frequent words. As most of the word tokens in natural language restrict to a very small inventory words, keeping this in a word lookup table would dramatically increase the performance of the system. Less frequent words, such as named entities, that are much more numerous would be stored in character-level basis and composed, which yields a much more compact model. Furthermore, as these tend to occur only once or twice, the character-based composition approach would allow these terms to generalize better, while this is not a problem for the more frequent word types. Furthermore, we can extend this model to use subword units rather than the character sequence. This allows different segmentations of words to be used for faster computation and better

generalization. For instance, morphological segmentors can be used to split words into morphemes rather than into characters for composition.

- **Pre-training the C2W model** - It has been shown that pre-training word lookup tables in raw text using objective functions to maximize context prediction [Mikolov et al., 2013] can drastically improve the quality of the word representations. The same can be applied to the C2W model, but it still remains an open question whether the model is expressive enough to capture all the semantic and syntactic information within large vocabularies.

- **Interpretability of the C2W model** We do not provide a detailed interpretation of mechanisms used in the C2W model to learn representations of words (e.g. is it learning subword units as prefixes or simply memorizing certain parts of words). This is because, interpreting a neural network model composed from BLSTMs is not trivial in itself. Thus, finding evidence on the approach used by the model to capture syntactic and semantic remains a future direction of work, as it would allow us to interpret whether the model is capturing the process words are formed. This information would permit us come up with better ways to learn word representations. Finally, if a process can be devised to induce morphemes from the model, these could be used to automatically segment morphologically rich languages, similar to the work in [Poon et al., 2009].

- **Comparison of different words representation models from characters** - Other approaches have been proposed to generate word representations from characters using bag-of-word [Chen et al., 2015] and convolutional approaches [Kim et al., 2015].

- **Subword V2C model** - Similarly to the C2W model, words can be generated at the subword level rather than at the character level. Thus, morphological segmenters can be used in the training set in order to adapt the V2C model to generate words from morphemes. In low resource languages, such an approach would be useful to add a bias towards valid segmentations of words in that language.

Bibliography

S. Afonso, E. Bick, R. Haber, and D. Santos. "Floresta sintá(c)tica": a treebank for Portuguese. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, 2002. 84

M. Ali, B. Shaik, D. Rybach, S. Hahn, R. Schlüter, and H. Ney. Hierarchical hybrid language models for open vocabulary continuous speech recognition using wfst, 2012. 94

R. Astudillo, S. Amir, W. Ling, B. Martins, M. Silva, and I. Trancoso. Inesc-id: A regression model for large scale twitter sentiment lexicon induction. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, 2015a. 7

R. Astudillo, S. Amir, W. Ling, B. Martins, M. Silva, and I. Trancoso. Inesc-id: Sentiment analysis without hand-coded features or liguistic resources using embedding subspaces. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, 2015b. 7

N. B. Atalay, K. Oflazer, and B. Say. The annotation process in the turkish treebank. In *In Proceedings of the 4th International Workshop on Linguistically Interpreted Corpora (LINC)*, Budapest, Hungary, 2003. Association for Computational Linguistics. 84

A. Aw, M. Zhang, J. Xiao, and J. Su. A phrase-based statistical model for SMS text normalization. In *Proceedings of the ACL*, COLING-ACL '06, pages 33–40, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1273073.1273078`. 14

A. Axelrod, R. B. Mayne, C. Callison-burch, M. Osborne, and D. Talbot. Edinburgh system description for the 2005 iwslt speech translation evaluation. In *In Proc.*

International Workshop on Spoken Language Translation (IWSLT, 2005. 48, 72, 108

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2015. URL `http://arxiv.org/abs/1409.0473`. 11, 91, 102, 107

M. Ballesteros, C. Dyer, and N. A. Smith. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proc. EMNLP*, 2015. 6, 111

C. Bannard and C. Callison-Burch. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P05-1074`. 3, 59

M. Bansal, C. Quirk, and R. C. Moore. Gappy phrasal alignment by agreement. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1308–1317, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL `http://dl.acm.org/citation.cfm?id=2002472.2002635`. 62

R. Barzilay and L. Lee. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 16–23, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073448. URL `http://dx.doi.org/10.3115/1073445.1073448`. 3

L. Beinborn, T. Zesch, and I. Gurevych. Cognate production using character-based machine translation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 883–891, 2013. 100

Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, Mar. 2003. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944919.944966`. 93

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=944919.944937`. 46

J. A. Botha and P. Blunsom. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Beijing, China, jun 2014. 78

S. Brants, S. Dipper, S. Hansen, W. Lezius, and G. Smith. The tiger treebank, 2002. 84

F. Braune and A. Fraser. Improved unsupervised sentence alignment for symmetrical and asymmetrical parallel corpora. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 81–89, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1944566.1944576. 13

P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992. 69

P. F. Brown, V. J. D. Pietra, S. A. D. Pietra, and R. L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Comput. Linguist.*, 19: 263–311, June 1993. ISSN 0891-2017. URL http://portal.acm.org/citation.cfm?id=972470.972474. 12, 20, 64

V. Chahuneau, E. Schlinger, N. A. Smith, and C. Dyer. Translating into morphologically rich languages with synthetic phrases. In *Proc. of EMNLP*, 2013. 100

D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014. 16, 76

X. Chen, L. Xu, Z. Liu, M. Sun, and H. Luan. Joint learning of character and word embeddings. 2015. 117

D. Chiang. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 263–270, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219873. URL http://dx.doi.org/10.3115/1219840.1219873. 11

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on*

Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/D14-1179. 80

P. Clarkson and T. Robinson. Towards improved language model evaluation measures. In *EUROSPEECH*. ISCA, 1999. 95

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011. 3, 16, 76, 104

F. de Saussure. *Course in General Linguistics*. 1916. 78

M. Denkowski and A. Lavie. Meteor 1.3: Automatic metric for reliable optimization and evaluation of machine translation systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland, July 2011. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W11-2107. 62

Q. Dou and K. Knight. Large scale decipherment for out-of-domain machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 266–275, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390948.2390982. 11

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July 2011. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1953048.2021068. 97, 108

C. Dyer, S. Muresan, and P. Resnik. Generalizing word lattice translation. In *Proceedings of HLT-ACL*, 2008. 69

C. Dyer, V. Chahuneau, and N. A. Smith. A simple, fast, and effective reparameterization of ibm model 2. In *Proceedings of NAACL-HLT*, pages 644–648, 2013. 38, 64

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. Transition-based dependeny parsing with stack long short-term memory. In *Proceedings of the* 53^{rd} *Annual Meeting of the Association of Computational Linguistics and the* 7^{th} *International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*. ACL, 2015. 76

C. J. Dyer. The "noisier channel": Translation from morphologically complex languages. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 207–211, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W07/W07-0729. 100

G. Foster and R. Kuhn. Mixture-model adaptation for smt. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 128–135, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1626355.1626372. 1

K. Fukushima, K. Taura, and T. Chikayama. A fast and accurate method for detecting English-Japanese parallel texts. In *Proceedings of the Workshop on Multilingual Language Resources and Interoperability*, pages 60–67, Sydney, Australia, July 2006. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W06/W06-1008. 2, 11

W. A. Gale and K. W. Church. A program for aligning sentences in bilingual corpora. In *Proceedings of the 29th Annual Meeting on Association for Computational Linguistics*, ACL '91, pages 177–184, Stroudsburg, PA, USA, 1991. Association for Computational Linguistics. doi: 10.3115/981344.981367. URL http://dx.doi.org/10.3115/981344.981367. 36

J. Ganitkevitch, B. VanDurme, and C. Callison-Burch. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2013)*, Atlanta, Georgia, June 2013. Association for Computational Linguistics. URL http://cs.jhu.edu/~ccb/publications/ppdb.pdf. 3

K. Gimpel, N. Schneider, B. O'Connor, D. Das, D. Mills, J. Eisenstein, M. Heilman, D. Yogatama, J. Flanigan, and N. A. Smith. Part-of-speech tagging for twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 42–47, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL http://dl.acm.org/citation.cfm?id=2002736.2002747. iii, v, 108

S. Gouws, D. Hovy, and D. Metzler. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First Workshop on Unsupervised Learning in NLP*, EMNLP '11, pages 82–90, Stroudsburg, PA, USA, 2011. Association for

Computational Linguistics. ISBN 978-1-937284-13-8. URL http://dl.acm.org/citation.cfm?id=2140458.2140468. 14

A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6), 2005. 78

M. Gutmann and A. Hyvarinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. 2010. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.7404. 92, 107

B. Haddow. Applying pairwise ranked optimisation to improve the interpolation of translation models. In *Proceedings of NAACL*, 2013. URL mixture.pdf. 1

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1): 10–18, Nov. 2009. ISSN 1931-0145. doi: 10.1145/1656274.1656278. URL http://doi.acm.org/10.1145/1656274.1656278. 44

B. Han and T. Baldwin. Lexical normalisation of short text messages: makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 368–378, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-87-9. URL http://dl.acm.org/citation.cfm?id=2002472.2002520. iii, v, 1, 3, 14

B. Han, P. Cook, and T. Baldwin. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 421–432, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390948.2391000. 3, 14, 15, 69

B. Han, P. Cook, and T. Baldwin. Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):5, 2013. 3, 14

C. Hawn. Take two aspirin and tweet me in the morning: how twitter, facebook, and other social media are reshaping health care. *Health affairs*, 28(2):361–368, 2009. iii, v, 1

L. Jehl, F. Hiebel, and S. Riezler. Twitter translation using translation-based cross-lingual retrieval. In *Proceedings of the Seventh Workshop on Statistical Machine*

Translation, pages 410–421, Montréal, Canada, June 2012. Association for Computational Linguistics. 13

J. H. Johnson and J. Martin. Improving translation quality by discarding most of the phrasetable. In *In Proceedings of EMNLP-CoNLL'07*, pages 967–975, 2007. 38

N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *EMNLP*, pages 1700–1709, 2013. 16, 76, 91

B.-J. Kang and K.-S. Choi. Automatic transliteration and back-transliteration by decision tree learning. In *LREC*. Citeseer, 2000. 100

M. Kaufmann. Syntactic Normalization of Twitter Messages. *studies*, 2, 2010. 3, 14

Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015. URL `http://arxiv.org/abs/1508.06615`. 117

P. Koehn. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand, 2005. AAMT, AAMT. URL `http://mt-archive.info/MTS-2005-Koehn.pdf`. 96, 108

P. Koehn and J. Schroeder. Experiments in domain adaptation for statistical machine translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 224–227, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1626355.1626388`. 1

P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1073445.1073462. URL `http://dx.doi.org/10.3115/1073445.1073462`. 48, 71, 108

P. Koehn, A. Axelrod, A. B. Mayne, C. Callison-Burch, M. Osborne, D. Talbot, and M. White. Edinburgh system description for the 2005 nist mt evaluation. In *Proceedings of Machine Translation Evaluation Workshop 2005*, 2005. 68

P. Koehn, H. Hoang, A. Birch, C. Callison-burch, R. Zens, R. Aachen, A. Constantin, M. Federico, N. Bertoldi, C. Dyer, B. Cowan, W. Shen, C. Moran, and O. Bojar. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 11, 15, 67, 75

M. Kozielski, D. Rybach, S. Hahn, R. Schl
"uter, and H. Ney. Open vocabulary handwriting recognition using combined word-level and character-level language models. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 8257–8261, Vancouver, Canada, May 2013. IBM Research Spoken Language Processing Student Travel Grant Award. 93

M. Kozielski, M. Matysiak, P. Doetsch, R. Schlueter, and H. Ney. Open-lexicon language modeling combining word and character levels. *International Conference on Frontiers in Handwriting Recognition*, pages 343–348, Sept. 2014. 93

H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *Proceedings of the 19th international conference on World wide web*, pages 591–600. ACM, 2010. iii, v, 1

M. Larson. Sub-word-based language models for speech recognition: Implications for spoken document retrieval. 93

S. Laviosa. Core patterns of lexical use in a comparable corpus of English lexical prose. *Meta*, 43(4):557–570, 1998. 59, 60

B. Li and J. Liu. Mining Chinese-English parallel corpora from the web. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP)*, 2008. 2, 11

J. Li, D. Jurafsky, and E. H. Hovy. When are tree structures necessary for deep learning of representations? *CoRR*, abs/1503.00185, 2015. URL `http://arxiv.org/abs/1503.00185`. 80

D. Lin, S. Zhao, B. Van Durme, and M. Paşca. Mining parenthetical translations from the web by word alignment. In *Proceedings of ACL-08: HLT*, pages 994–1002, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P/P08/P08-1113`. 13, 115

W. Ling, T. Luís, J. Graça, L. Coheur, and I. Trancoso. Towards a general and extensible phrase-extraction algorithm. In *IWSLT '10: International Workshop on Spoken Language Translation*, pages 313–320, Paris, France, 2010. 64, 76

W. Ling, P. Calado, B. Martins, I. Trancoso, and A. Black. Named entity translation using anchor texts. In *International Workshop on Spoken Language Translation (IWSLT)*, San Francisco, USA, December 2011a. 7

W. Ling, J. a. Graça, D. M. d. Matos, I. Trancoso, and A. Black. Discriminative phrase-based lexicalized reordering models using weighted reordering graphs. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, Chiang Mai, Thailand, November 2011b. Association for Computational Linguistics. 7

W. Ling, T. Luís, J. a. Graça, L. Coheur, and I. Trancoso. Reordering modeling using weighted alignment matrices. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers - Volume 2*, HLT '11, pages 450–454, Stroudsburg, PA, USA, 2011c. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL `http://dl.acm.org/citation.cfm?id=2002736.2002827`. 7

W. Ling, I. Trancoso, and R. Prada. An agent based competitive translation game for second language learning. August 2011d. 7

W. Ling, J. a. Graça, I. Trancoso, and A. Black. Entropy-based pruning for phrase-based machine translation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 962–971, Jeju Island, Korea, July 2012a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D/D12/D12-1088`. 7, 76

W. Ling, N. Tomeh, G. Xiang, A. Black, and I. Trancoso. Improving relative-entropy pruning using significance. In *Proceedings of the 25rd International Conference on Computational Linguistics (Coling 2012)*, Mumbai, India, December 2012b. Coling 2012 Organizing Committee. 7

W. Ling, C. Dyer, A. W. Black, and I. Trancoso. Paraphrasing 4 microblog normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 73–84, Seattle, Washington, USA, October 2013a. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D13-1008`. 5, 59, 76

W. Ling, G. Xiang, C. Dyer, A. Black, and I. Trancoso. Microblogs as parallel corpora. In *Proceedings of the 51st Annual Meeting on Association for Computational Linguistics*, ACL '13. Association for Computational Linguistics, 2013b. 4, 19, 26, 35, 39

W. Ling, L. Marujo, C. Dyer, A. Black, and I. Trancoso. Crowdsourcing high-quality parallel data extraction from twitter. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, WMT '14, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics. 31, 36, 40

W. Ling, L. Chu-Cheng, Y. Tsvetkov, S. Amir, R. F. Astudillo, C. Dyer, A. W. Black, and I. Trancoso. Not all contexts are created equal: Better word representations with variable attention. EMNLP, 2015a. 7

W. Ling, C. Dyer, A. Black, and I. Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2015b. 7, 17, 87

W. Ling, T. Luís, L. Marujo, R. F. Astudillo, S. Amir, C. Dyer, A. W. Black, and I. Trancoso. Finding function in form: Compositional character models for open vocabulary word representation. EMNLP, 2015c. 5, 7, 76, 94

S. Liu, N. Yang, M. Li, and M. Zhou. A recursive recurrent neural network for statistical machine translation. In *Proceedings of ACL*, pages 1491–1500, 2014. 16, 76

M. Lui, J. H. Lau, and T. Baldwin. Automatic detection and language identification of multilingual documents. 2014. 33

T. Luong, R. Socher, and C. Manning. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, chapter Better Word Representations with Recursive Neural Networks for Morphology. Association for Computational Linguistics, 2013. URL `http://aclweb.org/anthology/W13-3512`. 78

N. Madnani. The circle of meaning: from translation to paraphrasing and back. 2010. 3

N. Madnani and B. J. Dorr. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387, 2010. 3

C. D. Manning. Part-of-speech tagging from 97linguistics? In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing'11, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-19399-6. URL http://dl.acm.org/citation.cfm?id=1964799.1964816. 86, 88

M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2), June 1993. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=972470.972475. 84

W. Marslen-Wilson and L. K. Tyler. Rules, representations, and the English past tense. *Trends in Cognitive Science*, 2(11):428–435, 1998. 78

M. A. Martí, M. Taulé, L. Márquez, and M. Bertran. CESS-ECE: A multilingual and multilevel annotated corpus. In *LREC*, 2007. 84

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048, 2010. 80, 91

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013. 3, 7, 17, 77, 87, 117

A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. 2008. URL http://books.nips.cc/papers/files/nips21/NIPS2008_0918.pdf. 93

A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1081–1088. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3583-a-scalable-hierarchical-distributed-language-model.pdf. 91

F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *AISTATS'05*, pages 246–252, 2005. 92

T. Mueller, H. Schmid, and H. Schütze. Efficient higher-order crfs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural*

Language Processing. Association for Computational Linguistics, 2013. URL http://aclweb.org/anthology/D13-1032. 83

D. Munteanu and D. Marcu. Improving machine translation performance by exploiting comparable corpora. *Computational Linguistics*, 31(4):477–504, 2005. 35, 45

D. S. Munteanu, A. Fraser, and D. Marcu. Improved machine translation performance via parallel sentence extraction from comparable corpora. In *HLT-NAACL*, pages 265–272, 2004. 13

T. Nakagawa, T. Kudoh, and Y. Matsumoto. Unknown word guessing and part-of-speech tagging using support vector machines. In *In Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, 2001. 83

M. Nuhn, A. Mauser, and H. Ney. Deciphering foreign language by combining language models and context vectors. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 156–164, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=2390524.2390547. 11

F. J. Och. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 160–167, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1075096.1075117. URL http://dx.doi.org/10.3115/1075096.1075117. 48, 68, 72, 108

F. J. Och and H. Ney. The alignment template approach to statistical machine translation. *Comput. Linguist.*, 30(4):417–449, Dec. 2004. ISSN 0891-2017. doi: 10.1162/0891201042544884. URL http://dx.doi.org/10.1162/0891201042544884. 64

O. Owoputi, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *In Proceedings of NAACL*, 2013. 108

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL http://dx.doi.org/10.3115/1073083.1073135. 48, 72, 108

D. Pennell and Y. Liu. A character-level machine translation approach for normalization of sms abbreviations. In *IJCNLP*, pages 974–982, 2011. 15

D. L. Pennell and Y. Liu. Normalization of informal text. *Computer Speech & Language*, 28(1):256–277, 2014. 15

H. Poon, C. Cherry, and K. Toutanova. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, NAACL '09, pages 209–217, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-41-1. URL http://dl.acm.org/citation.cfm?id=1620754.1620785. 117

S. Ravi and K. Knight. Deciphering foreign language. In *ACL'11*, pages 12–21, 2011. 11

P. Resnik and N. A. Smith. The web as a parallel corpus. *Computational Linguistics*, 29:349–380, 2003. 2, 11, 12, 35, 45, 62, 113

T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010. iii, v, 1

B. Sankaran, M. Razmara, A. Farzindar, W. Khreich, F. Popowich, and A. Sarkar. Domain adaptation techniques for machine translation and their evaluation in a real-world setting. In *Proceedings of 25th Canadian Conference on Artificial Intelligence*, Toronto, Canada, may 2012. 1

C. D. Santos and B. Zadrozny. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826. JMLR Workshop and Conference Proceedings, 2014. URL http://jmlr.org/proceedings/papers/v32/santos14.pdf. 84, 85

R. Sennrich, B. Haddow, and A. Birch. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL http://arxiv.org/abs/1508.07909. 110

J. R. Smith, C. Quirk, and K. Toutanova. Extracting parallel sentences from comparable corpora using document level alignment. In *Proc. NAACL*, 2010. 13

A. Søgaard. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics. ISBN 978-1-932432-88-6. URL http://dl.acm.org/citation.cfm?id=2002736.2002748. 88

R. Soricut and F. Och. Unsupervised morphology induction using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1627–1637, Denver, Colorado, May–June 2015. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N15-1186. 78

D. Spoustová, J. Hajič, J. Raab, and M. Spousta. Semi-supervised training for the averaged perceptron pos tagger. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, 2009. 88

X. Sun. Structure regularization for structured prediction: Theories and experiments. *CoRR*, abs/1411.6243, 2014. URL http://arxiv.org/abs/1411.6243. 86, 88

M. Sundermeyer, R. Schl
"uter, and H. Ney. Lstm neural networks for language modeling. In *Interspeech*, pages 194–197, Portland, OR, USA, Sept. 2012a. 80

M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *INTERSPEECH*, 2012b. 95, 96

H. Tang, J. Keshet, and K. Livescu. Discriminative pronunciation modeling: A large-margin, feature-rich approach. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 194–203. Association for Computational Linguistics, 2012. 15

Y. Tsvetkov, M. Faruqui, W. Ling, G. Lample, and C. Dyer. Evaluation of word vector representations by subspace alignment. EMNLP, 2015. 7

F. Ture and J. Lin. Why not grab a free lunch? mining large corpora for parallel sentences to improve translation modeling. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 626–630, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/N12-1079. 2, 11

J. Uszkoreit, J. Ponte, A. C. Popat, and M. Dubiner. Large scale parallel document mining for machine translation. In *COLING*, pages 1101–1109, 2010a. 2

J. Uszkoreit, J. M. Ponte, A. C. Popat, and M. Dubiner. Large scale parallel document mining for machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1101–1109. Association for Computational Linguistics, 2010b. 11, 12, 34

D. Vilar, J.-T. Peter, and H. Ney. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 33–39, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1626355.1626360`. 101

S. Vogel, H. Ney, and C. Tillmann. Hmm-based word alignment in statistical translation. In *Proceedings of the 16th conference on Computational linguistics - Volume 2*, COLING '96, pages 836–841, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. doi: 10.3115/993268.993313. URL `http://dx.doi.org/10.3115/993268.993313`. 20, 115

V. Volansky, N. Ordan, and S. Wintner. On the features of translationese. *Literary and Linguistic Computing*, 2013. 59, 60

P. Wang and H. Ng. A beam-search decoder for normalization of social media text with application to machine translation. In *Proceedings of NAACL-HLT 2013*, NAACL '13. Association for Computational Linguistics, 2013. 15

J. Xu, R. Weischedel, and C. Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 105–110, New York, NY, USA, 2001. ACM. ISBN 1-58113-331-6. doi: 10.1145/383952.383968. URL `http://doi.acm.org/10.1145/383952.383968`. 13

J. Xu, R. Zens, and H. Ney. Sentence segmentation using ibm word alignment model 1. In *In Proceedings of EAMT 2005 (10th Annual Conference of the European Association for Machine Translation*, pages 280–287, 2005. 13

W. Xu, A. Ritter, and R. Grishman. Gathering and generating paraphrases from twitter with application to normalization. In *Proceedings of the Sixth Workshop on Building and Using Comparable Corpora*, pages 121–128, Sofia, Bulgaria, August

2013. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W13-2515. 16

Y. Yang and J. Eisenstein. A log-linear model for unsupervised text normalization. In *Proc. of EMNLP*, 2013. 14

S. Zhao, H. Wang, X. Lan, and T. Liu. Leveraging multiple mt engines for paraphrase generation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1326–1334. Association for Computational Linguistics, 2010. 3