Context Sensitive Lemmatization Using Two Successive Bidirectional Gated Recurrent Networks

Abhisek Chakrabarty **Onkar Arun Pandit** **Utpal Garain**
Computer Vision and Pattern Recognition Unit
Indian Statistical Institute
203 B.T. Road, Kolkata-700108, India
`abhisek0842@gmail.com, oapandit@gmail.com, utpal@isical.ac.in`

Abstract

We introduce a composite deep neural network architecture for supervised and language independent context sensitive lemmatization. The proposed method considers the task as to identify the correct edit tree representing the transformation between a word-lemma pair. To find the lemma of a surface word, we exploit two successive bidirectional gated recurrent structures - the first one is used to extract the character level dependencies and the next one captures the contextual information of the given word. The key advantages of our model compared to the state-of-the-art lemmatizers such as Lemming and Morfette are - (i) it is independent of human decided features (ii) except the gold lemma, no other expensive morphological attribute is required for joint learning. We evaluate the lemmatizer on nine languages - Bengali, Catalan, Dutch, Hindi, Hungarian, Italian, Latin, Romanian and Spanish. It is found that except Bengali, the proposed method outperforms Lemming and Morfette on the other languages. To train the model on Bengali, we develop a gold lemma annotated dataset[1] (having $1,702$ sentences with a total of $20,257$ word tokens), which is an additional contribution of this work.

1 Introduction

Lemmatization is the process to determine the root/dictionary form of a surface word. Morphologically rich languages suffer due to the existence of various inflectional and derivational variations of a root depending on several linguistic properties such as honorificity, parts of speech (POS), person, tense etc. Lemmas map the related word forms to lexical resources thus identifying them as the members of the same group and providing their semantic and syntactic information. Stemming is a way similar to lemmatization producing the common portion of variants but it has several limitations - (i) there is no guarantee of a stem to be a legitimate word form (ii) words are considered in isolation. Hence, for context sensitive languages i.e. where same inflected word form may come from different sources and can only be disambiguated by considering its neighbouring information, there lemmatization defines the foremost task to handle diverse text processing problems (e.g. sense disambiguation, parsing, translation).

The key contributions of this work are as follows. We address context sensitive lemmatization introducing a two-stage bidirectional gated recurrent neural network (BGRNN) architecture. Our model is a supervised one that needs lemma tagged continuous text to learn. Its two most important advantages compared to the state-of-the-art supervised models (Chrupala et al., 2008; Toutanova and Cherry, 2009; Gesmundo and Samardzic, 2012; Müller et al., 2015) are - (i) we do not need to define hand-crafted features such as the word form, presence of special characters, character alignments, surrounding words etc. (ii) parts of speech and other morphological attributes of the surface words are not required for joint learning. Additionally, unknown word forms are also taken care of as the transformation between word-lemma pair is learnt, not the lemma itself. We exploit two steps learning in our method. At first, characters in the words are passed sequentially through a BGRNN to get a syntactic embedding of each word and then the outputs are

[1]The dataset and the code of model architecture are released with the paper. They are also available in `http://www.isical.ac.in/~utpal/resources.php`

combined with the corresponding semantic embeddings. Finally, mapping between the combined embeddings to word-lemma transformations are learnt using another BGRNN.

For the present work, we assess our model on nine languages having diverse morphological variations. Out of them, two (Bengali and Hindi) belong to the Indic languages family and the rests (Catalan, Dutch, Hungarian, Italian, Latin, Romanian and Spanish) are taken from the European languages. To evaluate the proposed model on Bengali, a lemma annotated continuous text has been developed. As so far there is no such standard large dataset for supervised lemmatization in Bengali, the prepared one would surely contribute to the respective NLP research community. For the remaining languages, standard datasets are used for experimentation. Experimental results reveal that our method outperforms Lemming (Müller et al., 2015) and Morfette (Chrupala et al., 2008) on all the languages except Bengali.

1.1 Related Works

Efforts on developing lemmatizers can be divided into two principle categories (i) rule/heuristics based approaches (Koskenniemi, 1984; Plisson et al., 2004) which are usually not portable to different languages and (ii) learning based methods (Chrupala et al., 2008; Toutanova and Cherry, 2009; Gesmundo and Samardzic, 2012; Müller et al., 2015; Nicolai and Kondrak, 2016) requiring prior training dataset to learn the morphological patterns. Again, the later methods can be further classified depending on whether context of the current word is considered or not. Lemmatization without context (Cotterell et al., 2016; Nicolai and Kondrak, 2016) is closer to stemming and not the focus of the present work. It is noteworthy here that the supervised lemmatization methods do not try to classify the lemma of a given word form as it is infeasible due to having a large number of lemmas in a language. Rather, learning the transformation between word-lemma pair is more generalized and it can handle the unknown word forms too. Several representations of word-lemma transformation have been introduced so far such as shortest edit script (SES), label set, edit tree by Chrupala et al. (2008), Gesmundo and Samardzic (2012) and Müller et al. (2015) respectively. Following Müller et al. (2015), we consider lemmatization as the edit tree classification problem. Toutanova and Cherry (2009); Müller et al. (2015) also showed that joint learning of lemmas with other morphological attributes is mutually beneficial but obtaining the gold annotated datasets is very expensive. In contrast, our model needs only lemma annotated continuous text (not POS and other tags) to learn the word morphology.

Since our experiments include the Indic languages also, it would not be an overstatement to say that there have been little efforts on lemmatization so far (Faridee et al., 2009; Loponen and Järvelin, 2010; Paul et al., 2013; Bhattacharyya et al., 2014). The works by Faridee et al. (2009); Paul et al. (2013) are language specific rule based for Bengali and Hindi respectively. (Loponen and Järvelin, 2010)'s primary objective was to improve the retrieval performance. Bhattacharyya et al. (2014) proposed a heuristics based lemmatizer using WordNet but they did not consider context of the target word which is an important basis to lemmatize Indic languages. Chakrabarty and Garain (2016) developed an unsupervised language independent lemmatizer and evaluated it on Bengali. They consider the contextual information but the major disadvantage of their method is dependency on dictionary as well as POS information. Very recently, a supervised neural lemmatization model has been introduced by Chakrabarty et al. (2016). They treat the problem as lemma transduction rather than classification. The particular root in the dictionary is chosen as the lemma with which the transduced vector possesses maximum cosine similarity. Hence, their approach fails when the correct lemma of a word is not present in the dictionary. Besides, the lemmatization accuracy obtained by the respective method is not very significant. Apart from the mentioned works, there is no such commendable effort so far.

Rest of this paper is organized as follows. In section 2, we describe the proposed lemmatization method. Experimental setup and the results are presented in section 3. Finally, in section 4 we conclude the paper.

2 The Proposed Method

As stated earlier in section 1.1, we represent the mapping between a word to its lemma using edit tree (Chrupała, 2008; Müller et al., 2015). An edit tree embeds all the necessary edit operations within it i.e. insertions, deletions and substitutions of strings required throughout the transformation

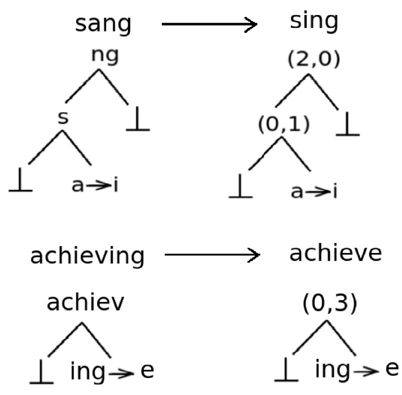

Figure 1: Edit trees for the word-lemma pairs 'sang-sing' and 'achieving-achieve'.

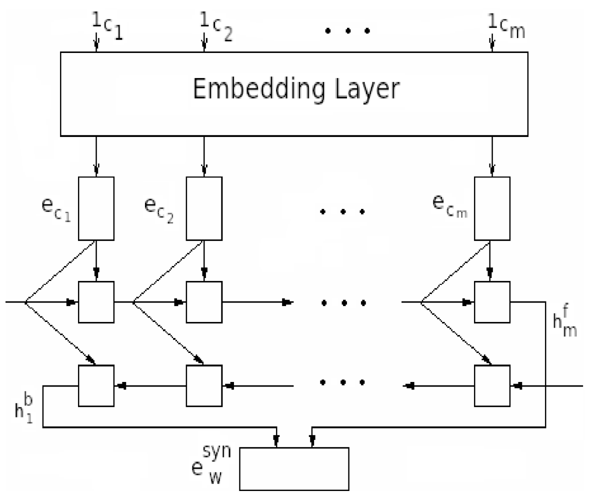

Figure 2: Syntactic vector composition for a word.

process. Figure 1 depicts two edit trees that map the inflected English words *'sang'* and *'achieving'* to their respective lemmas *'sing'* and *'achieve'*. For generalization, edit trees encode only the substitutions and the length of prefixes and suffixes of the longest common substrings. Initially, all unique edit trees are extracted from the associated surface word-lemma pairs present in the training set. The extracted trees refer to the class labels in our model. So, for a test word, the goal is to classify the correct edit tree which, applied on the word, returns the lemma.

Next, we will describe the architecture of the proposed neural lemmatization model. It is evident that for morphologically rich languages, both syntactic and semantic knowledge help in lemmatizing a surface word. Now a days, it is a common practice to embed the functional properties of words into vector representations. Despite the word vectors prove very effectual in semantic processing tasks, they are modelled using the distributional similarity obtained from a raw corpus. Morphological regularities, local and non-local dependencies in character sequences that play deciding roles to find the lemmas, are not taken into account where each word has its own vector interpretation. We address this issue by incorporating two different embeddings into our model. Semantic embedding is achieved using word2vec (Mikolov et al., 2013a,b), which has been empirically found highly successful. To devise the syntactic embedding of a word, we follow the work of Ling et al. (2015) that uses compositional character to word model using bidirectional long-short term memory (BLSTM) network. In our experiments, different gated recurrent cells such as LSTM (Graves, 2013) and GRU (Cho et al., 2014), are explored. The next subsection describes the module to construct the syntactic vectors by feeding the character sequences into BGRNN architecture.

2.1 Forming Syntactic Embeddings

Our goal is to build syntactic embeddings of words that capture the similarities in morphological level. Given an input word w, the target is to obtain a d dimensional vector representing the syntactic structure of w. The procedure is illustrated in Figure 2. At first, an alphabet of characters is defined as C. We represent w as a sequence of characters $c_1, \ldots, c_m$ where m is the word length and each character c_i is defined as a one hot encoded vector $\mathbf{1}_{c_i}$, having one at the index of c_i in the alphabet C. An embedding layer is defined as $\mathbf{E}_c \in \mathbb{R}^{d_c \times |C|}$, that projects each one hot encoded character vector to a d_c dimensional embedded vector. For a character c_i, its projected vector $\mathbf{e}_{c_i}$ is obtained from the embedding layer $\mathbf{E}_c$, using this relation $\mathbf{e}_{c_i} = \mathbf{E}_c \cdot \mathbf{1}_{c_i}$ where '$\cdot$' is the matrix multiplication operation.

Given a sequence of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_m$ as input, a LSTM cell computes the state sequence $\mathbf{h}_1, \ldots, \mathbf{h}_m$ using the following equations:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{V}_f\mathbf{c}_{t-1} + \mathbf{b}_f)\\
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{V}_i\mathbf{c}_{t-1} + \mathbf{b}_i)\\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1}\\
&\quad + \mathbf{i}_t \odot tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{V}_o\mathbf{c}_t + \mathbf{b}_o)\\
\mathbf{h}_t &= \mathbf{o}_t \odot tanh(\mathbf{c}_t),
\end{aligned}$$

Whereas, the updation rules for GRU are as follows

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1} + \mathbf{b}_z) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1} + \mathbf{b}_r) \\
\mathbf{h}_t &= (\mathbf{1} - \mathbf{z}_t) \odot \mathbf{h}_{t-1} \\
&\quad + \mathbf{z}_t \odot tanh(\mathbf{W}_h\mathbf{x}_t + \mathbf{U}_h(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h),
\end{aligned}$$

σ denotes the sigmoid function and $\odot$ stands for the element-wise (Hadamard) product. Unlike the simple recurrent unit, LSTM uses an extra memory cell $\mathbf{c}_t$ that is controlled by three gates - input ($\mathbf{i}_t$), forget ($\mathbf{f}_t$) and output ($\mathbf{o}_t$). $\mathbf{i}_t$ controls the amount of new memory content added to the memory cell, $\mathbf{f}_t$ regulates the degree to which the existing memory is forgotten and $\mathbf{o}_t$ finally adjusts the memory content exposure. **W**, **U**, **V** (weight matrices), **b** (bias) are the parameters.

Without having a memory cell like LSTM, a GRU uses two gates namely update ($\mathbf{z}_t$) and reset ($\mathbf{r}_t$). The gate, $\mathbf{z}_t$ decides the amount of update needed for activation and $\mathbf{r}_t$ is used to ignore the previous hidden states (when close to 0, it forgets the earlier computation). So, for a sequence of projected characters $\mathbf{e}_{c_1}, \ldots, \mathbf{e}_{c_m}$, the forward and the backward networks produce the state sequences $\mathbf{h}_1^f, \ldots, \mathbf{h}_m^f$ and $\mathbf{h}_m^b, \ldots, \mathbf{h}_1^b$ respectively. Finally, we obtain the syntactic embedding of w, denoted as $\mathbf{e}_w^{syn}$, by concatenating the final states of these two sequences.

$$\mathbf{e}_w^{syn} = [\mathbf{h}_1^b, \mathbf{h}_m^f]$$

2.2 Model

We present the sketch of the final integrated model in Figure 3. For a word w, let $\mathbf{e}_w^{sem}$ denotes its semantic embedding obtained using word2vec. Both the vectors, $\mathbf{e}_w^{syn}$ and $\mathbf{e}_w^{sem}$ are concatenated together to shape the composite representation $\mathbf{e}_w^{com}$ which carries the morphological and distributional information within it. Firstly, for all the words present in the training set, their composite vectors are generated. Next, they are fed sentence-wise into the next level of BGRNN to train the model for the edit tree classification task. This second level bidirectional network accounts the local context in both forward and backward directions, which is essential for lemmatization in context sensitive languages. Let, $\mathbf{e}_{w_1}^{com}, \ldots, \mathbf{e}_{w_n}^{com}$ be the input sequence of composite vectors to the BGRNN model, representing a sentence having n words $w_1, \ldots, w_n$. For the i^{th} vector $\mathbf{e}_{w_i}^{com}$, $\mathbf{h}_i^f$ and $\mathbf{h}_i^b$ denote the forward and backward states respectively carrying the informations of $w_1, \ldots, w_i$ and $w_i, \ldots, w_n$.

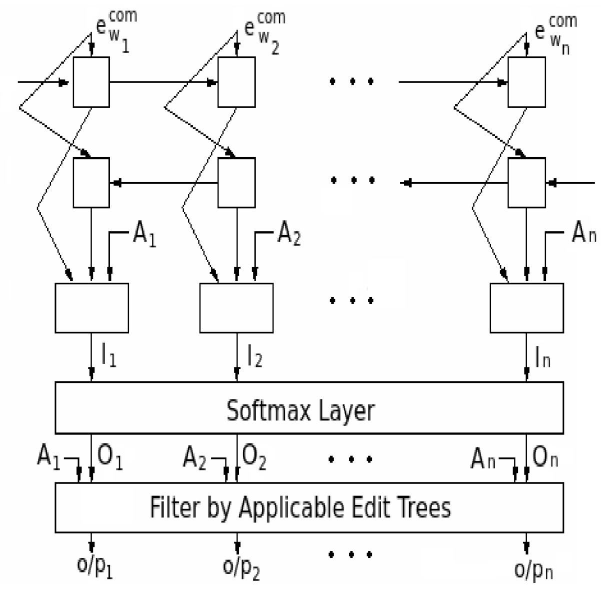

Figure 3: Second level BGRNN model for edit tree classification.

2.2.1 Incorporating Applicable Edit Trees Information

One aspect that we did not look into so far, is that for a word all unique edit trees extracted from the training set are not applicable as this would lead to incompatible substitutions. For example, the edit tree for the word-lemma pair 'sang-sing' depicted in Figure 1, cannot be applied on the word 'achieving'. This information is prior before training the model i.e. for any arbitrary word, we can sort out the subset of unique edit trees from the training samples in advance, which are applicable on it. In general, if all the unique edit trees in the training data are set as the class labels, the model will learn to distribute the probability mass over all the classes which is a clear-cut bottleneck. In order to alleviate this problem, we take a novel strategy so that for individual words in the input sequence, the model will learn, to which classes, the output probability should be apportioned.

Let $T = \{t_1, \ldots, t_k\}$ be the set of distinct edit trees found in the training set. For the word w_i in the input sequence $w_1, \ldots, w_n$, we define its applicable edit trees vector as $\mathbf{A}_i = (a_i^1, \ldots, a_i^k)$ where $\forall j \in \{1, \ldots, k\}$, $a_i^j = 1$ if t_j is applicable for w_i, otherwise 0. Hence, $\mathbf{A}_i$ holds the information regarding the set of edit trees to concentrate upon, while processing the word w_i. We combine $\mathbf{A}_i$ together with $\mathbf{h}_i^f$ and $\mathbf{h}_i^b$ for the final classification task as following,

$$\mathbf{l}_i = softplus(\mathbf{L}^f \mathbf{h}_i^f + \mathbf{L}^b \mathbf{h}_i^b + \mathbf{L}^a \mathbf{A}_i + \mathbf{b}_l),$$

where 'softplus' denotes the activation function $f(x) = ln(1 + e^x)$ and $\mathbf{L}^f, \mathbf{L}^b, \mathbf{L}^a$ and $\mathbf{b}_l$ are the parameters trained by the network. At the end, $\mathbf{l}_i$ is passed through the softmax layer to get the output labels for w_i.

To pick the correct edit tree from the output of the softmax layer, we exploit the prior information $\mathbf{A}_i$. Instead of choosing the class that gets the maximum probability, we select the maximum over the classes corresponding to the applicable edit trees. The idea is expressed as follows. Let $\mathbf{O}_i = (o_i^1, \ldots, o_i^k)$ be the output of the softmax layer. Instead of opting for the maximum over $o_i^1, \ldots, o_i^k$ as the class label, the highest probable class out of those corresponding to the applicable edit trees, is picked up. That is, the particular edit tree $t_j \in T$ is considered as the right candidate for w_i, where

$$j = \operatorname{argmax}_{j' \in \{1,\ldots,k\} \,\wedge\, a_i^{j'} = 1} \; o_i^{j'}$$

In this way, we cancel out the non-applicable classes and focus only on the plausible candidates.

3 Experimentation

Out of the nine reference languages, initially we choose four of them (Bengali, Hindi, Latin and Spanish) for in-depth analysis. We conduct an exhaustive set of experiments - such as determining the direct lemmatization accuracy, accuracy obtained without using applicable edit trees in training, measuring the model's performance on the unseen words etc. on these four languages. Later we consider five more languages (Catalan, Dutch, Hungarian, Italian and Romanian) mostly for testing the generalization ability of the proposed method. For these additional languages, we present only the lemmatization accuracy in section 3.2.

Datasets: As Bengali is a low-resourced language, a relatively large lemma annotated dataset is prepared for the present work using Tagore's short stories collection[2] and randomly selected news articles from miscellaneous domains. One

	# Sentences	# Word Tokens
Bengali	1,702	20,257
Hindi	36,143	819,264
Latin	15,002	165,634
Spanish	15,984	477,810

Table 1: Dataset statistics of the 4 languages.

linguist took around 2 months to complete the annotation which was checked by another person and differences were sorted out. Out of the 91 short stories of Tagore, we calculate the value of (# tokens / # distinct tokens) for each story. Based on this value (lower is better), top 11 stories are selected. The news articles[3] are crafted from the following domains: animal, archaeology, business, country, education, food, health, politics, psychology, science and travelogue. In Hindi, we combine the COLING'12 shared task data for dependency parsing and Hindi WSD health and tourism corpora[4] (Khapra et al., 2010) together[5]. For Latin, the data is taken from the PROIEL treebank (Haug and Jøhndal, 2008) and for Spanish, we merge the training and development datasets of CoNLL'09 (Hajič et al., 2009) shared task on syntactic and semantic dependencies. The dataset statistics are given in Table 1. We assess the lemmatization performance by measuring the direct accuracy which is the ratio of the number of correctly lemmatized words to the total number of input words. The experiments are performed using 4 fold cross validation technique i.e. the datasets are equi-partitioned into 4 parts at sentence level and then each part is tested exactly once using the model trained on the remaining 3 parts. Finally, we report the average accuracy over 4 fold.

Induction of Edit Tree Set: Initially, distinct edit trees are induced from the word-lemma pairs present in the training set. Next, the words in the training data are annotated with their corresponding edit trees. Training is accomplished on this edit tree tagged text. Figure 4 plots the growth of the edit tree set against the number of word-lemma samples in the four languages. With the increase of samples, the size of edit tree set gradually converges revealing the fact that most of the frequent transformation patterns (both regular and irregular) are covered by the induction process. From

[2]www.rabindra-rachanabali.nltr.org

[3]http://www.anandabazar.com/

[4]http://www.cfilt.iitb.ac.in/wsd/annotated_corpus/

[5]We also release the Hindi dataset with this paper as it is a combination of two different datasets.

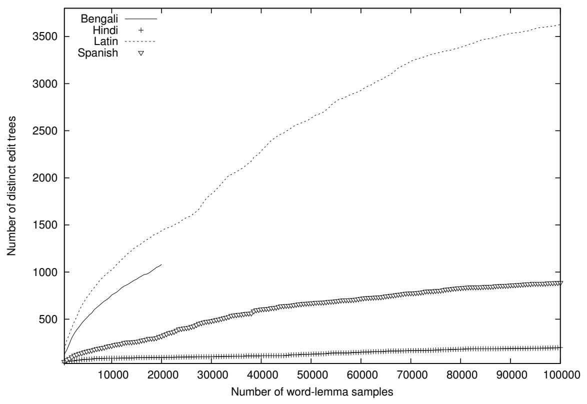

Figure 4: Increase of the edit tree set size with the number of word-lemma samples.

Figure 4, morphological richness can be compared across the languages. When convergence happens quickly i.e. at relatively less number of samples, it evidences that the language is less complex. Among the four reference languages, Latin stands out as the most intricate, followed by Bengali, Spanish and Hindi.

Semantic Embeddings: We obtain the distributional word vectors for Bengali and Hindi by training the word2vec model on FIRE Bengali and Hindi news corpora[6]. Following the work by Mikolov et al. (2013a), continuous-bag-of-words architecture with negative sampling is used to get 200 dimensional word vectors. For Latin and Spanish, we use the embeddings released by Bamman and Smith (2012)[7] and Cardellino (2016)[8] respectively.

Syntactic Representation: We acquire the statistics of word length versus frequency from the datasets and find out that irrespective of the languages, longer words (have more than 20-25 characters) are few in numbers. Based on this finding, each word is limited to a sequence of 25 characters. Smaller words are padded null characters at the end and for the longer words, excess characters are truncated out. So, each word is represented as a 25 length array of one hot encoded vectors which is given input to the embedding layer that works as a look up table producing an equal length array of embedded vectors. Initialization of the embedding layer is done randomly and the embedded vector dimension is set to 10. Eventually, the output of the embedding layer is passed to the first level BGRNN for learning the syntactic representation.

Hyper Parameters: There are several hyper parameters in our model such as the number of neurons in the hidden layer ($\mathbf{h}_t$) of both first and second level BGRNN, learning mode, number of epochs to train the models, optimization algorithm, dropout rate etc. We experiment with different settings of these parameters and report where optimum results are achieved. For both the bidirectional networks, number of hidden layer neurons is set to 64. Online learning is applied for updation of the weights. Number of epochs varies across languages to converge the training. It is maximum for Bengali (around 80 epochs), followed by Latin, Spanish and Hindi taking around 50, 35 and 15 respectively. Throughout the experiments, we set the dropout rate as 0.2 to prevent over-fitting. Different optimization algorithms like AdaDelta (Zeiler, 2012), Adam (Kingma and Ba, 2014), RMSProp (Dauphin et al., 2015) are explored. Out of them, Adam yields the best result. We use the categorical cross-entropy as the loss function in our model.

Baselines: We compare our method with Lemming[9] and Morfette[10]. Both the model jointly learns lemma and other morphological tags in context. Lemming uses a 2nd-order linear-chain CRF to predict the lemmas whereas, the current version of Morfette is based on structured perceptron learning. As POS information is a compulsory requirement of these two models, the Bengali data is manually POS annotated. For the other languages, the tags were already available. Although this comparison is partially biased as the proposed method does not need POS information, but the experimental results show the effectiveness of our model. There is an option in Lemming and Morfette to provide an exhaustive set of root words which is used to exploit the dictionary features i.e. to verify if a candidate lemma is a valid form or not. To make the comparisons consistent, we do not exploit any external dictionary in our experiments.

3.1 Results

The lemmatization results are presented in Table 2. We explore our proposed model with two types of gated recurrent cells - LSTM and GRU. As there

[6]http://fire.irsi.res.in/fire
[7]http://www.cs.cmu.edu/~dbamman/latin.html
[8]http://crscardellino.me/SBWCE/
[9]http://cistern.cis.lmu.de/lemming/
[10]https://github.com/gchrupala/morfette

	Bengali	Hindi	Latin	Spanish
BLSTM-BLSTM	90.84/91.14	94.89/**94.90**	89.35/89.52	97.85/97.91
BGRU-BGRU	90.63/90.84	94.44/94.50	89.40/**89.59**	98.07/**98.11**
Lemming	**91.69**	91.64	88.50	93.12
Morfette	90.69	90.57	87.10	92.90

Table 2: Lemmatization accuracy (in %) without/with restricting output classes.

	Bengali	Hindi	Latin	Spanish
BLSTM-BLSTM	86.46/89.52	94.34/94.52	85.70/87.35	97.39/97.62
BGRU-BGRU	86.39/88.90	93.84/94.04	85.49/86.87	97.51/97.73

Table 3: Lemmatization accuracy (in %) without using applicable edit trees in training.

are two successive bidirectional networks - the first one for building the syntactic embedding and the next one for the edit tree classification, so basically we deal with two different models BLSTM-BLSTM and BGRU-BGRU. Table 2 shows the comparison results of these models with Lemming and Morfette. In all cases, the average accuracy over 4 fold cross validation on the datasets is reported. For an entry 'x/y' in Table 2, x denotes the accuracy without output classes restriction, i.e. taking the maximum over all edit tree classes present in the training set, whereas y refers to the accuracy when output is restricted in only the applicable edit tree classes of the input word. Except for Bengali, the proposed models outperform the baselines for the other three languages. In Hindi, BLSTM-BLSTM gives the best result (94.90%). For Latin and Spanish, the highest accuracy is achieved by BGRU-BGRU (89.59% and 98.11% respectively). In the Bengali dataset, Lemming produces the optimum result (91.69%) beating its closest performer BLSTM-BLSTM by 0.55%. It is to note that the training set size in Bengali is smallest compared to the other languages (on average, $16,712$ tokens in each of the 4 folds). Overall, BLSTM-BLSTM and BGRU-BGRU perform equally good. For Bengali and Hindi, the former model is better and for Latin and Spanish, the later yields more accuracy. Throughout the experiments, restricting the output over applicable classes improves the performance significantly. The maximum improvements we get are: 0.30% in Bengali using BLSTM-BLSTM (from 90.84% to 91.14%), 0.06% in Hindi using BGRU-BGRU (from 94.44% to 94.50%), 0.19% in Latin using BGRU-BGRU (from 89.40% to 89.59%) and 0.06% in Spanish using BLSTM-BLSTM (from 97.85% to 97.91%). To compare between the two baselines, Lemming consistently performs better

Bengali	Hindi	Latin	Spanish
27.17	5.25	15.74	7.54

Table 4: Proportion of unknown word forms (in %) present in the test sets.

than Morfette (the maximum difference between their accuracies is 1.40% in Latin).

Effect of Training without Applicable Edit Trees: We also explore the impact of applicable edit trees in training. To see the effect, we train our model without giving the applicable edit trees information as input. In the model design, the equation for the final classification task is changed as follows,

$$\mathbf{l}_i = softplus(\mathbf{L}^f \mathbf{h}_i^f + \mathbf{L}^b \mathbf{h}_i^b + \mathbf{b}_l),$$

The results are presented in Table 3. Except for Spanish, BLSTM-BLSTM outperforms BGRU-BGRU in all the other languages. As compared with the results in Table 2, for every model, training without applicable edit trees degrades the lemmatization performance. In all cases, BGRU-BGRU model gets more affected than BLSTM-BLSTM. Language-wise, the drops in its accuracy are: 1.94% in Bengali (from 90.84% to 88.90%), 0.46% in Hindi (from 94.50% to 94.04%), 2.72% in Latin (from 89.59% to 86.87%) and 0.38% in Spanish (from 98.11% to 97.73%).

One important finding to note in Table 3 is that irrespective of any particular language and model used, the amount of increase in accuracy due to the output restriction on the applicable classes is much more than that observed in Table 2. For instance, in Table 2 the accuracy improvement for Bengali using BLSTM-BLSTM is 0.30% (from 90.84% to 91.14%), whereas in Table 3 the corresponding value is 3.06% (from 86.46% to 89.52%). These outcomes signify the fact that training with the ap-

	Bengali	Hindi	Latin	Spanish
BLSTM-BLSTM	71.06/72.10	87.80/88.18	60.85/61.63	88.06/88.79
BGRU-BGRU	70.44/71.22	88.34/88.40	60.65/61.52	91.48/92.25
Lemming	74.10	90.35	57.19	58.89
Morfette	70.27	88.59	47.41	57.61

Table 5: Lemmatization accuracy (in %) on unseen words.

	Bengali	Hindi	Latin	Spanish
BLSTM-BLSTM	56.16/66.26	87.42/88.41	49.80/56.05	86.22/87.97
BGRU-BGRU	59.45/66.84	87.19/88.26	50.24/55.35	86.74/88.49

Table 6: Lemmatization accuracy (in %) on unseen words without using applicable edit trees in training.

plicable edit trees already learns to dispense the output probability to the legitimate classes over which, output restriction cannot yield much enhancement.

Results for Unseen Word Forms: Next, we discuss about the lemmatization performance on those words which were absent in the training set. Table 4 shows the proportion of unseen forms averaged over 4 folds on the datasets. In Table 5, we present the accuracy obtained by our models and the baselines. For Bengali and Hindi, Lemming produces the best results (74.10% and 90.35%). For Latin and Spanish, BLSTM-BLSTM and BGRU-BGRU obtain the highest accuracy (61.63% and 92.25%) respectively. In Spanish, our model gets the maximum improvement over the baselines. BGRU-BGRU beats Lemming with 33.36% margin (on average, out of $9,011$ unseen forms, $3,005$ more tokens are correctly lemmatized). Similar to the results in Table 2, the results in Table 5 evidences that restricting the output in applicable classes enhances the lemmatization performance. The maximum accuracy improvements due to the output restriction are: 1.04% in Bengali (from 71.06% to 72.10%), 0.38% in Hindi (from 87.80% to 88.18%) using BLSTM-BLSTM and 0.87% in Latin (from 60.65% to 61.52%), 0.77% in Spanish (from 91.48% to 92.25%) using BGRU-BGRU.

Further, we investigate the performance of our models trained without the applicable edit trees information, on the unseen word forms. The results are given in Table 6. As expected, for every model, the accuracy drops compared to the results shown in Table 5. The only exception that we find out is in the entry for Hindi with BLSTM-BLSTM. Though without restricting the output, the accuracy in Table 5 (87.80%) is higher than the corresponding value in Table 6 (87.42%), but after output restriction, the performance changes (88.18% in Table 5, 88.41% in Table 6) which reveals that only selecting the maximum probable class over the applicable ones would be a better option for the unseen word forms in Hindi.

	Sem. Embedding	Syn. Embedding
Bengali	90.76/91.02	86.61/86.82
Hindi	94.86/94.86	91.24/91.25
Latin	88.90/89.09	85.31/85.49
Spanish	97.95/98	96.07/96.10

Table 7: Results (in %) obtained using semantic and syntactic embeddings separately.

	# Sentences	# Word Tokens
Catalan	14,832	474,069
Dutch	13,050	197,925
Hungarian	1,351	31,584
Italian	13,402	282,611
Romanian	8,795	202,187

Table 8: Dataset statistics of the 5 additional languages.

Effects of Semantic and Syntactic Embeddings in Isolation: To understand the impact of the combined word vectors on the model's performance, we measure the accuracy experimenting with each one of them separately. While using the semantic embedding, only distributional word vectors are used for edit tree classification. On the other hand, to test the effect of the syntactic embedding exclusively, output from the character level recurrent network is fed to the second level BGRNN. We present the results in Table 7. For Bengali and Hindi, experiments are carried out with the BLSTM-BLSTM model as it gives better results for these languages compared to BGRU-BGRU (given in Table 2). Similarly for Latin and Spanish, the results obtained from BGRU-BGRU are reported. From the outcome of these experiments, use of semantic vec-

	Catalan	Dutch	Hungarian	Italian	Romanian
BLSTM-BLSTM	97.93/**97.95**	93.20/**93.44**	91.03/**91.46**	96.06/**96.09**	94.25/**94.32**
Lemming	89.80	86.95	87.95	92.51	93.34
Morfette	89.46	86.62	86.52	92.02	94.13

Table 9: Lemmatization accuracy (in %) for the 5 languages.

tor proves to be more effective than the character level embedding. However, to capture the distributional properties of words efficiently, a huge corpus is needed which may not be available for low resourced languages. In that case, making use of syntactic embedding is a good alternative. Nonetheless, use of both types of embedding together improves the result.

3.2 Experimental Results for Another Five Languages

As mentioned earlier, five additional languages (Catalan, Dutch, Hungarian, Italian and Romanian) are considered to test the generalization ability of the method. The datasets are taken from the UD Treebanks[11] (Nivre et al., 2017). For each language, we merge the training and development data together and perform 4 fold cross validation on it to measure the average accuracy. The dataset statistics are shown in Table 8. For experimentation, we use the pre-trained semantic embeddings released by (Bojanowski et al., 2016). Only BLSTM-BLSTM model is explored and it is compared with Lemming and Morfette. The hyper parameters are kept same as described previously except for the number of epochs needed for training across the languages. We present the results in Table 9. For all the languages, BLSTM-BLSTM outperforms Lemming and Morfette. The maximum improvement over the baselines we get is for Catalan (beats Lemming and Morfette by 8.15% and 8.49% respectively). Similar to the results in Table 2, restricting the output over applicable classes yields consistent performance improvement.

4 Conclusion

This article presents a neural network based context sensitive lemmatization method which is language independent and supervised in nature. The proposed model learns the transformation patterns between word-lemma pairs and hence, can handle the unknown word forms too. Additionally, it does not rely on human defined features and various morphological tags except the gold lemma annotated continuous text. We explore different variations of the model architecture by changing the type of recurrent units. For evaluation, nine languages are taken as the references. Except Bengali, the proposed method outperforms the state-of-the-art models (Lemming and Morfette) on all the other languages. For Bengali, it produces the second best performance (91.14% using BLSTM-BLSTM). We measure the accuracy on the partial data (keeping the data size comparable to the Bengali dataset) for Hindi, Latin and Spanish to check the effect of the data amount on the performance. For Hindi, the change in accuracy is insignificant but for Latin and Spanish, accuracy drops by 3.50% and 6% respectively. The time requirement of the proposed method is also analyzed. Training time depends on several parameters such as size of the data, number of epochs required for convergence, configuration of the system used etc. In our work, we use the ‘keras’ software keeping ‘theano’ as backend. The codes were run on a single GPU (Nvidia GeForce GTX 960, 2GB memory). Once trained, the model takes negligible time to predict the appropriate edit trees for test words (e.g. 844 and 930 words/second for Bengali and Hindi respectively). We develop a Bengali lemmatization dataset which is definitely a notable contribution to the language resources. From the present study, one important finding comes out that for the unseen words, the lemmatization accuracy drops by a large margin in Bengali and Spanish, which may be the area of further research work. Apart from it, we intend to propose a neural architecture that accomplishes the joint learning of lemmas with other morphological attributes.

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

References

David Bamman and David Smith. 2012. Extracting two thousand years of latin from a million book library. *J. Comput. Cult. Herit.* 5(1):2:1–2:13. https://doi.org/10.1145/2160165.2160167.

Pushpak Bhattacharyya, Ankit Bahuguna, Lavita Talukdar, and Bornali Phukan. 2014. Facilitating multi-lingual sense annotation: Human mediated lemmatizer. In Heili Orav, Christiane Fellbaum, and Piek Vossen, editors, *Proceedings of the Seventh Global Wordnet Conference*. Tartu, Estonia, pages 224–231. http://www.aclweb.org/anthology/W14-0130.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606* https://arxiv.org/abs/1607.04606.

Cristian Cardellino. 2016. Spanish Billion Words Corpus and Embeddings. http://crscardellino.me/SBWCE/.

Abhisek Chakrabarty, Akshay Chaturvedi, and Utpal Garain. 2016. A neural lemmatizer for bengali. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France, pages 2558–2561. http://www.lrec-conf.org/proceedings/lrec2016/pdf/955$_paper.pdf$.

Abhisek Chakrabarty and Utpal Garain. 2016. Benlem (a bengali lemmatizer) and its role in wsd. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 15(3):12:1–12:18. https://doi.org/10.1145/2835494.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* https://arxiv.org/abs/1409.1259.

Grzegorz Chrupała. 2008. *Towards a machine-learning architecture for lexical functional grammar parsing*. Ph.D. thesis, Dublin City University. http://doras.dcu.ie/550/.

Grzegorz Chrupala, Georgiana Dinu, and Josef van Genabith. 2008. Learning morphology with morfette. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), Marrakech, Morocco. http://www.lrec-conf.org/proceedings/lrec2008/pdf/594$_paper.pdf$.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The sigmorphon 2016 shared task—morphological reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*. Association for Computational Linguistics, Berlin, Germany. http://aclweb.org/anthology/sigmorphon.html.

Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. 2015. Equilibrated adaptive learning rates for non-convex optimization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, NIPS'15, pages 1504–1512. http://dl.acm.org/citation.cfm?id=2969239.2969407.

Abu Zaher Md Faridee, Francis M Tyers, et al. 2009. Development of a morphological analyser for bengali. In *Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation*. Universidad de Alicante. Departamento de Lenguajes y Sistemas Informáticos, pages 43–50. http://www.mt-archive.info/FreeRBMT-2009-Faridee.pdf.

Andrea Gesmundo and Tanja Samardzic. 2012. Lemmatisation as a tagging task. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Jeju Island, Korea, pages 368–372. http://www.aclweb.org/anthology/P12-2072.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* https://arxiv.org/abs/1308.0850.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*. Association for Computational Linguistics, Boulder, Colorado, pages 1–18. http://www.aclweb.org/anthology/W09-1201.

Dag TT Haug and Marius Jøhndal. 2008. Creating a parallel treebank of the old indo-european bible translations. In *Proceedings of the Second Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2008)*. pages 27–34.

Mitesh Khapra, Anup Kulkarni, Saurabh Sohoney, and Pushpak Bhattacharyya. 2010. All words domain adapted wsd: Finding a middle ground between supervision and unsupervision. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Uppsala, Sweden, pages 1532–1541. http://www.aclweb.org/anthology/P10-1155.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* https://arxiv.org/abs/1412.6980.

Kimmo Koskenniemi. 1984. A general computational model for word-form recognition and production. In *Proceedings of the 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Stanford, California, USA, pages 178–181. https://doi.org/10.3115/980491.980529.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form:

Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 1520–1530. http://aclweb.org/anthology/D15-1176.

Aki Loponen and Kalervo Järvelin. 2010. A dictionary and corpus independent statistical lemmatizer for information retrieval in low resource languages. In *Multilingual and Multimodal Information Access Evaluation*, Springer, pages 3–14. https://doi.org/10.1007/978-3-642-15998-5$_3$.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Curran Associates Inc., USA, NIPS'13, pages 3111–3119. http://dl.acm.org/citation.cfm?id=2999792.2999959.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Atlanta, Georgia, pages 746–751. http://www.aclweb.org/anthology/N13-1090.

Thomas Müller, Ryan Cotterell, Alexander Fraser, and Hinrich Schütze. 2015. Joint lemmatization and morphological tagging with lemming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2268–2274. http://aclweb.org/anthology/D15-1272.

Garrett Nicolai and Grzegorz Kondrak. 2016. Leveraging inflection tables for stemming and lemmatization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 1138–1147. http://www.aclweb.org/anthology/P16-1108.

Joakim Nivre, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebirolu Eryiit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çar Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh Gökrmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà M, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phng Lê Hng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Lng Nguyn Th, Huyn Nguyn Th Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalnia, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0. LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics, Charles University. http://hdl.handle.net/11234/1-1983.

Snigdha Paul, Nisheeth Joshi, and Iti Mathur. 2013. Development of a hindi lemmatizer. *International Journal of Computational Linguistics and Natural Language Processing* 2(5):380–384. https://arxiv.org/abs/1305.6211.

Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. 2004. A rule based approach to word lemmatization. *Proceedings of IS-2004* pages 83–86.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, Suntec, Singapore, pages 486–494. http://aclweb.org/anthology/P/P09/P09-1055.pdf.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* https://arxiv.org/abs/1212.5701.