Syllables for Sentence Classification in Morphologically Rich Languages

Madhuri Tummalapalli
Language Technologies Research Center
KCIS
IIIT Hyderabad
madhuri.tummalapalli@research.iiit.ac.in

Radhika Mamidi
Language Technologies Research Center
KCIS
IIIT Hyderabad
radhika.mamidi@iiit.ac.in

Abstract

Sentence Classification is one of the most fundamental tasks in NLP, where the aim is to classify a given sentence into a pre-defined set of classes. A lot of work has been done in English in the last few years, which vary in their methodologies. A huge proportion of these works represent the input sentences as a sequence of words in their models. Only a few of them rely on character level representation. Through this work, we introduce a new method for representing a sentence - as a sequence of *syllables*. As we show in this work, syllables are a better choice to represent the sub-word level information in a sentence, which is essential for morphologically rich languages. We consider the tasks of Sentiment Analysis and Question Classification in three languages showing varied morphological richness - English, Hindi and Telugu. Through extensive evaluation, we show that syllables are the best performing input type when compared to words or characters for the morphologically rich languages - Hindi and Telugu.

1 Introduction

Sentence Classification is one of the most fundamental tasks in Natural Language Processing, where a given sentence is required to be classified into a pre-defined set of classes. This pre-defined set depends upon the specific task under consideration, which could be any of Sentiment Analysis, Question Classification, Subjectivity Analysis etc. There has been a lot of work done on each of these tasks separately, where the features and techniques chosen for classification are specific to the particular task (Silva et al., 2011; Huang et al., 2008; Pang and Lee, 2005a; Kouloumpis et al., 2011). However, the recent popularity and developments in Neural Networks have enabled researchers to build classifier models that are more generic in nature (Kim, 2014; Kalchbrenner et al., 2014). Given sufficient training data, these networks have the ability to learn task specific features for classification. Since these models do not require hand-crafted features, they make an attractive choice for the construction of task-independent or language-independent systems.

There are a huge number of languages used in different parts of the world and it is difficult and expensive to build resources for each language. Also, building an end-to-end task-specific language-specific system for every major language is a very tedious and expensive task. We, thus need to build systems which are generic in nature.

Most of the systems and techniques that have been built previously rely on word-level input. A few works use character level input (Zhang et al., 2015; Zhang and LeCun, 2015; Tummalapalli et al., 2018), as they show the benefit of being able to capture sub-word level or morphological information and are also helpful in dealing with Out Of Vocabulary (OOV) words. This is especially important in the case of morphological rich languages where a number of morphemes fuse together in a single word to express different types of grammatical or syntactical information. For classifying such languages, it is essential to move beyond word level, and capture sub-word level information.

This leads us to the idea of using sequence of syllables instead of characters as input for classifica-

32nd Pacific Asia Conference on Language, Information and Computation
Hong Kong, 1-3 December 2018

tion. Representing a sentence as a sequence of syllables, rather than as a sequence of character-ngrams reduces the noise created due to character-ngrams, especially for the case of morphologically rich languages such as Hindi and Telugu. We thus, through this work, introduce the idea of using syllables, in addition to words and characters, to represent a sentence in classification tasks.

Also, syllabification itself, in our opinion, does not become an extremely costly affair. This can be seen from the rules used for syllabification in Hindi and Telugu, listed in Section 5.1. These rules only require that the list of vowels and consonants be known for any given language. Or, a better method might be to convert any given linguistic text into IPA format. Since, the list of vowels and consonants is well defined for IPA, syllabification becomes almost language independent. Also, IPA is a standard for representing the spoken language and has been around for years now[1]. While converting a given text into IPA is language specific, it is not specific to our task and thus many researchers have already invested their time on it. In addition, the rules used in the paper, yield syllables for a given word in linear time (according to its length). Although, in our case we used a hyphenation library for English for simplicity, which is also available for several other languages Danish, French, Greek, Italian etc.

In this paper, we consider the task of Sentence Classification with five datasets, three for Sentiment Analysis and two for Question Classification, in three languages - English, Hindi and Telugu. Hindi and Telugu are relatively free-word order languages when compared to English. Also, the morphological richness of these languages increases from English to Hindi (an Indo-Aryan language) to Telugu (a Dravidian language). We compare the performance of syllables against words and characters with the help of a CNN based network, as it has been found to perform well for sentence classification. We then combine different types of inputs in an ensemble based *multiInput-CNN* model described by Tummalapalli et al. (2018). Our main contributions are:

- The idea of using syllable based input for sentence classification for morphologically rich languages.
- Extensive comparison of word, character and syllable based input for classification with the help of CNN based models.

The rest of the paper is organized as follows: In Section 2 we state the various reasons for choosing to use syllables for sentence representation instead of characters or words. Thereafter, Section 3 lists the related work. In Section 4 we describe the datasets used in our experiments. In Section 5 we describe the methodologies we follow, which is followed by a discussion of our results in Section 6 and conclusions and future work in Section 7.

2 Motivation

We had stated in Section 1 that the use of character n-grams can lead to better results, particularly for the morphologically rich languages - Telugu and Hindi. Morphologically rich languages are those languages in which a number of morphemes fuse together in a single word to express different types of grammatical or syntactical information (Tsarfaty et al., 2010). For classifying such languages, it is essential to move beyond word level, and capture sub-word level information.

In many languages, such as in most Indian languages, morphemes can be seen as ngrams of syllables. For example, in the Hindi verbs खाकर (KAkara[2]) or जाकर (jAkara) the roots खा (KA) or जा (jA) form a syllable while the common suffix -कर (-kara) (meaning 'after') forms the second and third syllable in both the words. In English, the word *beautiful* breaking into its syllables *beau-ti-ful* is a similar example.

Since the character-ngrams used for classification are generally overlapping in nature, every character n-gram obtained by splitting a word is not necessarily meaningful or important. There will be character n-grams which might be approximating morphemes. But, a huge proportion of these might be leading to noise. Consider, for example, the word and its constituent n-grams with $n = 2$ as given in Figure 1, *[खा]* ([KA]) and *[इए]* ([ie]), here, are n-grams that may help contribute towards the captur-

[1] https://en.wikipedia.org/wiki/International_Phonetic_Alphabet

[2] We have used the WX notation to transliterate Devanagari text for readability.

32nd Pacific Asia Conference on Language, Information and Computation
Hong Kong, 1-3 December 2018

ing of morphological information by the classifiers, while *[ाइ]* ([Ai]) may not be important and thus lead to noise.

Word
खाइए (KAie)

N-Gram Split
[खा] ([KA]) [ाइ] ([Ai]) [इए] ([ie])

Figure 1: Example showing how overlapping character-ngrams can lead to noise.

Also, in most Indian languages, each written character in the native script is a syllable (Kishore and Black, 2003). Although, when written in Unicode, it is possible that a single such character is broken down into two or more Unicode characters (code points), as in the case of की (kI). In Devanagari it is written as a single character, but in Unicode it is written using two characters one each for क (k) and ी (I). When using character-ngrams as input, we may have ngrams where the two characters क (k) and ी (I) are split into two different ngrams. This leads to noise. To avoid this, it is intuitive to use syllables as the basic unit for classification. Syllables too, like character-ngrams, can be useful in dealing with OOV words with the added advantage that they would be lesser in number than character-ngrams.

Another problem in capturing morphological information in character-ngrams is that, n is usually a constant referring to the number of characters grouped together in a single unit. However, not all morphemes are of the same length. This would imply, that the character-ngrams would not be able to successfully capture all the morphemes, but only those whose length is equal to the chosen ngram size. This is not the case with syllables, since each syllable may contain any number of characters in it. A simple example has been given in Figure 2 which shows a word and its split into constituent n-grams when $n = 2$, and a meaningful split of the same word into its morphemes. Out of the generated four n-grams shown in the n-gram split of the word, only two ([खा] ([KA]) and [गा] ([gA])) carry grammatical information. The other two are meaningless. While the morpheme - [ए] ([e]) - is not even generated as its size is 1.

Word
खाएगा (KAegA)

N-Gram Split
[खा] ([KA]) [ाए] ([Ae]) [एग] ([eg]) [गा] ([gA])

Meaningful Split
[खा] ([KA]) [ए] ([e]) [गा] ([gA])

Figure 2: Example showing how a fixed ngram size leads to noise in character-ngrams.

3 Related Work

A lot of work has been done in task-independent Sentence Classification in the last few years. Scholars have used a number of different neural network architectures for the purpose. Kim (2014) and Kalchbrenner et al. (2014) used CNNs for the task. These works were amongst the earliest on task-independent classification, and were later extended and modified by several works (Gan et al., 2017; Zhang et al., 2017; Li et al., 2017). Zhang et al. (2015), Zhang and LeCun (2015) and Becker et al. (2017) used very deep CNNs to extract information from character level input. Wehrmann et al. (2017) on the other hand, using a model similar to that of Kim (2014), showed that a single convolution layer with character embeddings as input is good enough for classification purposes. These works only used character unigrams as input. Tummalapalli et al. (2018) showed that character-ngram input can lead to much better performance in morphologically rich languages as compared to character unigrams.

Recurrent and recursive networks were also used (Socher et al., 2013; Dong et al., 2014; Irsoy and Cardie, 2014), some of them depend upon parse trees, which makes it difficult to adapt them to languages with scarce resources. It has also been noted that CNNs often give better performance than LSTMs in sentence classification (Kim, 2014; Tummalapalli et al., 2018). Yang et al. (2016) explored hierarchical attention networks for classification, while Zhang et al. (2016a) combined LSTMs and CNNs for a dependency sensitive model. In this paper, we experiment using models described by Tummalapalli et al. (2018) which was an extension of Kim (2014).

Little work has been done in Hindi and Telugu

sentence classification. Although, significant work has been done in Hindi and Telugu sentiment analysis (Singhal and Bhattacharyya, 2016; Mukku et al., 2016; Mukku and Mamidi, 2017). Tummalapalli et al. (2018) compared various deep architectures for morphologically rich languages and proposed an ensemble based model for sentence classification, where multiple CNN models trained on different inputs were combined by taking the average of the final layer outputs. CNN models combining multiple inputs were also proposed by Yin and Schütze (2016), who combined different types of word embeddings at the convolution stage, and Zhang et al. (2016b), who performed independent convolution operations on all inputs and combined them at the penultimate layer.

Syllables have been used as a basic processing unit in speech technology for quite some time now. Ganapathiraju et al. (2001) used syllables instead of triphone units as they are lesser in number and more intuitive for the task of speech recognition. While Schrumpf et al. (2005) used syllables to accommodate for OOV words, especially in morphologically rich or agglutinating languages. These works provide further evidence towards our motivation for using syllables as a basic unit for classification as compared to using character-ngrams for the same.

In text processing, very little work has been done using syllables. Recently, Nguyen et al. (2009) and Qun et al. (2017) used syllables for classifying text in languages which don't have clear word boundaries, for example, Vietnamese or Tibetan in these cases. Nguyen et al. (2009) used a bag-of-syllables based representation along with resources like infogain for feature selection for sentence classification in Vietnamese using SVM, KNN and Naive Bayes. While Qun et al. (2017) drew a comparison between RNNs, CNNs etc. with a combined syllable and character based representation for sentence classification in Tibetan.

4 Datasets

In this paper, we use five datasets for the purpose of our evaluation. Three of which are Sentiment Analysis datasets in three languages - English, Hindi and Telugu. And two are for Question Classification in two languages - English and Hindi.

The English Sentiment Analysis dataset is the *MR* dataset (Pang and Lee, 2005b). It consists of a total of 10662 annotated sentences from the movie review domain labeled for *positive* and *negative* sentiment. The Hindi dataset also consists of annotated sentences from the movie review domain. The Hindi dataset, *Senti-Hi*, and the Telugu dataset, *Senti-Te*, have been labeled for *positive*, *negative* and *neutral* sentiment. The Telugu dataset (Mukku and Mamidi, 2017) consists of 5409 annotated sentences, while the Hindi one has 6580 annotated sentences. Some statistics for the sentiment analysis datasets have been listed in Table 1. Since none of the Sentiment Analysis datasets consist of a train-test split, we perform 10-fold cross-validation to obtain results on them.

Dataset	Positive	Neutral	Negative
MR	5331	-	5331
Senti-Hi	2240	3408	932
Senti-Te	1491	2477	1441

Table 1: Statistics for Sentiment Analysis datasets. The table shows the number of sentences of each sentiment in different datasets.

The other two datasets are for Question Classification in English and Hindi. The English dataset, *TREC-En*, is the TREC-UIUC[3] dataset (Li and Roth, 2002). It has 6 core classes, namely *Abbreviation*, *Description*, *Entity*, *Human*, *Location*, *Numeric*, and 50 fine classes. This dataset was released with a train-test split, with 5452 questions in the train set and 500 in the test set. The Hindi dataset, *TREC-Hi* (Tummalapalli et al., 2018) was prepared by translating the *TREC-En* dataset into Hindi. The classes are the same in both the datasets. This dataset consists of 5444 questions in the train set and 499 in test set. We consider classification into the 6 core classes for this work. Statistics for the question classification datasets have been given in Table 2.

5 Methodology

5.1 Syllable Extraction

For English, syllabification was done using the python library *PyHyphen*[4]. Given a word as input,

[3] http://cogcomp.cs.illinois.edu/Data/QA/QC/

[4] https://pypi.org/project/PyHyphen

Abbreviation	Class	TREC-En		TREC-Hi	
		Train	Test	Train	Test
DESC	Description	1162	138	1162	138
ENTY	Entity	1250	94	1248	93
ABBR	Abbreviation	86	9	86	9
NUM	Number	1223	65	1219	65
HUM	Human	896	113	895	113
LOC	Location	835	81	834	81

Table 2: Statistics for Question Classification datasets. The table shows the number of questions in the train and test set for different core classes.

this library can be used to break it into its syllables. The library works on the C library libhyphen[5], which in turn is based on the Knuth-Liang Algorithm (Liang, 1983) for hyphenation. Liang (1983) base their hyphenation algorithm or the hyphenation rules on those used for syllabification.

For Hindi, we performed syllabification using the *LibIndic Syllabifier* module of the *LibIndic*[6] library. This library currently performs syllabification for the Indian languages Malayalam, Kannada, Bengali, Tamil and Hindi. Given the list of *matras* (or vowel-signs), *limiters* (or punctuations) and the *virama* (or *halant*) symbol[7] for a script, it uses a set of rules to identify the syllables, which are given as follows:

- If the given character is a *limiter*, it is considered as a separate syllable.
- If a given character is a *matra* it is added to the previous syllable.
- If the last character of the previous syllable is a *virama*, the current character is added to the previous syllable.
- Else, the current character starts a new syllable.

We extended the existing code for syllabification to Telugu by simply adding the required lists of *matras*, *limiters* and the *virama* symbol which were obtained using the Unicode chart for Telugu[8].

[5] https://github.com/hunspell/hyphen

[6] https://libindic.org

[7] https://en.wikipedia.org/wiki/Virama

[8] https://unicode.org/charts/PDF/U0C00.pdf

5.2 CNN Architecture

We borrow our CNN architecture from (Kim, 2014). Kim (2014) define a number of different CNN architectures with minor variations. We show the basic architecture in Figure 3. It takes a sentence represented as a sequence of word embeddings as input. That is, a sentence s with n words, and each word being represented by a word embedding of dimension d, will have a representation of size $n \times d$. This is followed by a convolution layer. Kim (2014), here, introduced the idea of having multiple filters of different window sizes. The convolution layer is applied separately for filters with different sizes. This is then followed by a global max pool layer. The outputs of this layer, from convolution filters of different sizes are concatenated together. This is followed by a dropout layer and a fully connected layer with softmax activation, which gives the classification output.

For our experiments, we use their *CNN-rand* and *CNN-non-static* models, which initialize the embedding layer randomly and with word2vec (Mikolov et al., 2013) word embeddings respectively. We implemented this model with the help of *Keras* library[9] (Chollet and others, 2015).

5.3 multiInput-CNN Architecture

We borrow this architecture from Tummalapalli et al. (2018). They apply the complete CNN architecture as described in Section 5.2, in particular *CNN-non-static* and *CNN-rand*, on two different inputs, words and character-ngrams, and take the average of the output of the final fully connected layer to get the final result. The network is trained using a *categorical crossentropy* loss function. In this work, we experiment with different combinations of inputs: words, character-ngrams and syllable-ngrams, to study their performance over different datasets. Figure 4 illustrates the architecture for two types of inputs. In this figure, the multiple convolution filters along with the max-pooling layer and the concatenation of their outputs have been shown by a single box.

The motivation behind using such an architecture is to be able to combine the contextual information from pre-trained word vectors and the sub-word

[9] https://keras.io/

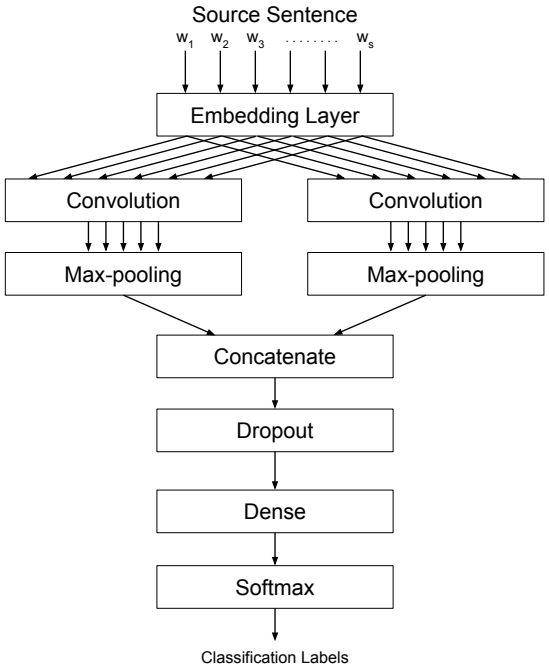

Figure 3: CNN Architecture as proposed by Kim (2014).

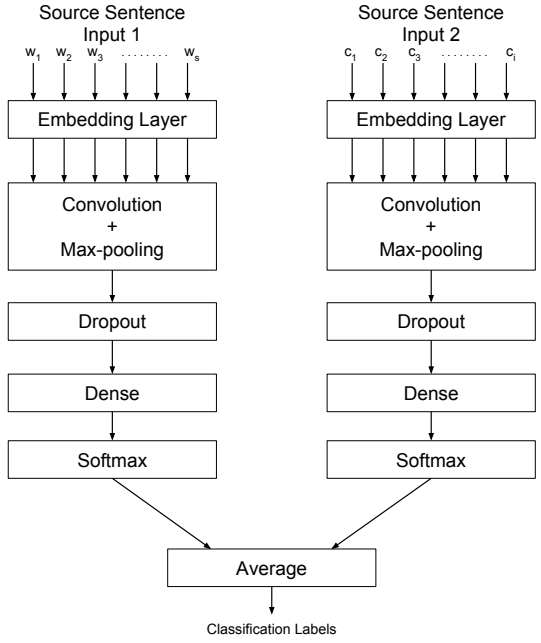

Figure 4: multiInput-CNN architecture as proposed by Tummalapalli et al. (2018)

level information from characters, or in our case, from syllables.

6 Experiments and Results

To evaluate the performance of syllables in sentence classification, we first conducted experiments with the *CNN-rand* architecture described in Section 5.2. We compare word (*rand-word*), character-ngram (*rand-char*) and syllable-ngram (*rand-syl*) inputs for classification in Table 3.

In all the experiments, the model parameters including the sizes of character-ngrams and syllable-ngrams were found through parameter tuning. For syllables, we chose the best out of syllable unigrams, bigrams and trigrams. The results reported are the average accuracies for 10 runs for experiments performed with the *CNN-rand* and *w2v-word* models. For experiments on *multiInput-CNN* listed in Table 4, the results reported are average accuracies of 5 runs.

As can be observed in Table 3, syllable level input leads to the best performance when classifying using a CNN network in all the three Indian language datasets.

The better performance of character and syllable based input in Hindi and Telugu shows the importance of capturing sub-word level information for morphologically rich languages. This also confirms that syllables capture morphological or sub-word level information as efficiently as characters. In fact, the better performance of syllables as compared to characters suggests that the use of syllables for input might be reducing the additional noise caused due to unwanted character-ngrams, at the same time capturing a larger number of meaningful morphological units, as was hypothesized in Section 2.

Input	MR	Senti-Hi	Senti-Te	TREC-En	TREC-Hi
rand-word	74.97	71.02	53.54	**90.74**	86.19
rand-char	**77.43**	72.73	57.97	87.80	85.35
rand-syl	76.63	**72.97**	**58.09**	90.60	**87.05**

Table 3: Performance of different inputs on *cnn-rand*. **Highlighted** are the best results for each dataset.

For English, we observe that while both the character and syllable based inputs perform better than word level input for the MR dataset, word based input performs the best for TREC-En. Thus, we find that, the helpfulness of sub-word level or morphological information in English is inconclusive.

Next, we conducted experiments with the

Model-Input	MR	Senti -Hi	Senti -Te	TREC -En	TREC -Hi
word+char	**81.03**	74.51	58.86	**93.96**	92.22
syl+char	78.24	73.36	58.31	90.60	86.17
word+syl	80.66	**74.71***	**59.21***	93.68	**92.79**
word+syl+char	80.15	73.92	58.91	92.80	89.94
w2v-word	80.21	73.86	55.50	93.24	**93.71***
CNN-non-static Kim (2014)	81.50	-	-	92.80	-
Gan et al. (2017)	77.77	-	-	92.60	-
Li et al. (2017)	82.10	-	-	94.40	-
Zhang et al. (2016a)	**82.20***	-	-	85.60	-
Zhang et al. (2016b)	-	-	-	**95.52***	-

Table 4: Performance of different input combinations on *multiInput-CNN* and other baselines. The best result in *multiInput-CNN* has been **highlighted**. The overall best result has an additional * over it.

multiInput-CNN model mentioned in Section 5.3. We experimented with all combinations of different input types: words (*word*) (with word2vec initialization), character-ngram (*char*) and syllable-ngram (*syl*).

In these experiments we initialized the word vectors with those trained using word2vec on Wikipedia English, Hindi and Telugu data[10]. We chose to use the vectors trained on Wikipedia data for all the three languages, in order to maintain uniformity during comparison.

We also mention the results of using word2vec vectors in the CNN network (*w2v-word*) (similar to *CNN-non-static* (Kim, 2014)) and other important works on MR and TREC-En datasets. Table 4 compares all these results.

We observe that *w2v-word* performs better than *CNN-rand* (all inputs) in most cases, because of the added contextual information from pre-trained vectors. Whereas, for Telugu, *rand-char* and *rand-syl* perform better than *w2v-word*. This is because of the huge number of OOV words (7541 out of a total of 24625 words in our case), arising due to the high number of possible inflections and agglutinations.

multiInput-CNN thus tries to take advantage of both the contextual information found in pre-trained word vectors and the sub-word level information in characters or syllables.

We also note that *word+syl* which combines word and syllable-ngrams for input, outperforms *word+char* input combining word and character-ngram for all the Indian language datasets. In addition, it outperforms all other combinations including *w2v-word* for two out of three Indian language datasets, indicating we are successfully able to harness the contextual information from pre-trained vectors and sub-word information with the help of syllables. For TREC-Hi as well, *w2v-word* performs only slightly better than *word+syl* mostly because Hindi is less morphologically rich when compared to Telugu. Also, the presence of Hindi on the web is much more as compared to Telugu, thus making its word embeddings richer. Although, syllable based input does not seem to work well for English, probably because many morphemes in English are not syllables, for example the plurality morpheme *-s*.

7 Conclusions and Future Work

In this paper, we introduced the idea of using syllable-level input instead of word or character level inputs in NLP tasks for morphologically rich and agglutinating languages. We evaluated syllable level input on two different neural network architectures and found that it performs the best for morphologically rich languages. In addition, using syllables along with pre-trained word input, helps us in harnessing both the contextual information and essential sub-word level information. We also found that for the morphologically rich languages, the combination of words and syllable level input outperformed the combination of words and character level input.

It can be observed from Table 4 that the multiInput-CNN model performs best with a combination of words and syl-ngrams for Hindi and Telugu language datasets, while a combination of word and char-ngrams performed best for the English language datasets. Thus, as a future work, we can experiment with attention based models, which can choose the best combination of inputs for the multiInput-CNN model. In addition, one can focus on pre-training embeddings for character n-grams and syllable n-grams, such that they are able to cap-

[10] https://dumps.wikimedia.org/

ture both morphological and contextual information.

Acknowledgments

The authors would like to thank Ayush Tripathi for the Senti-Hi dataset. The authors would also like to thank everyone who contributed towards this paper by providing their valuable suggestions and advices.

References

Willian Becker, Jnatas Wehrmann, Henry Cagnini, and Rodrigo Barros. 2017. An efficient deep neural architecture for multilingual sentiment analysis in twitter. 05.

François Chollet et al. 2015. Keras. https://github.com/keras-team/keras.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *AAAI*, pages 1537–1543.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2390–2400.

Aravind Ganapathiraju, Jonathan Hamaker, Joseph Picone, Mark Ordowski, and George R Doddington. 2001. Syllable-based large vocabulary continuous speech recognition. *IEEE Transactions on speech and audio processing*, 9(4):358–366.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. 2008. Question classification using head words and their hypernyms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 927–936. Association for Computational Linguistics.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in neural information processing systems*, pages 2096–2104.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

S Prahallad Kishore and Alan W Black. 2003. Unit size in unit selection speech synthesis. In *Eighth European Conference on Speech Communication and Technology*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna D Moore. 2011. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Shen Li, Zhe Zhao, Tao Liu, Renfen Hu, and Xiaoyong Du. 2017. Initializing convolutional filters with semantic features for text classification. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1885–1890.

Franklin Mark Liang. 1983. Word hy-phen-a-tion by com-put-er. Technical report, Calif. Univ. Stanford. Comput. Sci. Dept.

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

Sandeep Sricharan Mukku and Radhika Mamidi. 2017. Actsa: Annotated corpus for telugu sentiment analysis. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 54–58.

Sandeep Sricharan Mukku, Nurendra Choudhary, and Radhika Mamidi. 2016. Enhanced sentiment classification of telugu text using ml techniques. In *SAAIP@ IJCAI*, pages 29–34.

Giang-Son Nguyen, Xiaoying Gao, and Peter Andreae. 2009. Text categorization for vietnamese documents. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 03*, pages 466–469. IEEE Computer Society.

Bo Pang and Lillian Lee. 2005a. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 115–124. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005b. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Nuo Qun, Xing Li, Xipeng Qiu, and Xuanjing Huang. 2017. End-to-end neural text classification for tibetan. In *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data*, pages 472–480. Springer.

Christian Schrumpf, Martha Larson, and Stefan Eickeler. 2005. Syllable-based language models in speech recognition for english spoken document retrieval. In *Proc. of the 7th International Workshop of the EU Network of Excellence DELOS on AVIVDiLib, Cortona, Italy*, pages 196–205.

Joao Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2011. From symbolic to sub-symbolic

information in question classification. *Artificial Intelligence Review*, 35(2):137–154.

Prerana Singhal and Pushpak Bhattacharyya. 2016. Borrow a little from your rich cousin: Using embeddings and polarities of english words for multilingual sentiment classification. In *COLING*, pages 3053–3062.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (spmrl): what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12. Association for Computational Linguistics.

Madhuri Tummalapalli, Manoj Chinnakotla, and Radhika Mamidi. 2018. Towards better sentence classification for morphologically rich languages. *Polibits*, page to appear.

Joonatas Wehrmann, Willian Becker, Henry EL Cagnini, and Rodrigo C Barros. 2017. A character-based convolutional neural network for language-agnostic twitter sentiment analysis. In *Neural Networks (IJCNN), 2017 International Joint Conference on*, pages 2384–2391. IEEE.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489.

Wenpeng Yin and Hinrich Schütze. 2016. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Rui Zhang, Honglak Lee, and Dragomir Radev. 2016a. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*.

Ye Zhang, Stephen Roller, and Byron Wallace. 2016b. Mgnc-cnn: A simple approach to exploiting multiple word embeddings for sentence classification. *arXiv preprint arXiv:1603.00968*.

Ye Zhang, Matthew Lease, and Byron C Wallace. 2017. Active discriminative text representation learning. In *AAAI*, pages 3386–3392.