Sentence Classification for Morphologically Rich Languages

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computational Linguistics by Research

by

Tummalapalli Madhuri
201325191
madhuri.tummalapalli@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2018

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Sentence Classification in Morphologically Rich Languages" by Tummalapalli Madhuri, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Radhika Mamidi

This thesis is dedicated to my family, adviser and friends for their endless support and encouragement.

Acknowledgments

This work would not have been completed without the support, encouragement, advises and suggestions of a number of people.

Firstly, I would like to thank my parents and sister for their constant encouragement and motivation to put in my best efforts and complete my thesis.

This work would definitely not have been completed if it was not for the timely advises given by Dr. Radhika Mamidi, Dr. Manoj Chinnakotla and Dr. Manish Srivastava.

I would also like to thank my friends Kaveri, Nisarg, Pravallika and Srishti for the endless discussions and suggestions that made the target of completing this thesis achievable.

Abstract

Sentence Classification is one of the most fundamental tasks in Natural Language Processing (NLP), wherein the aim is to classify any given sentence into one of a given set of classes. The set of classes usually depends upon the specific task under consideration, such as determining the type of answer in Question Classification or finding the sentiment of a sentence in Sentiment Analysis or finding whether a sentence is subjective or objective in Subjectivity Analysis. These are some of the tasks that can be included in Sentence Classification. Some examples of other tasks being sarcasm or humour detection, detection of abusive text etc.

In sentence classification, the aim is to have a common network architecture for a range of classification tasks, rather than building an independent one for each task. In other words, the aim is to build a task-independent classification system.

Many methods have been developed for various sentence classification tasks for English, which usually exploit linguistic resources like parsers making it difficult to adapt them to other languages. There are a huge number of languages spoken around the world, and it is not possible to build an independent classifier for each task in every language. Thus, in this thesis, we explore language-independent task-independent sentence classification focusing on improvement in classification of morphologically rich languages. We perform sentence classification for five datasets in two tasks - Sentiment Analysis and Question Classification and three languages - English, Hindi and Telugu.

We present an evaluation of popular deep learning methods for sentence classification on the morphologically rich Indian languages, specifically Hindi and Telugu. For this purpose, we also created a question classification dataset for Hindi, by translating the TREC-UIUC English dataset. We show that character based input can enhance the performance of current classification systems for morphologically rich languages. Finally, we show that our proposed *multiInput-CNN* variant is able to perform better than our baselines in two out of three tasks in Hindi and Telugu, while giving comparable results for others.

Since a huge proportion of the work in sentence classification is specifically for English, the methods are often designed to best suit that language. A lot of the 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*. It is essential to capture sub-word level information while classifying in a morphologically rich language. Syllables are an ideal choice for this, as in many languages most of the morphemes can be seen as

n-grams of syllables. Thus, making them a more intuitive choice when compared with character n-grams to capture morphological information. Also, syllables can lead to a reduction in noise caused due to the usage of character n-grams. 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.

Finally, we make use of attention mechanism to automatically find the best combination of inputs for every dataset and language. We experiment with a number of different attention based CNN models in this part and find that attention can be used to obtain a model that performs decently well across all datasets.

Contents

Chapter Page

1 Introduction . . . 1
1.1 Sentence Classification . . . 1
1.1.1 Sentiment Analysis . . . 1
1.1.2 Question Classification . . . 2
1.2 Motivation . . . 2
1.2.1 Too Many Languages . . . 2
1.2.2 Morphologically Rich and Agglutinating Languages . . . 4
1.3 Thesis Overview . . . 5
1.4 Our Contributions . . . 6
1.5 Thesis Organization . . . 6

2 Related Work . . . 7
2.1 Task Specific Models . . . 7
2.1.1 Sentiment Analysis . . . 7
2.1.2 Question Classification . . . 8
2.2 Task Independent Models . . . 8
2.2.1 Convolutional Neural Network (CNN) . . . 8
2.2.2 Recurrent and Recursive Models . . . 8
2.2.3 Modifying Basic Models . . . 9
2.2.4 Attention Based Models . . . 10
2.3 Cross-Lingual Sentence Classification . . . 11
2.4 Sentence Classification in Hindi and Telugu . . . 11
2.5 Syllables . . . 11
2.6 Summary . . . 12

3 Datasets . . . 13
3.1 Question Classification Datasets . . . 13
3.1.1 TREC-UIUC Dataset (TREC-En) . . . 13
3.1.2 TREC Hindi Dataset (TREC-Hi) . . . 14
3.2 Sentiment Analysis Datasets . . . 15
3.2.1 Movie Reviews Dataset (MR) . . . 15
3.2.2 Sentiment Analysis - Hindi (Senti-Hi) . . . 16
3.2.3 Sentiment Analysis - Telugu (Senti-Te) . . . 16
3.3 Conclusion . . . 16

4 Baseline Experiments and Our MultiInput-CNN Model . . . 19
4.1 Methodology . . . 19
4.1.1 Input . . . 19
4.1.1.1 Sequence of Words . . . 19
4.1.1.2 Sequence of characters (char-1-gram) . . . 20
4.1.1.3 Sequence of character n-grams (char-ngram) . . . 20
4.1.2 Support Vector Machines . . . 21
4.1.3 Convolutional Neural Network (CNN) . . . 21
4.1.4 LSTM and Bi-LSTM . . . 23
4.1.5 Our multiInput-CNN variant . . . 23
4.2 Experiments and Results . . . 24
4.2.1 Evaluation of Baseline Models . . . 25
4.2.2 multiInput-CNN . . . 27
4.2.3 Experiments with Translated Datasets . . . 27
4.3 Conclusion . . . 28

5 Syllable Based Input for Sentence Classification . . . 29
5.1 Motivation . . . 29
5.2 Methodology . . . 30
5.2.1 Syllable Extraction for English . . . 31
5.2.2 Syllable Extraction for Hindi and Telugu . . . 31
5.3 Experiments and Results . . . 32
5.3.1 Experiments with CNN . . . 32
5.3.2 Experiments with MultiInput-CNN . . . 32
5.4 Conclusion . . . 34

6 Attention Mechanism For Combining Different Inputs . . . 35
6.1 Motivation . . . 35
6.2 Models and Methodology . . . 36
6.2.1 Attn-1 . . . 36
6.2.2 Attn-2 . . . 37
6.2.3 Attn-3 . . . 37
6.2.4 Attn-4 . . . 38
6.2.5 Attn-3-round and Attn-4-round . . . 38
6.3 Experiments and Results . . . 39
6.4 Conclusion . . . 42

7 Conclusions . . . 43

Appendix A: Experimental Settings and Model Parameters . . . 45
A.1 Support Vector Machines . . . 45
A.2 LSTMs . . . 45
A.3 Bi-LSTMs . . . 46
A.4 CNNs . . . 46
A.4.1 Random Word Vector Input . . . 46
A.4.2 Word2Vec Word Vector Input . . . 47
A.4.3 Character n-gram Input . . . 47

A.4.4 Syllable n-gram Input 47

Bibliography 49

List of Figures

Figure Page

3.1 Snippet of the English Question Classification Dataset (*TREC-En*). The answer-type label for a given question is followed by a space and the question itself. 14
3.2 Web interface used for validation of Hindi Question Classification Dataset (*TREC-Hi*). 15
3.3 Snippet of the Hindi Question Classification Dataset (*TREC-Hi*). As in *TREC-En*, here too the answer-type label for a given question is followed by a space and the question. . 16
3.4 Snippet of the English Sentiment Analysis Dataset (*MR*). The sentences are split into two files, one with positive reviews and the other with negative reviews. 17
3.5 Snippet of the Hindi Sentiment Analysis Dataset (*Senti-Hi*). Each sentence is separated from its sentiment label by a *tab* character. 17
3.6 Snippet of the Telugu Sentiment Analysis Dataset (*Senti-Te*). For each sentence, its sentiment label is separated with spaces from the sentence. 18

4.1 CNN Architecture as proposed by Kim [22]. 22
4.2 Figure shows the proposed *multiInput-CNN* architecture. 25

6.1 Attention in *multiInput-CNN* : *Attn-1* 37
6.2 Attention in *multiInput-CNN* : *Attn-2* 38
6.3 Attention in *multiInput-CNN* : *Attn-3* 39
6.4 Attention in *multiInput-CNN* : *Attn-4* 40

List of Tables

Table Page

1.1 Readership (in millions) of various language newspapers in India. 3

3.1 Statistics for Question Classification datasets. The table shows the number of questions in the train and test set for different core classes. 14
3.2 Statistics for Sentiment Analysis datasets. The table shows the number of sentences of each sentiment in different datasets. 15

4.1 Evaluation of baseline models. 26
4.2 Comparison of *multiInput-CNN* with the state-of-the arts. Each result has been reported by taking an average of 5 runs for *multiInput-CNN* and an average of 10 runs for others. 28
4.3 CNN Results on Translated datasets 28

5.1 Performance of different inputs on *CNN-rand*. **Highlighted** are the best results for each dataset. 32
5.2 Performance of different input combinations on *multiInput-CNN* and other baselines. . 33

6.1 Performance of different attention based models in comparison with our earlier models. Best results have been highlighted for attention and non-attention models. Overall best results have been marked with a *. 41

A.1 Model parameters for SVM experiments. In case the analyzer is *Word-Char*, the n-gram size is : *n-gram size range for words - n-gram size range for characters* 45
A.2 Model parameters for LSTM experiments. 46
A.3 Model parameters for Bi-LSTM experiments. 46
A.4 Model parameters for CNN *word-rand* experiments. 46
A.5 Model parameters for CNN *word-word2vec* experiments. 47
A.6 Model parameters for CNN *char-ngram* experiments. 47
A.7 Model parameters for CNN *syl-ngram* experiments. 47

Chapter 1

Introduction

1.1 Sentence Classification

Sentence Classification is one of the most fundamental tasks in Natural Language Processing (NLP), wherein, the aim is to classify any given sentence into one of a given set of classes. The set of classes usually depends upon the specific task under consideration, such as determining the type of answer in Question Classification or finding the sentiment of a sentence in Sentiment Analysis or finding whether a sentence is subjective or objective in Subjectivity Analysis. These are the most commonly considered sub-tasks when doing Sentence Classification in English [22, 21]. Some other tasks which could also be included are the detection of sarcasm, humour or abusive language in text.

Thus, in sentence Classification, the aim is to have a common network architecture for a range of classification tasks, rather than building an independent one for each task. In our work, we extended this notion of building a task-independent system to building a system that is both task and language independent. For this, we consider five datasets in three languages - English, Hindi and Telugu - and two tasks - Sentiment Analysis and Question Classification. In the following sub-sections, we define the individual tasks of Sentiment Analysis and Question Classification.

1.1.1 Sentiment Analysis

Recently, there has been a surge in the amount of data present on the web. A huge portion of which is user generated data. Users, generally humans, express their opinions about various things on social media platforms, be it movies, products, books, political parties, events etc. Mining this information can be useful for different stakeholders, as it gives them an idea about what people's opinions are. This can be achieved with the help of Sentiment Analysis.

In Sentiment Analysis, the aim is to classify a given piece of text, which could be either a sentence or a paragraph or a document, according to its sentiment. In most cases, the classification is between two (*positive* and *negative*) or three (*positive*, *negative* and *neutral*) classes. In other cases, the classification

can be more fine grained, such as between the classes - *very negative*, *negative*, *neutral*, *positive*, *very positive*. Some examples of sentences with their polarities are given in Example 1.1.

Example 1.1.	*On its own, it's not very interesting. As a remake, it's a pale imitation.*	**Negative**
	A thoughtful, provocative, insistently humanizing film.	**Positive**

1.1.2 Question Classification

A rapidly growing knowledge base of the Internet also implies that we need a web-scale question-answer system which can respond to a users' query with a concise response, rather than with a set of links. Such a system can respond to a users' need for information, while requiring minimal effort from his/her side. A question classifier is a module in the query processing stage of such a system [20]. The task of this module is to classify a given question according to the type of answer that it would expect.

The advantages of identifying the answer type for a given question are two-folds [33]. First, it can be used to determine the search strategy. For example, it can be used to form a template with the help of which the answer can be found. For instance, for the question "What is an atom?", if it is known that the answer-type is "*Definition*", the answer can be found using patterns like "atom is a...". Second, it can be used to limit the search space of the answer, in identifying useful resources from where to find the answer or determine the exact answer from a given paragraph. For example, for the question "Who is the president of India?", knowing the answer type to be "*Human*" can help limit the search for the answer to named entities of type "human".

Some examples of questions with their corresponding answer-types have been given in Example 1.2.

Example 1.2.	*Who was the prophet of the Muslim people?*	**Human**
	Where is the Kalahari desert?	**Location**
	How many Jews were executed in concentration camps during WWII?	**Number**

1.2 Motivation

1.2.1 Too Many Languages

There are a huge number of languages used by people in different parts of the world. The Indian subcontinent itself has more than 120 major languages out of which 29 languages are spoken by more than a million people each[1]. However, most of these languages are ill represented on the Internet, in terms of the amount of information available in these languages. Consequently, their representation in the NLP community is also limited. Most of the models, systems, resources and datasets have been developed only for English.

Table 1.1 gives a basic idea of the readership of different language newspapers in India. It can be observed that there is no English newspaper listed in the top five most read newspapers. And only two

[1] https://www.mapsofindia.com/culture/indian-languages.html

English newspapers have made their way into the list of top twenty read newspapers[2]. There is therefore a need to make data in different languages available and accessible to people. This makes it an important task to develop NLP systems and techniques for different languages.

Newspaper	Language	Readership (in millions)
Dainik Jagran	Hindi	16.631
Hindustan Dainik	Hindi	14.746
Dainik Bhaskar	Hindi	13.830
Malayala Manorama	Malayalam	8.803
Daily Thanthi	Tamil	8.283

Table 1.1: Readership (in millions) of various language newspapers in India.

Traditional Classification Systems

Traditionally, NLP systems used to be developed for specific tasks and languages. Researchers used carefully designed feature sets for their specific chosen tasks, which could be any of sentiment analysis, question classification, subjectivity analysis etc. These features were identified keeping their specific task under consideration, and the chances that the feature sets used in one task could be used as it is for another task, were low. The portability of feature sets across languages was also similarly low, as their extraction required the use of expensive to build linguistic resources like parsers. In some other cases, like sentiment analysis, resources specific to the task, such as sentiment lexicons were developed [25, 54]. Such resources, have to be developed separately for each language as well.

Although, these systems work very well for their specific tasks and languages, their task-specific or language-specific methodologies and dependence upon hand crafted features and expensive resources makes it difficult to adapt them to other tasks or languages. As seen earlier, there are a huge number of languages in the world which vary in their structure, morphology and various other properties and it is difficult and expensive to build resources and systems for every task in every language.

To avoid this tedious and expensive task of building end-to-end task-specific language-specific systems for every major language, we need to build systems which are generic and adaptable in nature.

Deep Learning Techniques for Generic Models

Recently, deep learning techniques have made it possible to develop end-to-end systems that are more generic in the set of tasks they can handle as they do not require hand-crafted features [22, 21]. These deep learning systems have been developed for various sentence classification tasks and rely majorly on the availability of large amount of labeled or unlabeled data. Such systems are therefore attractive when developing language independent systems.

[2] https://en.wikipedia.org/wiki/List_of_newspapers_in_India_by_readership

1.2.2 Morphologically Rich and Agglutinating Languages

A language is termed as a Morphologically Rich Language (MRL) in Computational Linguistics or Natural Language Processing when substantial grammatical information, that is, information about forming syntactic units or phrases is expressed at word level [55]. In other words, MRL are those languages where syntactical relations like subject, predicate, object etc. are indicated by making changes in the words itself, rather than with the help of relative positions, particles etc[3].

Agglutinating languages are MRLs where different morphemes join together in a word to bring in grammatical information, but the morphemes themselves retain their original form even after the unions[4].

Some examples from Hindi and Telugu have been given in Example 1.3 (borrowed from Singh and Sarma [50]) and Example 1.4. We have used the WX-notation[5] for transliterating the Indian language examples into English.

Example 1.3.

खा-त-ा *(KA-w-A)*	*eat-present participle-singular-masculine*
	eat-habitual-singular-masculine
खा-या *(KA-yA)*	*eat-perfective-singular-masculine*
	eat-past participle-singular-masculine
खाए-ग-ा *(KAe-g-A)*	*eat-future-3rd person-singular-masculine*

Example 1.4.

తిం-టా-డు *(wiM-t-Adu)*	*eat-future-3rd person-singular-masculine*
	eat-habitual-3rd person-singular-masculine
తిన్నా-ను *(wiM-nA-nu)*	*eat-perfective-1st person-singular-masculine*
తిం-దా-ము *(wiM-xA-mu)*	*eat-future-3rd person-plural*

These examples show how a single word contains various kinds of grammatical information. In particular, in Example 1.3 and Example 1.4, we show how a verb contains information regarding the gender, number, person, tense, aspect etc.

In addition, they give us an idea of the huge number of possible inflections of a single *root* word in a morphologically rich language. The result is that during classification the test data has a huge number of words that were not seen during training, which are also referred to as Out Of Vocabulary (OOV) words.

These examples suggest that words can not be treated as basic units in NLP techniques and it is essential to capture sub-word level information. This led us towards exploring the usage of character and syllable information for classification in morphologically rich languages such as Hindi and Telugu.

[3] https://www.quora.com/When-is-a-language-said-to-be-morphologically-rich/answer/Jose-Geraldo-Gouvea

[4] https://en.wikipedia.org/wiki/Agglutinative_language

[5] https://en.wikipedia.org/wiki/WX_notation

Zhang et al. [63] and Zhang and LeCun [62] use character level input in their respective works for classifying sentences. However, usage of syllables for classification has not been explored before. Thus we explore this aspect in the second part of our work.

1.3 Thesis Overview

In this work, we performed Sentence Classification in three languages - English, Hindi and Telugu, while considering two tasks - Sentiment Analysis and Question Classification. The three languages considered, differ in their linguistic structure and morphology. While, English is a fixed word order language, Hindi and Telugu are free word order languages. The morphological richness of the languages also increases from English to Hindi (an Indo-Aryan language) and to Telugu (a Dravidian language).

The goal was to identify techniques which are language and task independent, that is, techniques which can work well for all tasks and languages simultaneously, while also identifying possible novel extensions to existing ones.

For this, we first created a dataset for Hindi Question Classification. There are not many classification datasets available for Hindi and Telugu. Creating this dataset was thus essential for our goal of identifying and building task and language independent systems.

We then performed an extensive evaluation of the popular deep learning models on our datasets, with different parameters and inputs. We found that some models like Convolutional Neural Networks performed better than recurrent models for classification tasks. A second observation was that as a language became morphologically richer, character n-gram based input performed better than word-level input.

We then proposed an ensemble based CNN model, *multiInput-CNN*, that combined word and character based inputs to yield a model that was able to perform better than all our baselines in two out of three tasks in Hindi and Telugu, while giving comparable results for others.

In the next part, we proposed the usage of syllables, instead of characters to capture sub-word level information during classification. Syllable level input is more intuitive, as most of the morphemes in many languages can be seen as n-grams of syllables. In addition, it works to improve the performance of existing models by eliminating the noise created by character level input. We compared the performance of syllables against words and characters with the help of a CNN and the ensemble based *multiInput-CNN* model.

Finally, we experimented with attention mechanism in our *multiInput-CNN* model, so as to automatically find the best combination of inputs for every dataset. We experiment with a number of different attention based CNN models in this part and find that attention can be used to obtain a model that performs decently well across all datasets.

1.4 Our Contributions

- Creation of a dataset for Question Classification in Hindi. The dataset was prepared by manually translating the TREC-UIUC English dataset into Hindi.
- An evaluation of popular deep learning methods for Sentence Classification, while paying special attention to their performance on morphologically rich languages. Evaluations were done on English, Hindi and Telugu using five datasets for different types of inputs.
- Introduction of the idea of using syllable based input for sentence classification to capture sub-word level information efficiently in morphologically rich languages.
- *multiInput-CNN*, a ensemble-based CNN variant, that takes multiple types of inputs, such as word-level, character-level or syllable-level input for Sentence Classification.
- Experiments with attention mechanism in *multiInput-CNN* to automatically choose the best combination of inputs.

1.5 Thesis Organization

The rest of the thesis is organized as follows - Chapter 2 presents the related work done in Sentence Classification. In Chapter 3, we present the different datasets used for our evaluations, along with their statistics and snippets. In Chapter 4 we present the first part of our work which starts by describing the various models, input types and naming conventions and a description of the *multiInput-CNN* model. This is followed by the experiments and an analysis of their results. In the next chapter, Chapter 5, we present the second part of our work, the idea and motivation behind using syllables for classifying text, followed by experiments performed to show their usefulness. In Chapter 6 we show our experiments on using attention mechanism in the *mutliInput-CNN* model.

Chapter 2

Related Work

In this chapter, we list the relevant work done in Sentence Classification covering some of the most prominent works done in Sentiment Analysis and Question Classification, followed by a brief review of research in task-independent and language-independent sentence classification. Then we cover works in Hindi or Telugu Sentence Classification, and finally on syllables and their usage in text processing.

2.1 Task Specific Models

We start with a brief overview of some of the most prominent works in Sentiment Analysis and Question Classification.

2.1.1 Sentiment Analysis

One of the most prominent works in Sentiment Analysis has been that of Pang et al. [43]. They experimented with a number of features and classifiers in their work. For example, they experimented with different combinations of features which included the bag-of-words feature (unigrams and bigrams), part-of-speech (POS) tags, adjectives etc. For classification, they experimented with Naive Bayes, Maximum Entropy and Support Vector Machines (SVMs) and found that SVMs performed the best. Another prominent work is that of Pak and Paroubek [41], where they collect sentiment data from Twitter with the help of emoticons and experiment with Naive Bayes, SVMs and Conditional Random Fields (CRF) for classification and found that the Naive Bayes Classifier performed the best for bag-of-ngrams (unigrams, bigrams and trigrams) features.

Amongst other such works are Taboada et al. [54], who used a method based on a dictionary of words annotated according to their sentiment and strength for classification. Maas et al. [34] learned word vectors for sentiment analysis. There are several other prominent relevant works, we have mentioned just a few here for reference.

2.1.2 Question Classification

There are several works which use learning based approaches for question classification. The feature set in these works usually consists of the following three types: Lexical, which mainly includes bag-of-words, specific WH-word in the question [17], word shape [17], question length [2] etc., Syntactic features include POS-tags and head-words [17, 49], which were extracted using parse trees. And finally, Semantic features include the use of hypernyms (usually of the headword) [49, 17, 2, 30] and named entities [2]. Head words and related features, have been shown to contribute significantly towards the results [33]. Head words are usually extracted using rules on parse trees [49].

Cumbreras et al. [7] developed a classifier for Spanish by translating the Spanish questions into English first, and then using a classifier trained for English for the task. Raghavi et al. [46] used a similar approach for classifying code-mixed data. Solorio et al. [53] used some heuristics and the internet for extracting the features. Queries of the form "*Keyword* is a *Class i*", where "*Keyword*" is the optimized query is searched with each class "*Class i*". The number of results obtained was included as a feature. In another classifier built for Chinese, the most frequent words in each class were chosen as the keywords and they were extracted as features along with others like bag-of-words [32].

2.2 Task Independent Models

The use of deep learning techniques has helped researchers in building task-independent sentence classification models. Consequently, a lot of work has been done in task-independent Sentence Classification in the last few years using deep learning techniques. In this section, we present a brief review of the various works on task-independent Sentence Classification.

2.2.1 Convolutional Neural Network (CNN)

A particularly influential work in Sentence Classification has been [22], where a neural network with just one layer of convolution and with word vectors (random or word2vec [36] vectors) as input was shown to be effective on seven sentence classification tasks, out of which in four tasks, state-of the art was achieved.

In another work, a variant of CNN called Dynamic-CNN was proposed where the idea of *dynamic k-max pooling* was introduced [21]. A *dynamic k-max pooling* operation is where the k is decided as a function of the length of the sentence and the depth of the network.

2.2.2 Recurrent and Recursive Models

Since natural language text can simply be seen as a sequence of units - words, syllables or characters - recurrent and recursive models which have been shown to be suitable for sequential data have also been used. Works employing recursive or recurrent neural models for classification include the works

by Socher et al. [52], Komninos and Manandhar [24], Dong et al. [9], Irsoy and Cardie [18] and Hassan and Mahmood [14]. Hassan and Mahmood [14] apply Long Short-Term Memory (LSTM) network on the sequence of words in a sentence, which is a special form of Recurrent Neural Networks. Socher et al. [52], Dong et al. [9] and Irsoy and Cardie [18] exploit the phrase level information in their respective datasets and the tree structure of the sentence, to recursively find the sentiment of the entire sentence.

It should also be noted that a majority of the works mentioned in this subsection have focused on the task of Sentiment Analysis alone, rather than on Sentence Classification.

2.2.3 Modifying Basic Models

Many researchers have modified the basic convolution and recurrent models to improve upon their results.

Modifying LSTMs

Zhang et al. [61] make use of an LSTM-CNN network for a dependency sensitive model. They apply an LSTM on the sequential text data, and then apply a CNN on the LSTM outputs to get the final results. Li and Mak [28] use an LSTM language model to derive sentence representations and later use them for classification.

Komninos and Manandhar [24] depend on the usage of a dependency parser for sentence classification. They try to train word embeddings in a novel manner by using information from dependency graphs in addition to contextual information and use them for classifying sentences with the help of a number of different models like Support Vector Machines, Convolutional Neural Networks and Long Short Term Memory networks.

Modifying CNNs

Zhang et al. [65] make use of Active Learning (AL) for text classification in CNNs in order to choose the instances which may lead to maximum change in the word embeddings. This enables them to quickly learn task specific embeddings. Zhao et al. [66] bring in topic models for disambiguation of polysemic words when using word embeddings. Whereas, Li et al. [27] try to find a better initialization for the convolution filters. Conneau et al. [6] on the other hand, propose the use of very deep CNN for classification.

On a slightly different note, Gan et al. [11] use an encoder-decoder architecture, with a CNN for an encoder and an LSTM for a decoder to learn sentence representations and use the learned representations in classification tasks. And Johnson and Zhang [19] use Convolution Neural Networks to utilize word order information for classification. Rather than using word vectors as input, they directly apply convolution on text data.

Xia et al. [58], like Komninos and Manandhar [24] depend on the usage of a dependency parser for sentence classification. Xia et al. [58] make use of a bilingual dictionary and dependency parse trees for cross-lingual sentence classification using a Convolutional Neural Network. We attend to more works on cross-lingual sentence classification in Section 2.3.

Character Based Inputs in CNNs

Character based inputs for CNNs have also been used [63, 62, 57, 1]. Zhang et al. [63], Zhang and LeCun [62] and Becker et al. [1] use very deep CNNs to extract information from character level input. Wehrmann et al. [57] on the other hand, using a model similar to that of Kim [22], show that a single convolution layer with character embeddings as input is good enough for classification purposes. These works only use character 1-grams as input, we demonstrate later that character n-gram input can lead to much better performance in morphologically rich languages.

Multiple Inputs in CNNs

Proposing a yet another type of variance, Yin and Schütze [60] and Zhang et al. [64] modify the CNN model, for which they borrow ideas from Kalchbrenner et al. [21] and Kim [22], to account for multiple sets of word embeddings as input. While, Yin and Schütze [60] combine the embeddings at convolution stage, Zhang et al. [64] perform independent convolution operations for all sets, and combine them at the penultimate layer.

2.2.4 Attention Based Models

Attention based models have also been used for sentence classification. Hsu et al. [16] use a CNN-RNN hybrid attention based model for phrase aware classification. Yang et al. [59] use a hierarchical attention network to obtain sentence representation from word vectors, and then document representation from sentence representations. While Yang et al. [59] used attention on an LSTM network, Zhao and Wu [67] used attention on word vectors to obtain context vectors for each word. The context vectors were concatenated with word vectors which were used as input to the CNN model to capture long range dependencies in the sentence representation.

McCann et al. [35] modified the word vector inputs to add Context Vectors (CoVe) learnt using LSTM encoders trained on Machine Translation (MT) datasets. The MT system made use of attention mechanism while decoding to generate the output.

We elaborate more on attention based models in Chapter 6.

2.3 Cross-Lingual Sentence Classification

To the best of our knowledge, there has not been much work done in cross-lingual task-independent sentence classification. Most of the works in cross-lingual techniques are in sentiment analysis.

Some cross-lingual sentiment classification techniques, have been explored which use the resources in resource-rich languages for classification in resource-scarce languages. Deriu et al. [8] is able to use weakly supervised data for sentiment analysis. They used tweets for training their network, while the emoticons in them provided the supervision. Zhou et al. [68] use an attention based model similar to Yang et al. [59] for cross-lingual sentiment analysis.

Singhal and Bhattacharyya [51] translated their Hindi sentiment data word-to-word into English, and then augmented the training data with polarity words. Machine Translation was used in some part of the cross-lingual classification pipeline in several other works as well [56, 69, 4]. Although, this leads to the introduction of noise during classification as MT is still not perfect for many languages.

2.4 Sentence Classification in Hindi and Telugu

Few works have been done in Hindi and Telugu Sentiment analysis [51, 38]. Singhal and Bhattacharyya [51] perform sentiment analysis in Hindi, by first translating the Hindi data into English, and then augmenting their data with polarity words for classification. Mukku et al. [38] evaluate various non-deep learning approaches such as Naive Bayes, Logistic Regression, Support Vector Machines etc. for Telugu Sentiment Analysis. Mukku et al. [39] explore active learning for Telugu sentiment analysis.

2.5 Syllables

Syllables are commonly used for input representation in speech processing techniques. Here, we discuss some of the works in which syllables have been used and which motivated us to use syllables for sentence classification.

In Speech Processing

Ganapathiraju et al. [12] used a syllable level acoustic unit for speech recognition rather than a context-dependent phone unit. They showed that a system based on syllable level units instead of triphone units performed better. One of the reasons being that, they are lesser in number and more intuitive.

Schrumpf et al. [48] used syllables to accommodate for OOV words, especially in morphologically rich or agglutinating languages. They showed that syllable level transcripts perform better than word level transcripts for the task of spoken document retrieval.

Kishore and Black [23] and Kunchukuttan and Bhattacharyya [26] suggested in their work, that in most Indian languages, each written character in the native script is a syllable.

In Text Processing

In text processing, very little work has been done using syllables. Recently, Nguyen et al. [40] and Qun et al. [45] used syllables as the basic unit, for classifying text in languages which don't have clear word boundaries. For example, Vietnamese or Tibetan in these cases.

Apart from this, approximate syllables have been used for machine transliteration and machine translation in [10] and [26] respectively, where they have been shown to outperform other units like words, morphemes and character n-grams. In our work as well, we make use of approximate syllables for classification.

2.6 Summary

In this chapter, we try to present the diverse nature of works done in the task of sentence classification. Works have been done specifically for a single task like Sentiment Analysis or using deep learning, researchers have proposed models that are task-independent in their nature and do not depend on many linguistic resources for sentence classification. Researchers have used Convolutional Neural Networks, Recurrent Neural Networks and attention based models for the same. Researchers have also worked in the domain of cross-lingual sentence classification, where the aim is to use the linguistic resources in a resource rich language to classify sentences in a resource poor language. In Hindi and Telugu, researchers have mainly worked in the domain of Sentiment Analysis and the like such as hate speech detection, sarcasm or humour detection etc.

In our work, we extend the notion of task-independent sentence classification to task-independent and language-independent sentence classification. We found that a lot of work in the domain of cross-lingual sentence classification depends upon the usage of machine translation in some part of their model. But, since machine translation is still not perfect for many language pairs, our aim was to perform classification in the original language itself even for resource poor languages. Using the methods developed for English sentence classification, we tried to improve classification for the class of languages which are morphologically rich. For this, we explored various deep learning techniques and proposed the usage of sub-word level information for classification. We also proposed a novel classification model, called *multiInput-CNN* based on CNNs for the purpose. We then arrived at the idea of using syllable level input for classification.

Chapter 3

Datasets

For the purpose of our evaluations, we use five datasets. Three of which are for Sentiment Analysis, one each for English, Hindi and Telugu. The English dataset consists of only two classes (*positive* and *negative*), while the Hindi and Telugu datasets consist of three classes (*positive*, *negative* and *neutral*) each.

The other two datasets are for Question Classification in English and Hindi. The English dataset is the TREC-UIUC[1] dataset. The Hindi dataset has been created by the authors by translating the English TREC-UIUC dataset.

Specific details about each dataset have been mentioned in the following sub-sections.

3.1 Question Classification Datasets

3.1.1 TREC-UIUC Dataset (TREC-En)

This dataset was released by Li and Roth [29] along with an answer type taxonomy containing 6 core classes - *Abbreviation, Entity, Description, Human, Location* and *Numeric* - and 50 fine classes. That is, the taxonomy is hierarchical, with each core class containing a non-overlapping set of fine-classes. The dataset is divided into two sets of annotated questions, a training set with 5452 questions and a test set with 500 questions. The 6 core classes along with the number of instances in each have been listed in Table 3.1. A snippet of the dataset has been given in Figure 3.1. The answer-type label for a given question is followed by a space and the question. The label consists of two parts which are separated by a ':'. the first part is the core class label while the second is the finer label. We refer to this dataset as *TREC-En*.

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

```
DESC:manner How did serfdom develop in and then leave Russia ?
ENTY:cremat What films featured the character Popeye Doyle ?
HUM:title What is the oldest profession ?
DESC:def What are liver enzymes ?
HUM:ind Name the scar-faced bounty hunter of The Old West .
NUM:date When was Ozzy Osbourne born ?
HUM:ind Who killed Gandhi ?
HUM:ind Name 11 famous martyrs .
LOC:other What is the highest waterfall in the United States ?
ENTY:other Name a golf course in Myrtle Beach .
ENTY:religion In what religion was Isis the nature goddess ?
NUM:count How many people in the world speak French ?
```

Figure 3.1: Snippet of the English Question Classification Dataset (*TREC-En*). The answer-type label for a given question is followed by a space and the question itself.

3.1.2 TREC Hindi Dataset (TREC-Hi)

We created a new dataset for Question Classification in Hindi by translating the TREC-UIUC dataset described in Section 3.1.1. The TREC-UIUC dataset was first translated to Hindi with the help of Google Translate[2], and then was manually validated and corrected by the authors. A few sentences where the translations were not meaningful, were omitted from the train and test sets. The final train and test sets have 5444 and 499 questions respectively.

To validate the dataset, 6 Hindi native speakers were given 25 randomly selected sentences each, from the translated set, who rated them according to whether the sentences were completely incorrect (score 1) or completely correct and natural (score 5). We got an average score of 4.16 from our evaluations. Figure 3.2 shows the web interface used for the validation of the dataset. The dataset statistics have been mentioned in Table 3.1. Furthermore, Figure 3.3 shows a snippet of the dataset. We refer to this dataset as *TREC-Hi*. It has also been made publicly available to facilitate further research[3].

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

[2] https://translate.google.com

[3] https://github.com/tmadhuri/TREC-HindiData

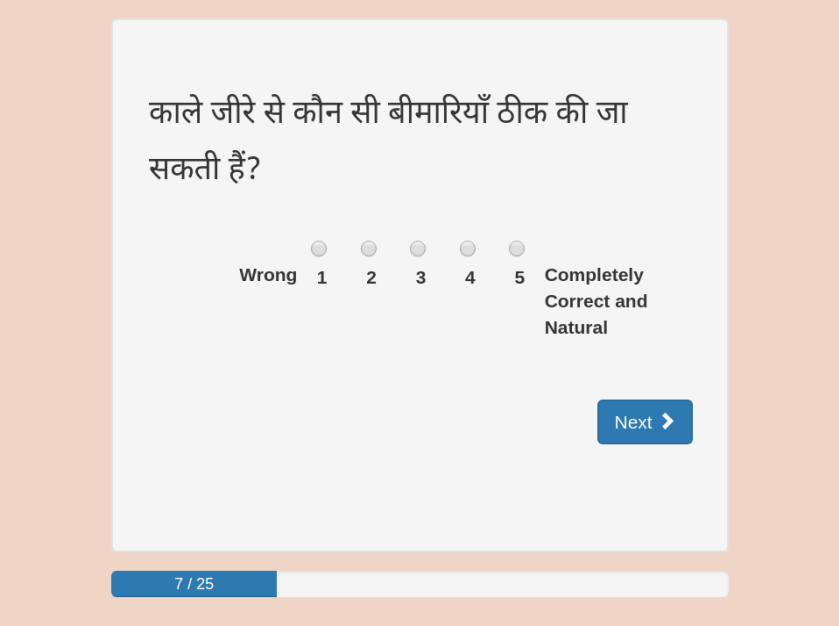

Figure 3.2: Web interface used for validation of Hindi Question Classification Dataset (*TREC-Hi*).

3.2 Sentiment Analysis Datasets

Table 3.2 contains statistics for all the sentiment analysis datasets, while their details have been listed in the following sub-sections. As none of the datasets contains a train-test split, we perform 10-fold cross validation to get results for these datasets.

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

Table 3.2: Statistics for Sentiment Analysis datasets. The table shows the number of sentences of each sentiment in different datasets.

3.2.1 Movie Reviews Dataset (MR)

This dataset contains 10662 movie reviews in English, and was released by Pang and Lee [42]. Each review contains only one sentence. The sentences are divided into two files. The first contains the reviews marked for *positive* sentiment, while the second contains the reviews marked for *negative* sentiment. A snippet of the dataset is given in Figure 3.4. We refer to this dataset as *MR*.

```
DESC:manner रूस में पहला दासत्व कैसे विकसित हुआ और फिर खतम हो गया?
ENTY:cremat किन फिल्मों ने पोपय डोयल के चरित्र को चित्रित किया था?
HUM:title सबसे पुराना पेशा कौनसा है?
DESC:def लीवर एंजाइम क्या हैं?
HUM:ind पुराने पस्चिम के ज़ख्मी चहरे वाले बाउंटी हंटर के नाम बताएँ।
NUM:date ओजी ऑजबॉर्न कब पैदा हुआ था?
HUM:ind गांधी को किसने मारा था?
HUM:ind ११ प्रसिद्ध शहीदों के नाम बताएँ।
LOC:other संयुक्त राज्य अमेरिका में सबसे ऊँचा झरना कौन सा है?
ENTY:other मिर्टल बीच में एक गोल्फ कोर्स का नाम बताएँ।
ENTY:religion किस धर्म में आईसिस प्रकृति देवी थी?
NUM:count दुनिया में कितने लोग फ्रेंच बोलते हैं?
```

Figure 3.3: Snippet of the Hindi Question Classification Dataset (*TREC-Hi*). As in *TREC-En*, here too the answer-type label for a given question is followed by a space and the question.

3.2.2 Sentiment Analysis - Hindi (Senti-Hi)

This dataset[4] also contains movie reviews and has three classes: *positive*, *negative* and *neutral*. There are a total of 6580 sentences. A snippet of the dataset has been given in Figure 3.5. In the figure, "*p*" stands for positive reviews, "*n*" stands for negative reviews and "*o*" stands for neutral reviews. We refer to this dataset as *Senti-Hi*.

3.2.3 Sentiment Analysis - Telugu (Senti-Te)

The Telugu sentiment analysis dataset contains 5409 sentences annotated for *positive*, *negative* or *neutral* sentiment. It was released by Mukku and Mamidi [37]. A snippet of the dataset has been given in Figure 3.6. In the figure, "*+1*" stands for positive reviews, "*-1*" stands for negative reviews and "*0*" stands for neutral reviews. We refer to this dataset as *Senti-Te*.

3.3 Conclusion

In this chapter, we presented the various datasets we used for our experiments in sentence classification. For the purpose of our evaluation, we have even created one dataset, *TREC-Hi* which we showed in Section 3.1.2. We used these five datasets, in two tasks - Question Classification and Sentiment Analysis and three languages - English, Hindi and Telugu, for our work shown in Chapter 4, Chapter 5 and Chapter 6.

[4]This dataset is yet to be published.

```
simplistic , silly and tedious .
it's so laddish and juvenile , only teenage boys could
possibly find it funny .
exploitative and largely devoid of the depth or
sophistication that would make watching such a graphic
treatment of the crimes bearable .
not so much farcical as sour .
interesting , but not compelling .
```

(a) Negative Reviews.

```
scores a few points for doing what it does with a
dedicated and good-hearted professionalism .
occasionally melodramatic , it's also extremely
effective .
spiderman rocks
an engaging overview of johnson's eccentric career .
in its ragged , cheap and unassuming way , the movie
works .
```

(b) Positive Reviews.

Figure 3.4: Snippet of the English Sentiment Analysis Dataset (*MR*). The sentences are split into two files, one with positive reviews and the other with negative reviews.

```
माही गिल ने भी अपने किरदार को इमानदारी से निभाया है। p
यही सीक्रेट साहिर की ताकत भी है और कमजोरी भी। o
फिर भी बाल कलाकार एहसास ने हर बार की तरह बेहेतरीन एक्टिंग की है। p
फिल्म की कहानी थोड़ी कसी हुई होनी चाहिए जिसकी कमी साफ साफ भाग मिल्खा
भाग में देखने को मिली। n
हेट स्टोरी की सफलता ने फिल्म के निर्देशक को फिल्म का सीक्वल बनाने को
प्रेरित किया। p
संगीत संगीत के मामले में भी हेट स्टोरी काफी बुरी है। n
यह विद्याधर और महक की एक सिम्पल सी लव स्टोरी है। o
अभिनय रणविजय सिंह ने फिल्म में अच्छा काम किया है। p
हालांकि फिल्म की घिटी पिटी कहानी के चलते फिल्म का बॉक्स ऑफिस रिजल्ट
कुछ खास नज़र नहीं आ रहा है। n
रज्जो के किरदार में माधुरी बहुत ही खूबसूरत और बेहतरीन नज़र आई हैं। p
कौन किसको बेवकूफ बनाता है ये ही पूरी फिल्म का सस्पेंस है। o
फिल्म के स्पेशल इफेक्ट अच्छे हैं। p
```

Figure 3.5: Snippet of the Hindi Sentiment Analysis Dataset (*Senti-Hi*). Each sentence is separated from its sentiment label by a *tab* character.

+1 వాహనంలో ఉన్న పోలీసు సిబ్బందికి బుల్లెట్ ప్రూఫ్ కవచాలు లేవన్నది అటుంచితే, వారు ఏమాత్రం అప్రమత్తంగా లేరన్నది కూడా వాస్తవం.
-1 దేశంపట్ల కానీ, ప్రజలపట్ల కానీ బాధ్యత కనిపించదు.
0 ఒక పనిమనిషి కుమారుడు ఈ దేశ ప్రధాని కావడం అంబేద్కర్ కారణంగానే సాధ్యమైందని చెప్పుకొచ్చారు.
0 మత ప్రాతిపదికన మనుషుల్ని రెచ్చగొట్టే ఉగ్రవాదం ఏదైనా అది ప్రపంచానికి ప్రమాదకారి అనే విషయాన్ని అందరూ గుర్తించాలి.
+1 జిల్లాలో గోదావరి పుష్కరాలు నిర్వహించేందుకు ముందస్తు బందోబస్తు ప్రణాళిక సిద్ధం చేయాలని రాష్ట్ర పోలీసు డీజీపీ అనురాగ్శర్మ జిల్లా పోలీసు అధికారులను ఆదేశించారు.
-1 దీంతో తాన్యానాయక్తాండాలో విషాదం నెలకొంది.
0 అయితే పన్ను ఎంత ఉండాలనే అంశాన్ని కేంద్ర ఆర్థిక మంత్రి, ఆర్థిక శాఖ కార్యదర్శి, రాష్ట్రాల ఆర్థిక మంత్రులతో కూడిన కమిటీ నిర్ణయిస్తుంది.
+1 ఈ వ్యవస్థ కేవలం నేలకు పరిమితం కాకుండా, నింగికీ నీటికీ కూడా విస్తరించినందున, విమానాలతో పాటు, పర్వతారోహకులకూ పనికివస్తుంది.
+1 యావత్ ప్రపంచమూ ఉగ్రవాదంపై కలసికట్టుగా పోరాడనిపక్షంలో వినాశనం తప్పదని టర్కీ అధ్యక్షుడు ఎర్డగాన్ హెచ్చరిస్తున్నారు.
+1 తెలంగాణలోని అన్ని పట్టణాలను అభివృద్ధి చేయాలని ప్రభుత్వం కృత నిశ్చయంతో ఉందన్నారు.
-1 పాతికేళ్ళ నిరీక్షణ అనంతరం వెలువడిన ఈ తీర్పు బాధితుల కుటుంబీకుల మనసులకు కాస్తంత ఉపశమనాన్ని కలిగిస్తుందేమో కాని, చెదిరిన వారి జీవితాల్లో మార్పు తీసుకురాలేదు.
0 బ్రిటన్ తో చారిత్రక, వాణిజ్య సంబంధాలున్న దాదాపు ప్రతిదేశమూ అక్కడి ఓటర్లకు హితవు చెబుతూనే ఉన్నది.

Figure 3.6: Snippet of the Telugu Sentiment Analysis Dataset (*Senti-Te*). For each sentence, its sentiment label is separated with spaces from the sentence.

Chapter 4

Baseline Experiments and Our MultiInput-CNN Model

In this chapter, we first introduce all the models we use in our experiments, along with the different methods by which we represent a sentence. We then report our baseline evaluations, which is followed by our work on the *multiInput-CNN* model.

4.1 Methodology

4.1.1 Input

Any given sentence which needs to be classified is broken down into a sequence of units. The units could be words, characters or syllables. We elaborate on syllable level representation in Chapter 5 and focus on word and character level representation alone here.

Each unit is represented by a fixed length vector. Thus, we get a 2-dimensional representation for a sentence. That is, a sentence s with n units is represented by a 2-dimensional matrix of size $n \times d$. Here, d is the dimension of the vectors with which each unit is represented.

4.1.1.1 Sequence of Words

The first and the most basic manner in which a sentence is represented is at the word level. In our case, we represent a sentence as a sequence of words. An example is given in Example 4.1.

Example 4.1.	*the film was good .*	**Original Sentence**
	[the] [film] [was] [good] [.]	**Sentence Split**

Each word in turn, is represented by a vector of fixed dimension d. The vector might be a random vector or a word2vec vector.

Random Vector (word-rand)

In the random vector representation, the vector representation for each word is sampled from a uniform distribution. We refer to this type of input as *word-rand*.

Word2Vec Vector (word-word2vec)

Word2Vec word embeddings have been used extensively in the NLP community to initialize word embeddings for their respective tasks. This model for training word vectors was proposed by Mikolov et al. [36] where in, they make use of contextual information for a given word (that is, its neighbours) from a large text corpora to finds its embedding. The result is that contextually similar words are closer together in the vector space as compared to dissimilar ones.

In our case, we obtain the pre-trained word2vec vectors by training the word2vec model on English, Hindi and Telugu Wikipedia dumps[1] respectively for the three languages. The dumps have a size of *57 GB*, *978 MB* and *1 GB* respectively, with *4398843*, *303691* and *446992* unique words respectively. The dimension of the word vectors, d, was fixed at 300 as given by Mikolov et al. [36]. The training was done with the help of the *Gensim* library[2] [47]. The resulting word2vec word embeddings were used to initialize the word vectors in our neural network models. We refer to this type of embedding as *word-word2vec*.

4.1.1.2 Sequence of characters (char-1-gram)

In this case, a sentence is represented as a sequence of characters. Each character is then represented by a vector of fixed dimension d. Example 4.2 shows how a sentence can be seen as a sequence of characters.

Example 4.2.

the film was good .	**Original Sentence**
[t] [h] [e] [f] [i] [l] [m] [w] [a] [s] [g] [o] [o] [d] [.]	**Sentence Split**

4.1.1.3 Sequence of character n-grams (char-ngram)

Here, a sentence is represented as a sequence of character n-grams which do not span across words. Each character n-gram is then represented by a random vector of fixed dimension d.

The idea behind using character n-grams is that they can help us represent the sentence as a sequence of morphemes, which are the smallest meaningful unit in a language. This, in turn can be beneficial for morphologically rich languages, where each word contains substantial information often due to the number of morphemes present in it. In the set of languages chosen by us, the morphological richness

[1] `https://dumps.wikimedia.org/`
[2] `https://radimrehurek.com/gensim/models/word2vec.html`

increases from English to Hindi and to Telugu. The sequence of characters input described previously, is a special case where $n = 1$.

Example 4.3 shows an example of how a sentence can be split into a sequence of character n-grams, where $n = 3$.

Example 4.3.	*the film was good .*	**Original Sentence**
	[the] [fil] [ilm] [was] [goo] [ood] [.]	**Sentence Split**

This type of input is referred to as *char-ngram* henceforth.

4.1.2 Support Vector Machines

For our experiments, we considered a non-deep learning model, in addition to the deep learning based models described in Section 4.1.3, Section 4.1.4 and Section 4.1.5.

Support Vector Machine (SVM) is a type of supervised learning model. An SVM model is generally a linear classifier. But, it can be used for non-linear classification as well with the help of kernel trick[3]. Given sufficient data marked with its target class, an SVM model can be trained to classify a new data point.

Model

We trained an SVM model with a linear and an RBF kernel using the *Scikit*[4] library [44]. Since the RBF kernel always gave very poor results, we do not mention it's results here. The features used were - bag-of-words, bag-of-ngrams of words and bag-of-ngrams of characters. We also feed a combination of words and character n-grams to the SVM model for our experiments.

4.1.3 Convolutional Neural Network (CNN)

Convolutional Neural Network is a widely used type of deep neural network. For different tasks in text processing, 1-dimensional CNNs are generally used, where a convolution operation is applied over a sentence. A convolution operation consists of a set of one or more filters of same or different window sizes (or widths). The filters slide over different windows of inputs, thus giving one output value for each n-gram of the input sentence. A local or global pooling layer is generally applied on these outputs. In our experiments here, we use a global max-pooling layer which retains the maximum value out of all the outputs given by a filter.

As we can see, each filter in a CNN captures local information about different windows, independent of other parts of the sentence. Multiple such filters are thus applied to capture different kinds of information present in the sentence which is essential for performing the required task. Secondly, since

[3] https://en.wikipedia.org/wiki/Kernel_method
[4] http://scikit-learn.org/stable/index.html

a single filter applies on different parts of the sentence, it can easily capture the required information no matter where it is present in the sentence. Thus, CNNs can be extremely helpful when working with languages which do not have a fixed word order.

Model

We borrow our CNN architecture from [22]. Kim [22] defines a number of different CNN architectures with minor variations. We show the basic architecture in Figure 4.1. 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 [22], 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.

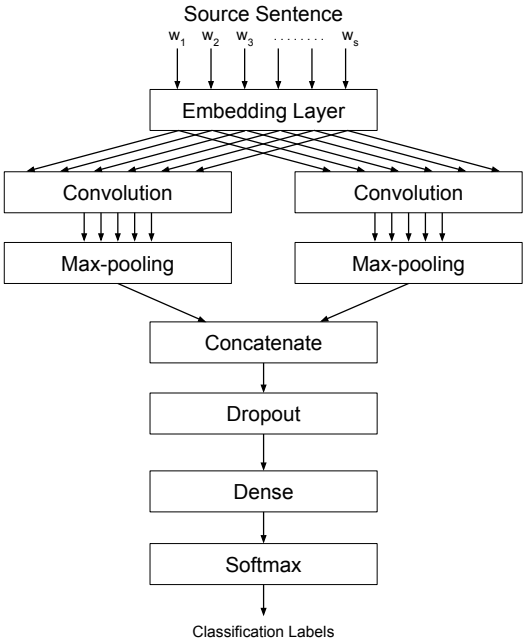

Figure 4.1: CNN Architecture as proposed by Kim [22].

Of the multiple variations defined by Kim [22], we show our results on *CNN-rand* and *CNN-non-static*. In *CNN-rand*, the word embeddings are initialized randomly, while in *CNN-non-static* they are initialized with word2vec word embeddings. In both the cases the word embeddings are modified while training. We also did our experiments with *CNN-static* where the word embeddings are initialized with word2vec word embeddings as in the case of *CNN-non-static* but are kept static throughout the training

process. Since, *CNN-non-static* always performed better than *CNN-static* in our experiments, we do not mention those results here. The model was implemented with the help of *Keras*[5] [5].

4.1.4 LSTM and Bi-LSTM

Recurrent Neural Network (RNNs) is a neural network model which has the ability to process temporal data by recursing over a sequence of arbitrary length. This makes it a suitable choice for processing textual data which do not have a fixed size.

The Long Short Term Memory (LSTM) architecture was introduced by Hochreiter and Schmidhuber [15]. LSTMs are a special case of RNNs which with the help of special - *input*, *forget* and *output* - gates can selectively retain the desired information in long sequences of inputs. Thus, being helpful in processing long sentences by capturing its most essential parts.

The Bidirectional LSTM (Bi-LSTM) architecture is a variant of the LSTM architecture and was introduced by Graves et al. [13]. A Bi-LSTM consists of two LSTMs. In the first, the input is processed from beginning of the sequence to the end, whereas in the other it is processed backward - from the end to the beginning of the sequence. The outputs from both the networks is concatenated to give the final output.

Model

For our experiments, we apply a vanilla LSTM or Bi-LSTM (implemented in Keras [5]) over different representation of sentences as described in Section 4.1.1. The outputs from the LSTM or Bi-LSTM are fed to a fully connected layer, with *softmax* activation and trained using *categorical-crossentropy* loss for classification.

4.1.5 Our multiInput-CNN variant

Motivation

Kim [22] had also proposed a multichannel variant of their basic CNN architecture called *CNN-multichannel*, where there were two input channels, both initialized with pre-trained word vectors. The same convolution filters were applied on both the channels. While the weights in one of the channels was updated, the other was kept static. Yin and Schütze [60] used a similar idea of multichannel input, but initialized each channel with a different set of pre-trained word embeddings. On the other hand, Zhang et al. [64] also used multiple pre-trained word embeddings as inputs, but the convolution filters were applied independently to all the inputs, and the resultant output was joined at the penultimate layer. Applying an independent convolution on each input gives the freedom of finding and using the best filter dimensions and other parameters for each input.

[5] `https://keras.io/`

Combining different inputs together helps the model to leverage the different types of information present in them. Researchers have tried to combine different types of word embeddings together, that is, all the inputs they combine together belong to the word level, and can thus provide information that is limited to that level. However, there hasn't been any work combining word level information with sub-word level information in sentence classification. Thus, in our work here, we propose a model that seeks to combine word and sub-word level information. Another way in which our model is different from the previous attempts is that usually, when different inputs are combined together, it is done by concatenating them at the penultimate layer. Whereas, we simply take an average of the final layer outputs produced by different inputs.

Our Model

We propose the use of both word and character based inputs in a CNN based model. The inputs could be word level, character level or syllable level. We mention our experiments with syllable level input in Chapter 5.

In this chapter, we describe our model which combines two types of inputs together, word and character level. Specifically, the inputs are - sequence of character n-grams (*char-ngram*) and sequence of words with word2vec vector (*word-word2vec*) initialization. These inputs were chosen on the basis of their performance in the baseline experiments (described in Section 4.2).

We apply the complete CNN architecture as described in *CNN-rand* and *CNN-non-static* in Kim [22] on both the inputs respectively. We then take an average of the outputs of the final (*softmax*) layers of the individual networks to get our final output. This network is also trained with a *categorical-crossentropy* loss. The architecture is shown in Figure 4.2. 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.

This network, in our opinion is language independent, as it is able to leverage the morphological information with the help of character n-grams that is beneficial for morphologically rich languages, while the information captured in the rich word embeddings is also retained. This is further confirmed by our experiments.

4.2 Experiments and Results

The experiments have been divided into three parts. Section 4.2.1 presents detailed evaluation and discussion of different baseline models and inputs on all datasets. Section 4.2.2 compares our *multiInput-CNN* model's performance with the baselines and other state-of-the art results. In Section 4.2.3, we try to understand the inherent dataset biases across languages.

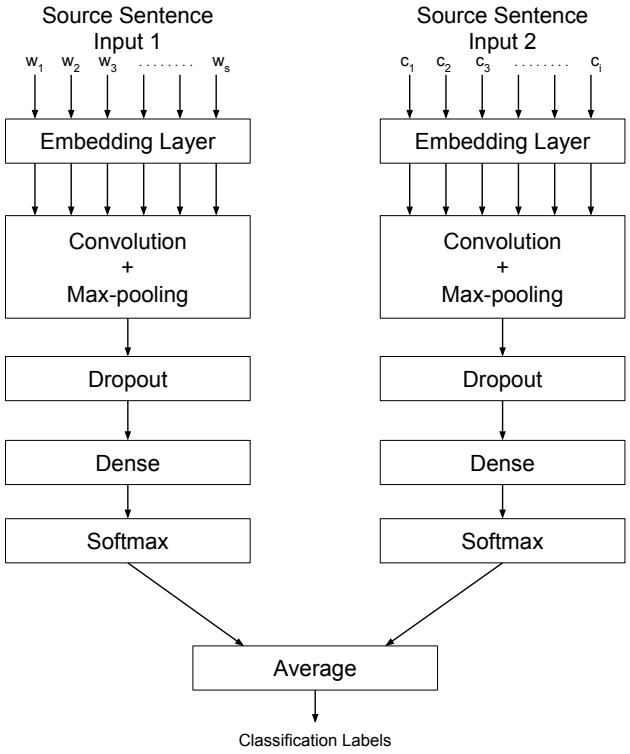

Figure 4.2: Figure shows the proposed *multiInput-CNN* architecture.

4.2.1 Evaluation of Baseline Models

For the purpose of evaluating our baseline models, which are - SVMs, CNNs, LSTMs and Bi-LSTMs - described in Section 4.1, for different inputs, we conducted a number of experiments. Parameter tuning was done for each model and each dataset, and the results have been presented in Table 4.1.

Performance of Character 1-grams

As we can see from Table 4.1, character 1-grams input in CNN performs consistently bad for all the datasets. A possible reason could be that our network is not deep enough to be able to process the character 1-grams and draw useful information from them. Even though Wehrmann et al. [57] showed that a single layer of convolution was enough for classifying sentences with *character-1-gram* input, this does not seem to be true in our case.

LSTM and Bi-LSTM for Classification

Sequential models like RNNs, LSTMs and Bi-LSTMs seem to be a more intuitive option, as compared to other models like CNNs, when dealing with textual data due to their inherent sequential nature. However, we notice from our results that LSTMs and Bi-LSTMs perform bad for all languages and all

Model	Input	MR	Senti-Hi	Senti-Te	TREC-En	TREC-Hi
CNN	word-rand	76.35	70.44	52.18	92.20	86.97
	word-word2vec	**81.04**	**73.67**	55.79	**94.00**	**93.19**
	char-1-gram	66.38	67.51	51.70	76.2	75.95
	char-ngram	77.40	72.06	**58.05**	89.20	88.18
LSTM	word-rand	**75.41**	**56.83**	45.82	**84.00**	**81.76**
	word-word2vec	71.68	51.81	45.79	82.60	81.56
	char-ngram	74.63	51.85	**45.83**	83.40	78.76
Bi-LSTM	word-rand	**75.19**	**68.65**	**52.30**	**88.80**	**84.17**
	word-word2vec	75.14	68.14	50.78	86.80	**84.17**
	char-ngram	74.91	68.18	50.99	82.20	82.57
SVM	word n-grams	75.2	71.00	53.95	90.00	87.17
	char n-grams	77.07	70.91	57.00	**91.20**	**87.58**
	(word+char) n-grams	**77.45**	**71.28**	**57.18**	**91.20**	87.17

Table 4.1: Evaluation of baseline models.

classification tasks. Convolutional Neural Networks seem to be much more efficient for classification tasks.

Character n-grams for Classification

It is also essential to note that as we move towards morphologically richer languages, it is the character n-gram input that leads to better results. When using a CNN, character n-gram input initialized using random vectors outperforms word level input also initialized using random vectors for all the Indian language datasets. Even for *MR*, character input performs better, although the gap is much smaller. In *Senti-Te*, the result obtained with character n-grams is even better than that obtained using word2vec vectors to initialize word embeddings. This justifies our first statement of character input performing better than words for morphologically rich languages.

Even in SVMs, character level features seem to outperform word level features, and the gap is the highest for Telugu. It can be reiterated here, that of the three languages chosen by us, the morphological richness increases from English, to Hindi and to Telugu.

This trend does not work for LSTMs and Bi-LSTMs. While CNNs and SVMs might be classifying the sentences by identifying the most salient n-grams from the input, LSTMs and Bi-LSTMs, it seems, are not able to capture a similar information, as they work by finding a compositional representation of the entire sentence. LSTMs and Bi-LSTMs tend to determine the importance of each input unit in context to the rest of the sentence (which is generally determined by the hidden state of the LSTM cell), CNNs on the other hand, treat each local region (whose size is equal to the filter width operating on it) independently.

Another possible reason might be, that while we break a sentence into its constituent character n-grams, there is a huge amount of noise that is introduced. Consider, for example, the word and its

constituent n-grams with $n = 2$ as given in Example 4.4, *[खा] ([KA])* and *[इए] ([ie])*, here, are n-grams that may help contribute towards the capturing of morphological information by the classifiers, while *[◌ाइ] ([Ai])* may not be important and thus lead to noise. CNNs might be able to ignore such noise better as compared to LSTMs and Bi-LSTMs.

Example 4.4.	खाइए *(KAie)*			**Word**
	[खा] ([KA])	*[◌ाइ] ([Ai])*	*[इए] ([ie])*	**N-Gram Split**

Insights on CNNs Performance

Since our CNN model has just one layer of convolution, and used *global max pooling* instead of *local max pooling*, the model essentially picks the most salient n-grams from the input. This idea is confirmed from the results where, the SVM results for all languages, with word level input (*word n-grams*) can be seen to be roughly equivalent to the ones obtained with CNN model also with word level input (*word-rand*). Which also means, that one of the greatest benefits of using a CNN lies in the ability to leverage the information in pre-trained word embeddings like word2vec.

The results obtained from SVM when using both word and character level n-grams as input, and the results obtained with character n-grams as input in CNNs lead us to our final model, *multiInput-CNN*, where we combine character n-grams and pre-trained word embedding inputs.

4.2.2 multiInput-CNN

We implemented the model as described in Section 4.1.5. This model combines character n-grams and pre-trained word embedding inputs. While the character level inputs capture important information for morphologically rich languages, pre-trained word embedding inputs enable the model to capture important contextual information. The results of this model in comparison with some of the state-of-the art result have been shown in Table 4.2.

As we can see, the *multiInput-CNN* model is able to achieve better results than all our baselines for 2 out of 3 datasets in Hindi and Telugu, while obtaining results comparable to the state-of-the arts for English. Thus, we can say that with our *multiInput-CNN* model, we are able to successfully combine the information present in character n-grams (morphological information) with the contextual information present in word2vec word vectors.

4.2.3 Experiments with Translated Datasets

To understand possible biases created due to different datasets, we repeated our baseline experiments with CNNs on datasets obtained by automatically translating each sentiment analysis dataset into the other two languages with the help of Google Translate. The results showed that in general, the translated data always obtained a lower accuracy than the original data. This may have two implications, first that

Model	MR	Sent-Hi	Sent-Te	TREC-En	TREC-Hi
multiInput-CNN	81.03	**74.51**	**58.86**	93.96	92.22
CNN word-word2vec (similar to *CNN-non-static* [22])	80.21	73.86	55.50	93.24	**93.71**
CNN char-ngram	77.43	72.73	57.97	87.80	85.35
Kim [22]	81.50	-	-	92.80	-
Gan et al. [11]	77.77	-	-	92.60	-
Li et al. [27]	82.10	-	-	94.40	-
Zhao et al. [66]	**83.00**	-	-	84.10	-
Zhang et al. [61]	82.20	-	-	85.60	-
Zhang et al. [64]	-	-	-	95.52	-
McCann et al. [35]	-	-	-	**95.80**	-

Table 4.2: Comparison of *multiInput-CNN* with the state-of-the arts. Each result has been reported by taking an average of 5 runs for *multiInput-CNN* and an average of 10 runs for others.

translating the data into another language like English and then classifying in that language can induce more errors.

Second, these results also help us establish that the Hindi and Telugu sentiment analysis datasets, are inherently much difficult as compared to the English dataset, and the difference in accuracies between the datasets may not be a reflection of difficulty of classification in that particular language only. This is especially true for Telugu Sentiment Analysis (*Senti-Te*), for which the classification result are much smaller than that for Hindi or English.

Thus, throughout our paper, we have only focused on the trends in the performance of different models within a single dataset. These results have been listed in Table 4.3

Data	Original	English	Hindi	Telugu
MR	81.04	-	73.20	71.21
Senti-Hi	73.23	71.65	-	69.52
Senti-Te	58.05	51.57	51.54	-

Table 4.3: CNN Results on Translated datasets

4.3 Conclusion

Through our experiments with different deep learning models with different types of inputs, we showed that as we move to morphologically rich languages, character n-gram based inputs lead to better performance than word based input for most of the models. This conclusion also enabled us to present a model, *multiInput-CNN* that capitalizes on both word and character based inputs to give us a language independent model. While the character based embeddings provide morphological information to the model, pre-trained word embeddings bring in knowledge from a larger monolingual corpus.

Chapter 5

Syllable Based Input for Sentence Classification

In this chapter, we describe in detail the second part of our work. We introduce the idea of using syllables for sentence classification, starting with the motivation behind using it. Then we state the manner in which syllables were extracted from the text. Finally, we show, with the help of experiments on Convolution Neural Networks and the *multiInput-CNN* model, the benefit of using syllables over character n-grams for capturing morphological information.

5.1 Motivation

For Morphologically Rich Languages

We had concluded in Section 4.2 that the use of character n-grams can lead to better results, particularly for the morphologically rich languages - Telugu and Hindi. As described in Section 1.2.2, 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. For classifying such languages, it is essential to move beyond word level, and capture sub-word level information.

Morphemes as N-Grams of Syllables

In many languages, such as in most Indian languages, morphemes can be seen as n-grams of syllables. For example, in the Hindi verbs खाकर (KAkara) or जाकर (jAkara) the roots खा (KA) or जा (jA) form a syllable while the common suffix -कर (-kara) (meaning 'after') is an n-gram of two syllables - क (ka) and र (ra). In English, the word *beautiful* breaking into its syllables *beau-ti-ful* is a similar example.

Character N-Grams Leading to Noise

We saw in Example 4.4 that splitting a word into character n-grams can lead to noise. Since the character n-grams we consider are 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.

Also, in most Indian languages, each written character in the native script is a syllable [23]. Although, when written in Unicode, it is possible that a single such character is broken down into two or more Unicode characters, as in the case of the character 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 n-grams as input, we may have n-grams where the two characters क (k) and ी (I) are split into two different n-grams. This, again, leads to noise. To avoid this, it is intuitive to use syllables as the basic unit for classification. Syllables too, like character n-grams, can be useful in dealing with Out Of Vocabulary (OOV) words with the added advantage that they would be lesser in number than character n-grams.

Morphemes can be of Different Lengths

Researchers have tried to capture morpheme level information by using character n-grams, where n is 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 n-grams would not be able to successfully capture all the morphemes, but only those whose length is equal to the chosen n-gram size.

A simple example has been given in Example 5.1 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.

Example 5.1.

खाएगा *(KAegA)*	**Word**
[खा] ([KA]) *[ाए] ([Ae])* *[एग] ([eg])* *[गा] ([gA])*	**N-Gram Split**
[खा] ([KA]) *[ए] ([e])* *[गा] ([gA])*	**Meaningful Split**

5.2 Methodology

In this section, we describe the manner by which we extract the syllables from text. Once each word is broken down into its constituent syllables, we can express the given text as a sequence of syllables. Then each syllable can be represented with a random vector of fixed dimension d. These can then be used as input to our neural network models as in the earlier cases. We refer to this type of input as *syl-ngram*. We evaluate the proposed syllable based input using the CNN and *multiInput-CNN* models described in Section 4.1.3 and Section 4.1.5. It should be noted here that while syllabification is difficult, what we extract here are approximate syllables.

5.2.1 Syllable Extraction for English

For English, syllabification was done using the python library *PyHyphen*[1]. Given a word as input, this library can be used to break it into its syllables.

The library works on the C library *libhyphen*[2], which in turn is based on the Knuth-Liang Algorithm [31] for hyphenation. Liang [31] base their hyphenation algorithm or the hyphenation rules on those used for syllabification.

5.2.2 Syllable Extraction for Hindi and Telugu

For Hindi, we performed syllabification using the *LibIndic Syllabifier*[3] module of the *LibIndic*[4] library. This library currently performs syllabification for the Indian languages Malayalam, Kannada, Bengali, Tamil and Hindi.

The library uses a rule based method to break a word into its constituent syllables. It requires a list of special symbols to be specified for each language or script. First, it requires a list of *matras*, which refer to the vowels-signs or diacritics used to express the vowels. Some examples in Hindi are '◌ा', '◌ी', '◌ो', '◌ु' etc. Second, it requires a list of *limiters*, which refer to the punctuation marks used. Examples include '.', '"', '*!*', '*;*', '*,*', '*?*'. The third symbol that is needed to be specified is the *virama*[5] or the *halant* symbol, which is a diacritic used to suppress the inherent vowel present in a consonant. In Hindi, it is the '◌्' symbol.

Given this list of *matras*, *limiters* and the *virama* (or *halant*) symbol 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 implement the code for Telugu syllabification using these rules. The required lists of *matras*, *limiters* and the *virama* symbol were obtained using the Unicode chart for Telugu[6].

[1] https://pypi.org/project/PyHyphen
[2] https://github.com/hunspell/hyphen
[3] https://github.com/libindic/syllabalizer
[4] https://libindic.org
[5] https://en.wikipedia.org/wiki/Virama
[6] https://unicode.org/charts/PDF/U0C00.pdf

5.3 Experiments and Results

5.3.1 Experiments with CNN

To evaluate the performance of syllables in sentence classification, we first conducted experiments with the *CNN-rand* architecture described in Section 4.1.3. We compare word (*word-rand*), character n-gram (*char-ngram*) and syllable n-gram (*syl-ngram*) inputs for classification in Table 5.1.

In all the experiments, the model parameters including the sizes of character n-grams and syllable n-grams were found through parameter tuning. For syllables, we chose the best out of syllable unigrams, bigrams and trigrams. All the results have been reported by taking an average of 5 runs for models based on *multiInput-CNN* and an average of 10 runs for others.

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

Table 5.1: Performance of different inputs on *CNN-rand*. **Highlighted** are the best results for each dataset.

Observations

As can be observed in Table 5.1, 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, as compared to word based input shows the importance of capturing sub-word level information for morphologically rich languages. This also confirms that syllables can 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 n-grams, at the same time capturing a larger number of meaningful morphological units, as was hypothesized in Section 5.1.

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.

5.3.2 Experiments with MultiInput-CNN

Next, we conducted experiments with the *multiInput-CNN* model mentioned in Section 4.1.5. We experimented with multiple combinations of different input types: words (with word2vec initialization) (*word-word2vec*), character n-gram (*char-ngram*) and syllable n-gram (*syl-ngram*).

Model-Input	MR	Senti -Hi	Senti -Te	TREC -En	TREC -Hi
word-word2vec + char-ngram	81.03	74.51	58.86	93.96	92.22
syl-ngram + char-ngram	78.24	73.36	58.31	90.60	86.17
word-word2vec + syl-ngram	80.66	**74.71**	**59.21**	93.68	92.79
word-word2vec + syl-ngram + char-ngram	80.15	73.92	58.91	92.80	89.94
CNN word-word2vec (similar to *CNN-non-static* [22])	80.21	73.86	55.50	93.24	**93.71**
Kim [22]	81.50	-	-	92.80	-
Gan et al. [11]	77.77	-	-	92.60	-
Li et al. [27]	82.10	-	-	94.40	-
Zhao et al. [66]	**83.00**	-	-	84.10	-
Zhang et al. [61]	82.20	-	-	85.60	-
Zhang et al. [64]	-	-	-	95.52	-
McCann et al. [35]	-	-	-	**95.80**	-

Table 5.2: Performance of different input combinations on *multiInput-CNN* and other baselines.

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

These results have been mentioned in Table 5.2. Similar to Table 4.2, we mention the results of using word2vec vectors in the CNN network (*word-word2vec*) and other important works on *MR* and *TREC-En* datasets as well.

Observations

We observe that *CNN word-word2vec* performs better than *CNN-rand* (for all inputs) in most cases, perhaps because of the added contextual information from pre-trained vectors. Whereas, for Telugu, *char-ngram* and *syl-ngram* perform better than *word-word2vec*. A possible reason for this could be the huge number of Out Of Vocabulary (OOV) words (7541 out of a total of 24625 words in our case), arising due to the high number of possible inflections and agglutination. This reason is apart from those mentioned in Section 5.3.1.

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

We also note that *word-word2vec + syl-ngram* which combines word and syllable n-grams for input, outperforms all other combinations including *word-word2vec* for two out of three Indian language datasets, indicating we are successfully able to harness the information from pre-trained vectors and sub-word information with the help of syllables. Additionally, we can see that *word-word2vec + syl-ngram* performs better than *word-word2vec + char-ngram* for all the Indian language datasets.

Although, syllable based input does not seem to work well for English, probably because not all morphemes in English may be syllables, for example the plurality morpheme *-s*.

5.4 Conclusion

In this chapter, we introduced the idea of using syllable-level input instead of word or character level inputs in Natural Language Processing 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, helped us in harnessing both the contextual information and essential sub-word level information, as was also seen in Chapter 4 for character level input. 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.

Chapter 6

Attention Mechanism For Combining Different Inputs

In this final part of the thesis, we describe our attempts and experiments with using attention mechanism in the *multiInput-CNN* model. The motivation for using attention mechanism has been described in Section 6.1. This is followed by a description of the models we experimented with in Section 6.2. Finally, our experiments and results have been noted in Section 6.3.

6.1 Motivation

Attention Networks

Attention networks have recently become a popular choice for natural language processing tasks. The attention mechanism aims to combine a number of arguments to obtain a single output, which can be seen as a summary of the input arguments[1]. Usually, the input arguments are combined by taking their weighted sum, where the weights are obtained from the inputs themselves.

Attention in Sentence Classification

Attention networks have been used for sentence classification as well. Yang et al. [59] used a hierarchical attention network to obtain sentence representation from word vectors, and then document representation from sentence representations. To obtain the representation for a sentence, they passed the sentence, represented as a sequence of words, through an LSTM and used the hidden vectors of the LSTM model as input arguments in their attention network. Equation 6.1 shows how the attention is applied over the hidden vectors h_{it} in Yang et al. [59]. Where i refers to the sentence id, s_i is the representation for sentence i. t is the word id in sentence i. A fully connected layer with *tanh* activation is applied on the hidden units, followed by finding similarity measure between u_{it} and a word context vector u_w. In some other cases, this is implemented simply using a *softmax* layer. This gives

[1] `https://blog.heuritech.com/2016/01/20/attention-mechanism/`

the attention weights. These weights are then multiplied by the hidden vectors to obtain the sentence representation.

$$
\begin{aligned}
u_{it} &= \tanh(W_w h_{it} + b_w) \\
\alpha_{it} &= \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)} \\
s_i &= \sum_t \alpha_{it} h_{it}
\end{aligned}
\tag{6.1}
$$

Zhao and Wu [67] similarly used attention on the output of a CNN model to capture long range dependencies in the sentence representation. We use the same method as in Equation 6.1 to implement attention mechanism in our models (except for using *softmax* instead of inner product in the second step) as we describe in Section 6.2.

Attention in multiInput-CNN

It can be observed from Table 5.2 that the *multiInput-CNN* model performs best with a combination of *word-word2vec* and *syl-ngrams* for Hindi and Telugu language datasets, while a combination of *word-word2vec* and *char-ngrams* performed best for the English language datasets. This led us to experiment with an attention based model, which can choose the best combination of inputs for the *multiInput-CNN* model.

6.2 Models and Methodology

For the purpose of our experiments, we experimented with six different models, with minor variations. We reiterate here, that in the *multiInput-CNN* model, we run the complete CNN architectures, *CNN-rand* and *CNN-non-static* as described in Section 4.1.3 and take the average of the final layer outputs, to combine the different models. We make use of the attention layers to combine the different branches of *multiInput-CNN* in a better manner as compared to a simple *average* function. The models have been described in the following sub-sections. In each case, we implement the attention mechanism in the same manner as has been described in Equation 6.1.

6.2.1 Attn-1

In the first model, we apply attention to combine the convolution outputs for different types of inputs - words, character, syllables. For this, we apply the attention mechanism on top of the convolution layer. The convolution layer followed by the max pooling layer is applied independently for each input type. This is followed by a dense layer to bring all the vectors to a common size. On this is applied the attention layer which takes the different sentence representations obtained from the convolution operations

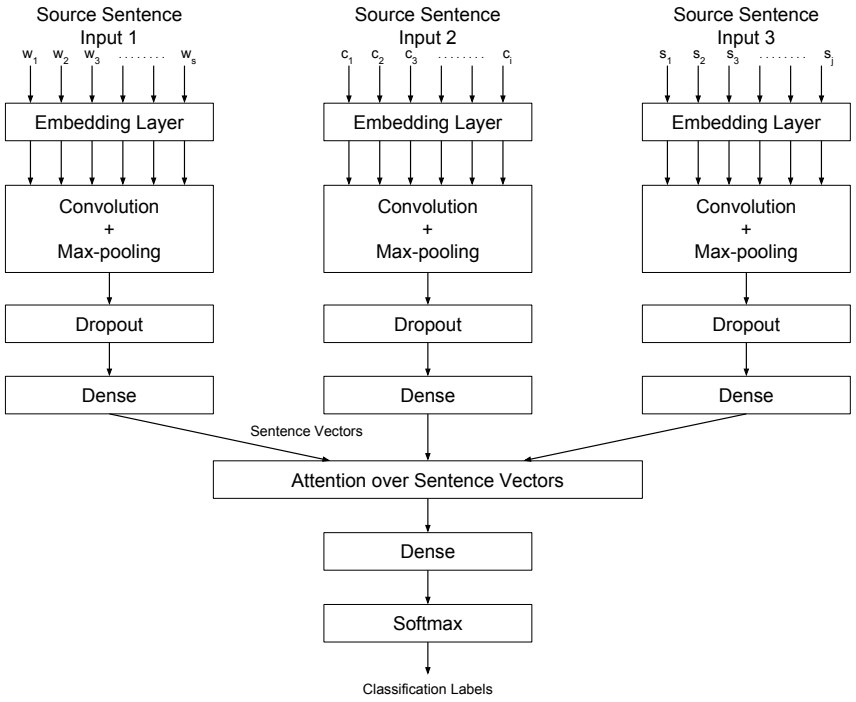

Figure 6.1: Attention in *multiInput-CNN* : *Attn-1*

as input arguments. In other words, h_{it} in Equation 6.1 is, in this case, the sentence representation for each input type. It has been shown in Figure 6.1.

6.2.2 Attn-2

This model is similar to *Attn-1* described in Section 6.2.1, except that in this case, the attention weights α_{it} in Equation 6.1 are obtained using the output of the dense layer that follows the convolution and max pooling layer. However, these weights are used to find the weighted average of the final layer's output (output of the *softmax* layer). This model has been shown in Figure 6.2.

6.2.3 Attn-3

This model is also similar to *Attn-1* (Section 6.2.1), except that, the attention layer is used to combine only the sub-word level inputs - characters and syllables together. The output of the attention layer is then passed to a dense layer with *softmax* activation which is combined with the final layer (also the *softmax* layer) output of the *multiInput-CNN* model's branch which processes word level input. Figure 6.3 illustrates this.

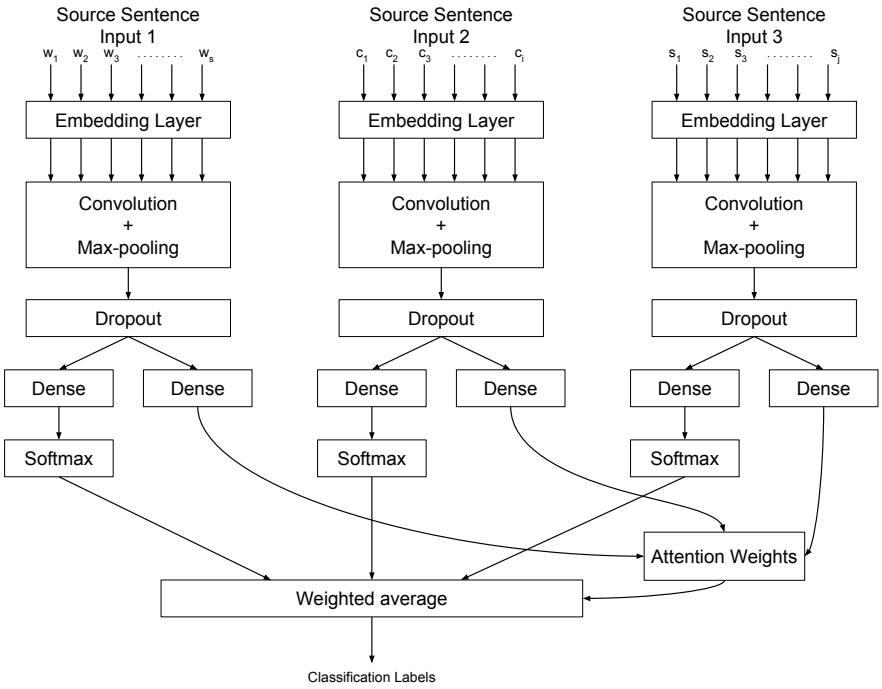

Figure 6.2: Attention in *multiInput-CNN* : *Attn-2*

6.2.4 Attn-4

This model is similar to *Attn-2* (Section 6.2.2), except that the attention weights are found only for the sub-word level inputs. First, the output of the final *softmax* layers of the sub-word branches is combined. We then take an average of this combined output with the output of the branch processing word level input to find the final output. The model has been shown in Figure 6.4.

6.2.5 Attn-3-round and Attn-4-round

Attn-3-round and *Attn-4-round* are also simple modifications of *Attn-3* (Section 6.2.3) and *Attn-4* (Section 6.2.4) respectively. The attention weights computed for the sub-word level inputs, characters and syllables, are rounded off to 0 or 1, so that only one of character or syllable level input is combined with the word level input. As discussed in Section 6.1 combining word level input with one of character or syllable level input, depending upon the language, leads to the best result. The idea in this model is to automatically find the better of character or syllable level input for a given dataset and combine only that with the word level input.

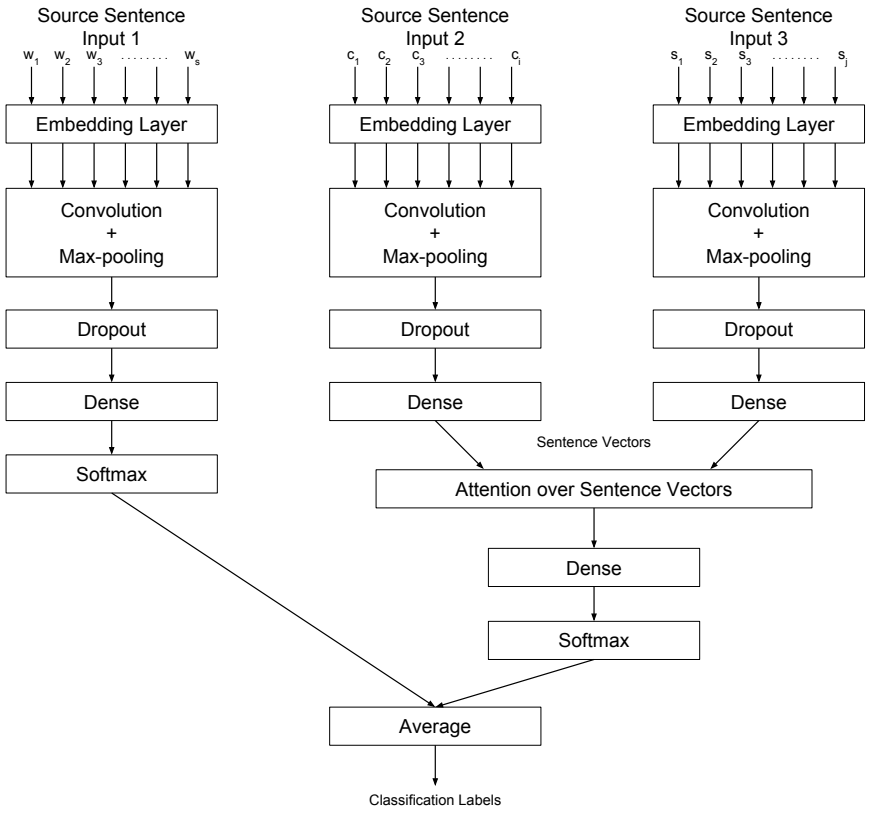

Figure 6.3: Attention in *multiInput-CNN* : *Attn-3*

6.3 Experiments and Results

We experimented with all the six models on our datasets and report the results and our observations in this section. The input for all the models is a combination of word, character and syllables (*word-word2vec* + *char-ngram* + *syl-ngram*). The parameters and n-gram sizes were fixed to the ones found to be best in previous experiments. For *Attn-1* and *Attn-2*, we experiment with a combination of only words and characters (*word-word2vec* + *char-ngram*) and words and syllables (*word-word2vec* + *syl-ngram*) in addition to a combination of all three - words, characters and syllables.

We note the results in Table 6.1. The results reported are average accuracies of five runs. In this table, we refer to *word-word2vec*, *char-ngram* and *syl-ngram* simply as *word*, *char* and *syl* respectively. Similarly, *word-word2vec* + *char-ngram* + *syl-ngram*, *word-word2vec* + *char-ngram* and *word-word2vec* + *syl-ngram* are referred to as *word+char+syl*, *word+char* and *word+syl* respectively.

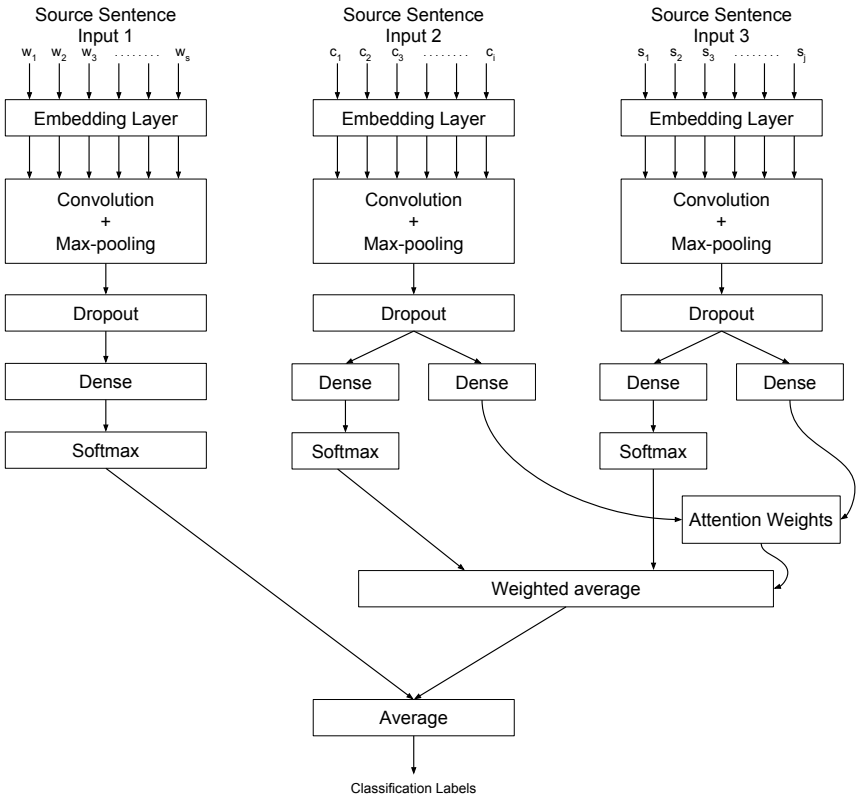

Figure 6.4: Attention in *multiInput-CNN* : *Attn-4*

Observations

In Table 6.1 we list the different experiments performed along with the results obtained in them. For each dataset, we have highlighted the maximum performance from attention based models and from our previous experiments. The overall best result has an additional '*' on it.

It can be observed from the results that the attention based models perform decently well in general, by giving the best results for two datasets - *TREC-En* and *TREC-Hi* - out of the five datasets on which we tested them. Although, different models perform well for different datasets.

The Hindi and Telugu Sentiment Analysis datasets, *Senti-Hi* and *Senti-Te*, perform the best with *Attn-4-round* model. While, English Sentiment Analysis dataset, *MR*, performs the best with *Attn-4* model. The *TREC-En* dataset performs the best for *Attn-2* model when the input is a combination of *words* and *characters*, while the *TREC-Hi* dataset performs the best with the same model but for an input combination of *words* and *syllables*. This is similar to the case of *multiInput-CNN* model where the *words* + *characters* combination yields better results for English, while the *word* + *syllables* combination gives better results for the morphologically rich languages - Hindi and Telugu.

Models where attention is used to find the better of characters or syllables (*Attn-3*, *Attn-4*, *Attn-3-round* and *Attn-4-round*) perform better than the other models for the Sentiment Analysis datasets while

Model	Model-Input	MR	Senti-Hi	Senti-Te	TREC-En	TREC-Hi
Attn-1	*word+char*	79.79	72.41	57.92	92.48	93.47
	word+syl	79.44	73.88	57.34	91.68	93.87
	word+syl+char	79.24	73.31	57.77	91.20	93.55
Attn-2	*word+char*	80.05	73.07	58.74	**94.36***	93.51
	word+syl	78.91	73.52	57.92	90.76	**94.03***
	word+syl+char	80.01	73.37	58.31	91.12	93.59
Attn-3	*word+syl+char*	78.61	68.04	58.34	93.08	91.98
Attn-4	*word+syl+char*	**80.95**	74.47	58.66	93.32	91.90
Attn-3-round	*word+syl+char*	78.12	63.38	58.47	92.48	91.58
Attn-4-round	*word+syl+char*	80.90	**74.58**	**58.80**	93.72	92.26
multiInput-CNN	*word+char*	**81.03***	74.51	58.86	**93.96**	92.22
	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
CNN-non-static	*word*	80.21	73.86	55.50	93.24	**93.71**
CNN-rand	*word-rand*	74.97	71.02	53.54	90.74	86.19
	char	77.43	72.73	57.97	87.80	85.35
	syl	76.63	72.97	58.09	90.60	87.05

Table 6.1: Performance of different attention based models in comparison with our earlier models. Best results have been highlighted for attention and non-attention models. Overall best results have been marked with a *.

the other models, where attention is applied on all the three inputs, perform better for the Question Classification datasets.

Rounding in model *Attn-4* helps in most datasets (all except for *MR*), as can be seen from the results, although the results show very small improvements. While, this is not true for *Attn-3*.

It is interesting to note that the models where the attention weights are derived from the output of the dense layer which follows the convolution and max pooling layer and used to find the weighted average of the final (*softmax*) layer output perform better than those where the weights are derived from and applied to find the weighted average of the same layer output. As, *Attn-2* can be seen to be performing better than *Attn-1*, *Attn-4* performs better than *Attn-3* and *Attn-4-round* performs better than *Attn-3-round*.

Although, attention models perform decently well, it can be seen that the *multiInput-CNN* model performs better than the attention based models in three out of five datasets. The benefit of the attention based models, particularly *Attn-4* and *Attn-4-round* models, lies in that they perform decently well for all languages and datasets for the same input.

6.4 Conclusion

In this chapter, we experimented with various attention based extensions of our *multiInput-CNN* model introduced in Chapter 4. As we had found in Chapter 5, the *multiInput-CNN* model performs best with a combination of *word-word2vec* and *syl-ngrams* for Hindi and Telugu language datasets, while a combination of *word-word2vec* and *char-ngrams* performed best for the English language datasets. This lead us to experiment with attention mechanism to build a model that can automatically determine the best combination of inputs for a given language and dataset.

We experimented with six attention based models. We found that, although, attention based models achieve the state-of-the art for only two out of five datasets, they performed decently well for all the datasets (in particular, models *Attn-4* and *Attn-4-round*) with the same set of inputs. This makes them readily usable for any language or task.

Chapter 7

Conclusions

The huge number of languages spoken around the world make it difficult to build an independent classifier for each task and language. We thus, aimed at building a language-independent and task-independent classifier in this work. We first experimented with different deep learning models on different types of inputs. This helped us to show that as we move towards morphologically rich languages, character n-gram based inputs lead to better performance than other input methods for most of the models. Another major conclusion from this part of our work was that models based on Convolution Neural Networks (CNNs) are more suitable for classification purposes as compared to other models based on Recurrent or Recursive Neural Networks (RNNs).

The conclusions drawn from the evaluations enabled us to present a model, *multiInput-CNN* that capitalizes on both word and character based inputs to give us a language independent model. While the character based embeddings provide morphological information to the model, pre-trained word embeddings carry contextual information with them.

In the second part of our work, we introduced the idea of using syllable-level input instead of word or character level input in NLP tasks for morphologically rich and agglutinating languages. We evaluated syllable level input on two different neural network architectures and found that it outperformed the other types of inputs for morphologically rich languages. Similarly, a combination of words and syllable based input performed better than a combination of words and characters for Hindi and Telugu.

Therefore, in the last part we tried out various attention models so as to automatically find the best combination of inputs for a given dataset. We found that, although *multiInput-CNN* model performed better than the attention based models, the *Attn-4* and *Attn-4-round* models performed decently well for all languages. This makes them readily usable for any language or task.

In the future, we can focus on pre-training embeddings for character n-grams and syllable n-grams, that are able to capture morphological and contextual information. Word embeddings that take sub-word level information into account, for example FastText[1] [3]. Vectors can also be considered. Additionally, one could focus on improving the syllabification process for English, perhaps by converting every word

[1] `https://fasttext.cc/`

first into its phonetic representation. Another possible extension is to build a model that can find the best split for a given sentence (into words, characters and syllables) depending upon the task.

Appendix A

Experimental Settings and Model Parameters

In this chapter, we describe in detail the model parameters used for our experiments.

A.1 Support Vector Machines

We experimented with linear and RBF kernels using the *Scikit* [44] library. For all the experiments, we set minimum and maximum frequency parameters to 1. Since RBF kernel lead to bad results consistently across all tasks and languages, the parameters mentioned here are only for linear kernel. In table A.1, we mention the analyzer (*word*, *character* or a combination of both *word-char*) and n-gram sizes found to lead to the best results.

Dataset	**Analyzer**	**N-gram size**
MR	Word-Char	1,3 - 5,8
Senti-Te	Word-Char	1,2 - 5,7
Senti-Hi	Word-Char	1,2 - 5,5
TREC-En	Word-Char	1,2 - 4,6
TREC-Hi	Character	3,6

Table A.1: Model parameters for SVM experiments. In case the analyzer is *Word-Char*, the n-gram size is : *n-gram size range for words - n-gram size range for characters*

A.2 LSTMs

We used *tanh* activation, *sigmoid* recurrent activation and *adam* optimizer for our experiments. In table A.2 we mention the input type, input vector dimension and number of hidden units which lead to best results in our experiments.

Dataset	Input Type	Input Dimension	Hidden Units
MR	word-rand	1500	100
Senti-Te	char-ngram	300	100
Senti-Hi	word-rand	1500	500
TREC-En	word-rand	1500	100
TREC-Hi	word-rand	1500	500

Table A.2: Model parameters for LSTM experiments.

A.3 Bi-LSTMs

We used *tanh* activation, *sigmoid* recurrent activation and *adam* optimizer for our experiments. In table A.3 we mention the input type, input vector dimension and number of hidden units which lead to best results in our experiments. The network was trained for 10 epochs.

Dataset	Input Type	Input Dimension	Hidden Units
MR	word-rand	100	50
Senti-Te	word-rand	1500	100
Senti-Hi	word-rand	2000	100
TREC-En	word-rand	100	50
TREC-Hi	word-rand	2000	50

Table A.3: Model parameters for Bi-LSTM experiments.

A.4 CNNs

We fixed the dropout rate at *0.5*, linear decay at *0.95* and activation to *ReLU*. The network was trained for 25 epochs.

A.4.1 Random Word Vector Input

In table A.4 we mention the input dimension, number of hidden units and filter sizes which lead to best results in our experiments.

Dataset	Input Dimension	Hidden Units	Filter Sizes
MR	100	600	3, 4, 5, 6
Senti-Te	1500	700	1, 2
Senti-Hi	1000	600	2, 5, 7, 8
TREC-En	1500	300	6
TREC-Hi	300	800	6

Table A.4: Model parameters for CNN *word-rand* experiments.

A.4.2 Word2Vec Word Vector Input

The input dimension in this case is always fixed to *300*. In table A.5 we mention the number of hidden units and filter sizes which lead to best results in our experiments.

Dataset	Hidden Units	Filter Sizes
MR	100	6
Senti-Te	100	2
Senti-Hi	600	2, 5, 7, 8
TREC-En	600	6
TREC-Hi	800	6

Table A.5: Model parameters for CNN *word-word2vec* experiments.

A.4.3 Character n-gram Input

In table A.6 we mention the n-gram size, input dimension, number of hidden units and filter sizes which lead to best results in our experiments.

Dataset	N-Gram Size	Input Dimension	Hidden Units	Filter Sizes
MR	4	300	700	3, 4, 5
Senti-Te	3	500	300	2, 3, 4, 5
Senti-Hi	3	300	500	2, 4, 6
TREC-En	5	300	300	5
TREC-Hi	8	300	300	7

Table A.6: Model parameters for CNN *char-ngram* experiments.

A.4.4 Syllable n-gram Input

In table A.7 we mention the n-gram size, input dimension, number of hidden units and filter sizes which lead to best results in our experiments.

Dataset	N-Gram Size	Input Dimension	Hidden Units	Filter Sizes
MR	3	1000	600	1, 2, 3, 4
Senti-Te	2	1000	500	2
Senti-Hi	2	1000	700	3, 4, 5
TREC-En	3	1500	200	3, 5, 6, 8
TREC-Hi	2	100	700	7

Table A.7: Model parameters for CNN *syl-ngram* experiments.

Related Publications

- **Madhuri Tummalapalli**, Manoj Chinnakotla and Radhika Mamidi.
  Towards Better Sentence Classification for Morphologically Rich Languages.
  In Proceedings of the 19th International Conference on Computational Linguistics and Intelligent Text Processing (CICLING, 2018) in Hanoi, Vietnam. *TO APPEAR*.

- **Madhuri Tummalapalli** and Radhika Mamidi.
  Syllables for Sentence Classification in Morphologically Rich Languages.
  In Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation (PACLIC), December 1-3, 2018 at Hong Kong Polytechnic University, Hong Kong. *TO APPEAR*.

Bibliography

[1] W. Becker, J. Wehrmann, H. Cagnini, and R. Barros. An efficient deep neural architecture for multilingual sentiment analysis in twitter. 05 2017.

[2] P. Blunsom, K. Kocik, and J. R. Curran. Question classification with log-linear models. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 615–616. ACM, 2006.

[3] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[4] X. Chen, Y. Sun, B. Athiwaratkun, C. Cardie, and K. Weinberger. Adversarial deep averaging networks for cross-lingual sentiment classification. *arXiv preprint arXiv:1606.01614*, 2016.

[5] F. Chollet et al. Keras. https://github.com/keras-team/keras, 2015.

[6] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun. Very deep convolutional networks for text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 1107–1116, 2017.

[7] M. Á. G. Cumbreras, L. López, and F. M. Santiago. Bruja: question classification for spanish. using machine translation and an english classifier. In *Proceedings of the Workshop on Multilingual Question Answering*, pages 39–44. Association for Computational Linguistics, 2006.

[8] J. Deriu, A. Lucchi, V. De Luca, A. Severyn, S. Müller, M. Cieliebak, T. Hofmann, and M. Jaggi. Leveraging large amounts of weakly supervised data for multi-language sentiment classification. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1045–1052. International World Wide Web Conferences Steering Committee, 2017.

[9] L. Dong, F. Wei, M. Zhou, and K. Xu. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *AAAI*, pages 1537–1543, 2014.

[10] A. Ekbal, S. K. Naskar, and S. Bandyopadhyay. A modified joint source-channel model for transliteration. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 191–198. Association for Computational Linguistics, 2006.

[11] Z. Gan, Y. Pu, R. Henao, C. Li, X. He, and L. Carin. Learning generic sentence representations using convolutional neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2390–2400, 2017.

[12] A. Ganapathiraju, J. Hamaker, J. Picone, M. Ordowski, and G. R. Doddington. Syllable-based large vocabulary continuous speech recognition. *IEEE Transactions on speech and audio processing*, 9(4):358–366, 2001.

[13] A. Graves, N. Jaitly, and A.-r. Mohamed. Hybrid speech recognition with deep bidirectional lstm. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 273–278. IEEE, 2013.

[14] A. Hassan and A. Mahmood. Deep learning for sentence classification. In *Systems, Applications and Technology Conference (LISAT), 2017 IEEE Long Island*, pages 1–5. IEEE, 2017.

[15] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] S. T. Hsu, C. Moon, P. Jones, and N. F. Samatova. A hybrid cnn-rnn alignment model for phrase-aware sentence classification. *EACL 2017*, page 443, 2017.

[17] Z. Huang, M. Thint, and Z. Qin. 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, 2008.

[18] O. Irsoy and C. Cardie. Deep recursive neural networks for compositionality in language. In *Advances in neural information processing systems*, pages 2096–2104, 2014.

[19] R. Johnson and T. Zhang. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*, 2014.

[20] D. Jurafsky and J. H. Martin. Speech and language processing, 2nd edition, 2008.

[21] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[22] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[23] S. P. Kishore and A. W. Black. Unit size in unit selection speech synthesis. In *Eighth European Conference on Speech Communication and Technology*, 2003.

[24] A. Komninos and S. Manandhar. Dependency based embeddings for sentence classification tasks. In *HLT-NAACL*, pages 1490–1500, 2016.

[25] E. Kouloumpis, T. Wilson, and J. D. Moore. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164, 2011.

[26] A. Kunchukuttan and P. Bhattacharyya. Orthographic syllable as basic unit for smt between related languages. *arXiv preprint arXiv:1610.00634*, 2016.

[27] S. Li, Z. Zhao, T. Liu, R. Hu, and X. Du. 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, 2017.

[28] W. Li and B. Mak. Derivation of document vectors from adaptation of lstm language model. *EACL 2017*, page 456, 2017.

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

[30] X. Li and D. Roth. Learning question classifiers: The role of semantic information. In *Proceedings of the International conference on Computational linguistics*, pages 556–562, 2004.

[31] F. M. Liang. Word hy-phen-a-tion by com-put-er. Technical report, Calif. Univ. Stanford. Comput. Sci. Dept., 1983.

[32] B. Liu, Z. Hao, X. Yang, and X. Lin. Chinese question classification with support vector machine. *IJCSNS Int. Journal of Computer Science and Network Security*, 6:231–240, 2006.

[33] B. Loni. A survey of state-of-the-art methods on question classification. 2011.

[34] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics, 2011.

[35] B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308, 2017.

[36] 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.

[37] S. S. Mukku and R. Mamidi. Actsa: Annotated corpus for telugu sentiment analysis. In *Proceedings of the First Workshop on Building Linguistically Generalizable NLP Systems*, pages 54–58, 2017.

[38] S. S. Mukku, N. Choudhary, and R. Mamidi. Enhanced sentiment classification of telugu text using ml techniques. In *SAAIP@ IJCAI*, pages 29–34, 2016.

[39] S. S. Mukku, S. R. Oota, and R. Mamidi. Tag me a label with multi-arm: Active learning for telugu sentiment analysis. In *International Conference on Big Data Analytics and Knowledge Discovery*, pages 355–367. Springer, 2017.

[40] G.-S. Nguyen, X. Gao, and P. Andreae. 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, 2009.

[41] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, 2010.

[42] B. Pang and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*, 2005.

[43] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[44] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[45] N. Qun, X. Li, X. Qiu, and X. Huang. 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, 2017.

[46] K. C. Raghavi, M. K. Chinnakotla, and M. Shrivastava. Answer ka type kya he?: Learning to classify questions in code-mixed language. In *Proceedings of the 24th International Conference on World Wide Web*, pages 853–858. ACM, 2015.

[47] R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

[48] C. Schrumpf, M. Larson, and S. Eickeler. 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, 2005.

[49] J. Silva, L. Coheur, A. C. Mendes, and A. Wichert. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35(2):137–154, 2011.

[50] S. Singh and V. M. Sarma. Verbal inflection in hindi: A distributed morphology approach. In *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation*, 2011.

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

[52] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. 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, 2013.

[53] T. Solorio, M. Pérez-Coutino, M. Montes-y Gémez, L. Villasenor-Pineda, and A. López-López. A language independent method for question classification. In *Proceedings of the 20th international conference on Computational Linguistics*, page 1374. Association for Computational Linguistics, 2004.

[54] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

[55] R. Tsarfaty, D. Seddah, Y. Goldberg, S. Kübler, M. Candito, J. Foster, Y. Versley, I. Rehbein, and L. Tounsi. 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, 2010.

[56] X. Wan. Co-training for cross-lingual sentiment classification. 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: Volume 1-volume 1*, pages 235–243. Association for Computational Linguistics, 2009.

[57] J. Wehrmann, W. Becker, H. E. Cagnini, and R. C. Barros. 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, 2017.

[58] Y. Xia, Z. Wei, and Y. Liu. An efficient cross-lingual model for sentence classification using convolutional neural network. *ACL 2016*, page 126, 2016.

[59] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489, 2016.

[60] W. Yin and H. Schütze. Multichannel variable-size convolution for sentence classification. *arXiv preprint arXiv:1603.04513*, 2016.

[61] R. Zhang, H. Lee, and D. Radev. Dependency sensitive convolutional neural networks for modeling sentences and documents. *arXiv preprint arXiv:1611.02361*, 2016.

[62] X. Zhang and Y. LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[63] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[64] Y. Zhang, S. Roller, and B. Wallace. Mgnc-cnn: A simple approach to exploiting multiple word embeddings for sentence classification. *arXiv preprint arXiv:1603.00968*, 2016.

[65] Y. Zhang, M. Lease, and B. C. Wallace. Active discriminative text representation learning. In *AAAI*, pages 3386–3392, 2017.

[66] R. Zhao, K. Mao, R. Zhao, and K. Mao. Topic-aware deep compositional models for sentence classification. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 25 (2):248–260, 2017.

[67] Z. Zhao and Y. Wu. Attention-based convolutional neural networks for sentence classification. In *INTERSPEECH*, pages 705–709, 2016.

[68] X. Zhou, X. Wan, and J. Xiao. Attention-based lstm network for cross-lingual sentiment classification. In *EMNLP*, pages 247–256, 2016.

[69] X. Zhou, X. Wan, and J. Xiao. Cross-lingual sentiment classification with bilingual document representation learning. In *ACL (1)*, 2016.