State-of-the-art contextual word emebeddings for SMS Spam Detection

Sarthak Jindal
2015169 / IIITD
sarthak15169@iiitd.ac.in

Abstract

In this project, I will implement the state-of-the-art representation for words which are contextual string embeddings. The main idea behind contextual string embeddings is to capture the context in which a word occurs and also treat a word as a sequence of characters. This is important and improves performance of both NLP and IR models since some words have different meanings in different contexts. Also, misspelled words cannot be handled properly if words are not looked at as a sequence of characters. After processing the document and queries as vectors of contextual embeddings, I will use neural pseudo relevance feeback mechanism to create better modified queries, suited to the information need. This approach will definitely outperform trivial methods for Pseudo Relevance feedback such as Rochhio algorithm.

1 Introduction

The main discriminative content within a data point is text. Deep learning has been frequently used on textual data since a long time now. However, raw text cannot be fed as input to deep learning models. So, we convert the raw text into a matrix or a 2d-tensor. This can be done by converting each word into a vector or a 1d-tensor and then visualising the text as a matrix in which each row is a word vector. The conversion of word to vectors is explained in detail below.

Once we have converted our words to vectors, we simply feed our text matrix as an input to the text branch for each data point. Convolutional neural networks can then be used to capture spatial relationships in this text matrix. These are mostly relationships between similar dimensions of consecutive or adjacent words in the text.

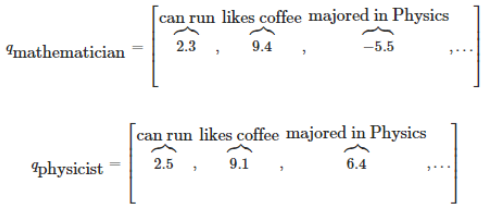

Figure 1: An example of the idea of word embeddings. Both the mathematician and physicist have high values in the semantic dimensions which are similar for both of them. But they have alternate sign values in the semantic dimension for which they are opposite to each other. The image has been borrowed from PyTorch tutorials.

1.0.1 Converting words to vectors

One simple idea to convert words into vectors could be to use one-hot encoded vectors. For this we come up with the entire vocabulary set which is the set of all words present in the text of all our data points. Then we create word vectors with number of dimensions equal to the number of words in the vocabulary. To get the word vector of a given word, we simply set a 1 in the position corresponding to that word in an indexed list of all words present in the text of all our data points. In this way, we get one-hot encoded vectors representing our words.

However, this approach has two problems. First, the size of the vector is very large as the number of dimensions is equal to the number of words in the entire vocabulary. All of the dimensions are 0 except one of them which is 1. So, these word vectors are very sparse, using more space to represent less information. And the second reason is that, such a representation of a word considers no semantic relationship between

words. For example, consider the one hot encoded vectors of cat and animal. They look completely different because they have ones in different positions and zeroes in all others. However, cat and animals are related words. We need to capture this semantic relationship between them.

So, in a better word vector representation, there must be some dimensions in the word vector which has some non-zero values for both of them. Hence, we decide to use word embeddings. Word embeddings are dense vectors represnting words in which each dimension represents some hidden semantic relationship. These vectors are dense because they have mostly non-zero real values in each dimension. Each dimension represents a semantic relationship. For example, the word embeddings of cat and animal would both have high values in some common dimensions representing the commonality between the two words (i.e. they are animals). Therefore, word embeddings of similar words point in similar directions when visualised in the d-dimensional space where d is the dimension of the word embedding. Similarly, word embeddings of different meaning words point in almost opposite directions. An example of word embeddings is also shown.

Word embeddings are fundamental to most NLP and IR Problems. The most simple word embeddings do not take into account the context of a word. However, as is obvious, different words can mean different things in separate contexts. A recent approach to better word embeddings is contextual word embeddings.

These use the context of a word to create an embedding for it. So, the same word in different contexts have different embeddings. Also, it is important to view words as a sequence of characters rather than a single block. This is exactly another one of the features of contextual word embeddings due to which they can cpture subword structures such as prefixes etc.

Once the word embedding are trained building the IR model is a fairly simple and already well explored domain. As is known, pseudo relevance feedback can help modify a query to better suit it to the information need without involving any human annotation as in direct relevance feedback. A neural network based Pseudo Relevance feedback mechanism, can help improve performance to a large extent.

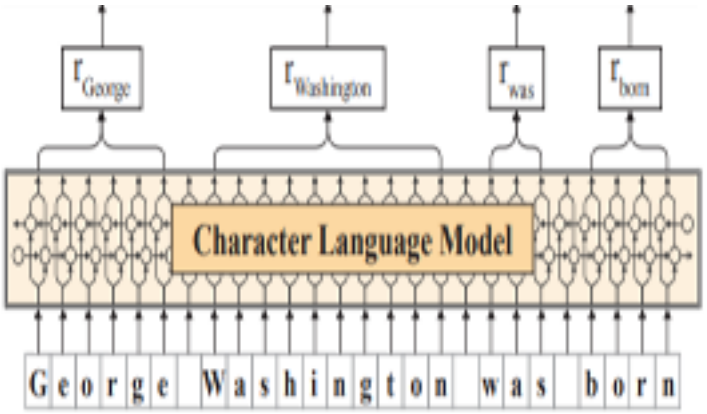

Figure 2: Overview of the contextual word embeddings borrowed from their research paper. The main idea is to use the hidden states of BiRNN based character level language model to predict the contextual word embeddings.

2 Dataset

The dataset has been taken from Kaggle. The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according being ham (legitimate) or spam.

The files contain one message per line. Each line is composed by two columns: v1 contains the label (ham or spam) and v2 contains the raw text.

This corpus has been collected from free or free for research sources at the Internet. A collection of 425 SMS spam messages was manually extracted from the Grumbletext Web site. This is a UK forum in which cell phone users make public claims about SMS spam messages, most of them without reporting the very spam message received. The identification of the text of spam messages in the claims is a very hard and time-consuming task, and it involved carefully scanning hundreds of web pages. The Grumbletext Web site is a subset of 3,375 SMS randomly chosen ham messages of the NUS SMS Corpus (NSC), which is a dataset of about 10,000 legitimate messages collected for research at the Department of Computer Science at the National University of Singapore. The messages largely originate from Sin-

Source of SMS	No. of Ham SMS collected	No. of Spam SMS collected
Grumbletext Web site	0	425
NUS SMS Corpus (NSC)	3,375	0
Caroline Tag's PhD Thesis	450	0
SMS Spam Corpus v.0.1 Big	1,002	322
Total	4827	747

Table 1: SMS Collection sources used in the Dataset

Ground Truth/ Prediction	Ham	Spam
Ham	452	8
Spam	3	54

Table 2: Confusion Matrix for Legitimate(Ham) and Spam SMS messages

Metric	Ham	Spam
Precision	0.9826	0.9474
Recall	0.9934	0.8710
Accuracy	0.9762	0.8308
F1-Score	0.9880	0.9076

Table 3: Evaluation Metrics for Legitimate(Ham) and Spam SMS messages

gaporeans and mostly from students attending the University. These messages were collected from volunteers who were made aware that their contributions were going to be made publicly available. The NUS SMS Corpus is a list of 450 SMS ham messages collected from Caroline Tag's PhD Thesis. Finally, we have incorporated the SMS Spam Corpus v.0.1 Big. It has 1,002 SMS ham messages and 322 spam messages. This distribution is summarized in Table 1.

3 Results

The confusion matrix results are summarized in the Table 2. The evaluation metric results are summarized in the Table 3.

4 Merits/Demerits of the Proposed Solution Sketch

4.1 Merits

The biggest advantage is the incorporation of context in deciding a word embedding. This lets us capture the semantic relationships between words in different contexts. Since a word may not mean the same in different contexts it is not accurate to have a fixed word embedding for it independent of contexts.

Another advantage is the handling of misspelled words correctly. A misspelled word may have just one or two characters displaced as compared to their original counterpart. But an embedding procedure that looks at the word as a whole will not be able to recognize it. Hence, the contextual word embeddings perform better here as well. This is mainly because they look at words like a sequence of characters rather than merely a single block.

The major advantage of using a neural pseudo relevance feed mechanism is that it can adaptively provide feedback for different queries, instead of just expanding the query through the top-K ranked documents.

4.2 Demerits

The main demerit of our approach is that it takes time to train the contextual word embeddings each time, since the context keeps changing. It will also require more memory to store different contextual embeddings of a word.

References

[1] Rada Mihalcea and Carlo Strapparava. The lie detector: Explorations in the automatic recognition of deceptive language. In Proceedings of the ACL-IJCNLP 2009 Conference Short Papers, pages 309312. Association for Computational Linguistics, 2009.

[2] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 168177. ACM, 2004

[3] Song Feng, Ritwik Banerjee, and Yejin Choi. Syntactic stylometry for deception detection. In Pro-

ceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, pages 171175. Association for Computational Linguistics, 2012.