Sentiment Analysis of Drug Reviews

Natural Language Processing

CS60075

Members:

Rajesh Khanna (16CS10014)

Uppada Vishnu (16CS30037)

Lakkam Sai Krishna Reddy (16CS30018)

Seelaboyina Sasi Bhushan (16CS30032)

Table of Contents

`

1.Task Overview

Reviews posted on various websites contain important information regarding the usage and sale of a product. There is a huge corpus containing the reviews of drugs given by the customers. Each review is rated on a scale of 1-10. The reviews rated 1-4 are negative, 5-6 are neutral and 7-10 are positive. In this task, we explore classifying each review as positive, negative or neutral.

2. Introduction

Sentiment analysis is a type of subjectivity analysis which analyzes sentiment in a given textual unit with the objective of understanding the sentiment polarities (i.e. positive, negative, or neutral) of the opinions regarding various aspects of a subject (drug reviews). It is still considered as a very challenging problem since user generated content is described in various and complex ways using natural language.

For sentiment analysis, most researchers have worked on general domains (such as electronic products, movies, and restaurants), but not extensively on health and medical domains. Previous studies have shown that this health-related user-generated content is useful from different points of view. First, users are oen looking for stories from “patients like them” on the Internet, which they cannot always find among their friends and family (Sarasohn-Kahn, 2008). Moreover, studies investigating the impact of social media on patients have shown that for some diseases and health problems, online community support can have a positive effect (Jaloba, 2009; Schraefel et al., 2009). Because of its novelty as well as quality and trustworthiness issues, user-generated content of social media in health and medical domains is underexploited. It needs to be further studied, understood, and then leveraged in designing new online tools and applications.

3. Motivation

As we know that Sentiment Analysis has not been used much in Medical Domain. It has been shown that performing sentiment analysis on drug reviews is useful in many ways

like when a new drug is released or used users or patients publish their opinions about the drug on the social Web. The sentiment analysis results of drug reviews will be useful not only for patients to decide which drugs they should buy or take, but also for drug makers and clinicians to obtain valuable summaries of public opinion and feedback. Sentiment analysis can also highlight patients' misconceptions and dissenting opinions about a drug. Therefore, we perform Sentiment Analysis on drug reviews using various methods to know the pros and cons of a specific drug based on its review.

4. Implementations

We have implemented 3 models TextCNN [1], RCNN [2] and Seq2seq with attention [3]. Each model is implemented with 3 embeddings GloVe [4], Word2Vec [5] and ELMo [6] independently. So there are a total of 9 models.

4.1. Model Architecture

4.1.1. TextCNN

Convolutional Neural Networks (CNN) are used for image classification or recognition etc.that use convolution in place of general matrix multiplication in at least one of their layers. TextCNNs [1] are inspired from CNNs that is used for document classification (sentiment analysis) in which the document (review) is seen as an image i.e. just as we use CNN for images, we use textCNN for documents or reviews.

The words in each review are converted to vectors using the embeddings and this review now becomes a 2d matrix ($number\ of\ words \times vector\ size$) where as an image is a 2d matrix of pixel values. Instead of image pixels, the input to the tasks is sentences or documents (reviews) represented as a matrix. Each row of the matrix corresponds to one-word vector.

Structure:

embedding ---> conv ---> max pooling ---> fully connected layer ---> softmax

Sentence length will be different from one to another. So we will use pad to get fixed length, n. For each token in the sentence, we will use word embedding to get a fixed dimension vector, d. So our input is a 2-dimension matrix: (n, d). This is similar with

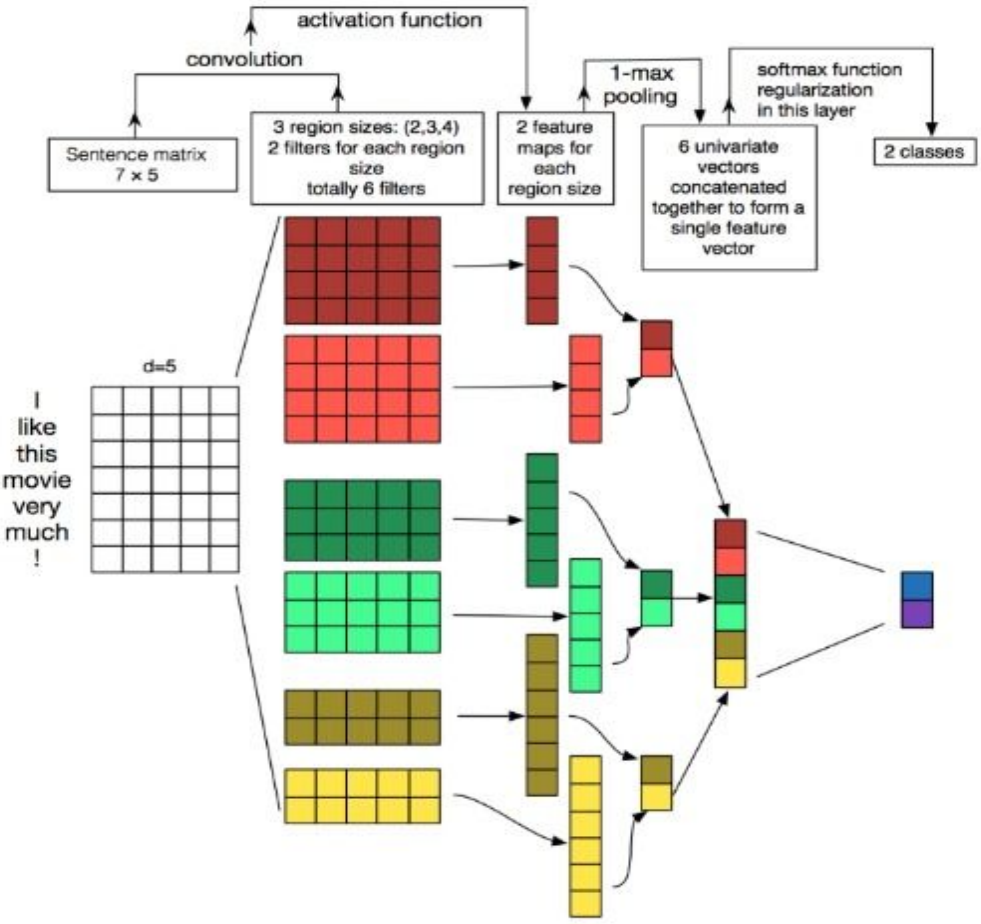
image for CNN.

First, we will do convolutional operation to our input. It is a element-wise multiply between filter and part of input. We use k number of filters, each filter size is a 2-dimension matrix (f, d). Now the output will be k number of lists. Each list has a length of $n+f+1$. each element is a scalar. Notice that the second dimension will be always the dimension of word embedding. We are using different sizes of filters to get rich features from text inputs. And this is something similar with n-gram features.

Second, we will do max pooling for the output of convolutional operation. For k number of lists, we will get k number of scalars.

Third, we will concatenate scalars to form final features. It is a fixed-size vector. And it is independent from the size of filters we use.

Finally, we will use linear layer to project these features to per-defined labels.



A Sample textCNN that classifies the sentence ‘I like this movie very much!’

4.1.2. RCNN

RCNN (Recurrent Convolutional Neural Network) [2] is used for text classification (Sentiment Analysis) with captures contextual information with the recurrent structure and constructs the representation of text using a convolutional neural network.

In this model, we apply a recurrent structure to capture contextual information as far as possible when learning word representations, which introduces considerably less noise compared to traditional window-based neural networks. We also employ a max-pooling layer that automatically judges which words play key roles in text classification to capture the key components in texts.

Structure:

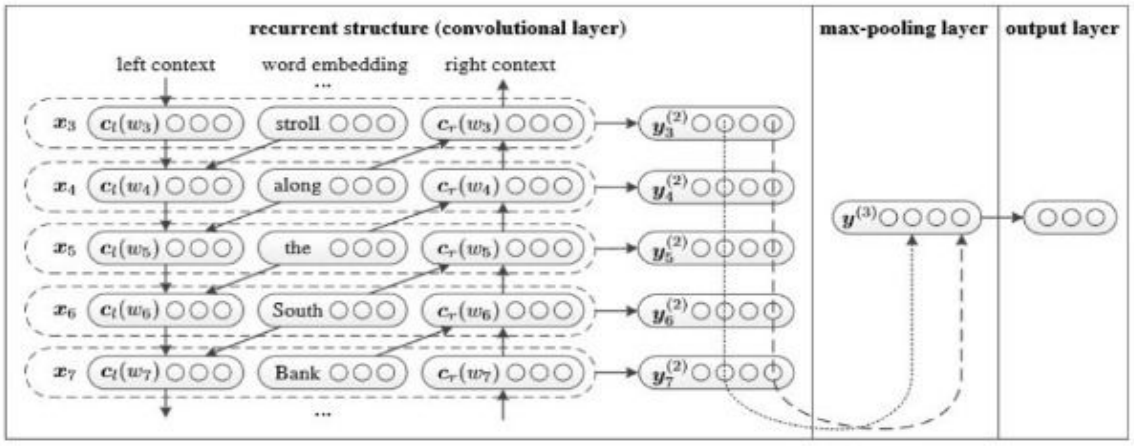
1. Recurrent structure (Convolutional layer)
2. Max Pooling
3. Fully connected layer + Softmax.

It learns the representation of each word in the sentence or document with left side context and right side context

Representation:

$$current\_word = [\ left\ side\ context\ vector,\ current\ word\ embedding,\ right\ side\ context\ vector\ ]$$

We pass the embeddings of each word in the review. For the left side context, it uses a recurrent structure, a non-linear transform of previous word and left side previous context; similarly to capture the right side context, it uses the recurrent structure, a non-linear transform of the next word.



The structure of the recurrent convolutional neural network. This figure is a partial example of the sentence "*A sunset stroll along the South Bank affords an array of stunning vantage points*" , and the subscript denotes the position of the corresponding word in the original sentence.

4.1.3. Seq2seq with Attention

This model belongs to the family of encoder–decoders which is usually used in machine translation [3]. In this model encode a source sentence into a fixed-length vector from which a decoder generates a translation. This model is converted for the purpose of

sentiment analysis (classification) by converting the decoder which generates an output vector and the class is assigned based on which class vector is most similar to the output vector generated by the decoder and attention is also applied to reduce the bottle-neck problem.

Structure:

1. Embedding
2. Bi-GRU to get rich representation from source sentences(forward & backward).
3. decoder with attention.

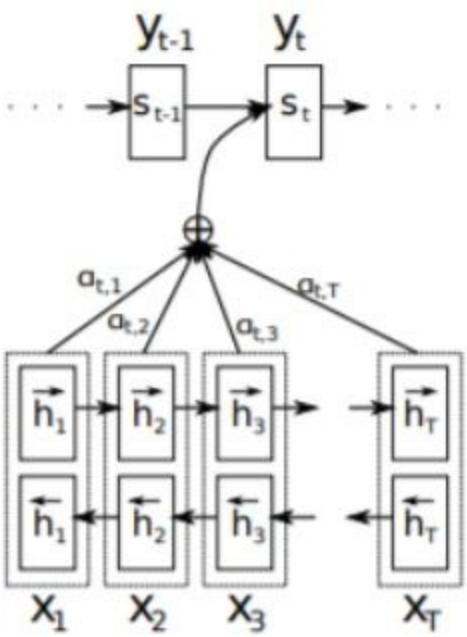
There are two kinds of three kinds of inputs:

1. Encoder inputs, which is a sentence
2. Decoder inputs, it is labels list with fixed length
3. Target labels, it is also a list of labels.
   For example, if labels are: "$L1\ L2\ L3\ L4$" , then decoder inputs will be: $[\_GO,\ L1,\ L2,\ L2,\ L3,\ \_PAD]$ and target label will be: $[L1,\ L2,\ L3,\ L3,\ \_END,\ \_PAD\ ]$. length is fixed to 6, any exceed labels will be truncated, will pad if label is not enough to fill.

Attention Mechanism:

1. Transfer encoder input list and hidden state of decoder.
2. Calculate similarity of hidden state with each encoder input, to get possibility distribution for each encoder input.
3. Weighted sum of encoder input based on possibility distribution. Go through RNN Cell using this weighted sum together with decoder input to get new hidden state.

for the vocabulary of labels, 3 special tokens are considered: "$\_GO$", "$\_END$", "$\_PAD$"; "$\_UNK$" is not used, since all labels are pre-defined.



The Seq2seq model trying to generate the prediction y_t from the decoder given a source review ($x_1, x_2, \ldots, x_T$)

4.2. Embeddings

We have used 3 embeddings for each model: GloVe, Word2vec and ELMo.

4.2.1. GloVe

GloVe [4], coined from Global Vectors , because the global corpus statistics are captured directly by the model is a model for distributed word representation. The model is an unsupervised learning algorithm for obtaining vector representations for words. This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

It is developed as an open-source project at Stanford. As log-bilinear regression model for unsupervised learning of word representations, it combines the features of two

model families, namely the global matrix factorization and local context window methods.

4.2.2. Word2vec

Word2vec [5,7] is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space.

Word2vec can utilize either of two model architectures to produce a distributed representation of words: continuous bag-of-words (CBOW) or continuous skip-gram. In the continuous bag-of-words architecture, the model predicts the current word from a window of surrounding context words. The order of context words does not influence prediction (bag-of-words assumption). In the continuous skip-gram architecture, the model uses the current word to predict the surrounding window of context words. The skip-gram architecture weighs nearby context words more heavily than more distant context words. CBOW is faster while skip-gram is slower but does a better job for infrequent words.

4.2.3. ELMo

ELMo [6] is a deep contextualized word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy). These word vectors are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. They can be easily added to existing models and significantly improve the state of the art across a broad range of challenging NLP problems, including question answering, textual entailment and sentiment analysis.

ELMo representations are:

- ***Contextual*:** The representation for each word depends on the entire context in which it is used.

- ***Deep*:** The word representations combine all layers of a deep pre-trained neural network.
- ***Character based*:** ELMo representations are purely character based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training.

5. Pre-Processing

The Data (Reviews) was scraped and hence contained html elements which were removed or processed. Also the reviews were tokenized and stop words were removed. The reviews were assigned positive, negative and neutral class.

6. Results

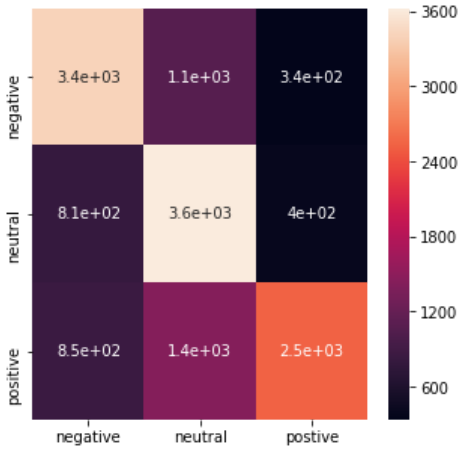
The Classes are 0 (negative), 1 (neutral) and 2 (positive).

TextCNN with Glove Embeddings:

Embedding Vector dimension: 50

class	**precision**	**recall**	**f1-score**	**support**
0	0.67	0.71	0.69	4829
1	0.59	0.75	0.66	4829
2	0.78	0.53	0.63	4829

Accuracy: 66%.



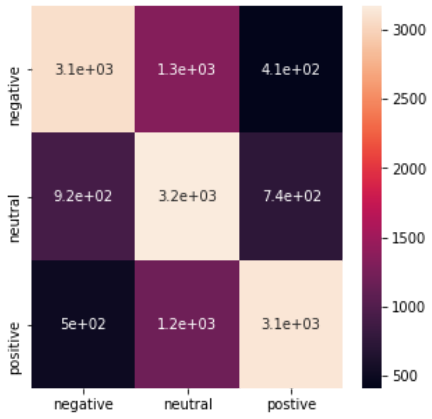
Heatmap of the Results

TextCNN with Word2Vec Embeddings:

Embedding Vector dimension: 300

class	precision	recall	f1-score	support
0	0.68	0.64	0.66	4829
1	0.56	0.66	0.60	4829
2	0.73	0.65	0.69	4829

Accuracy: 65%.



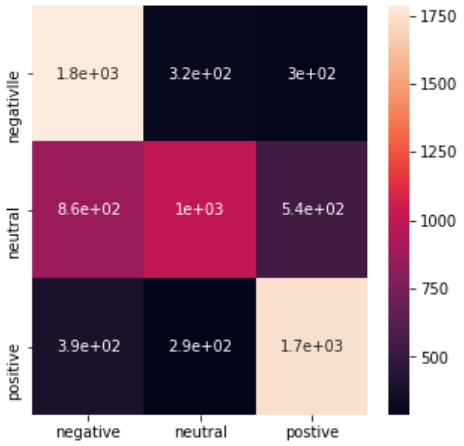
Heatmap of the Results

TextCNN with ELMo Embeddings:

Embedding Vector dimension: 1024

class	precision	recall	f1-score	support
0	0.59	0.74	0.65	2413
1	0.62	0.42	0.50	2405
2	0.67	0.72	0.69	2425

Accuracy: 63%.



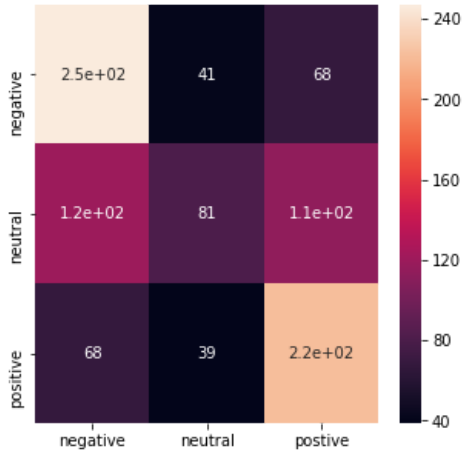
Heatmap of the Results

RCNN with Glove Embeddings:

Embedding Vector dimension: 50

class	precision	recall	f1-score	support
0	0.57	0.69	0.62	356
1	0.50	0.26	0.34	315
2	0.55	0.67	0.61	329

Accuracy: 55%.



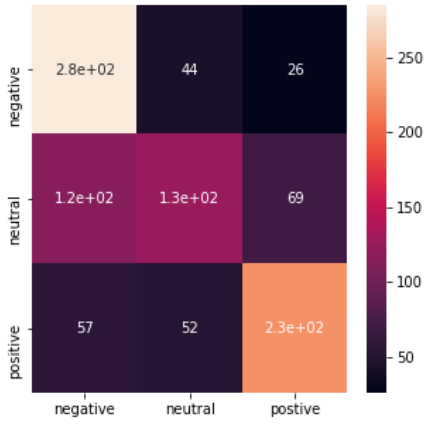
Heatmap of the Results

RCNN with Word2Vec Embeddings:

Embedding Vector dimension: 300

class	precision	recall	f1-score	support
0	0.62	0.80	0.70	355
1	0.57	0.41	0.47	310
2	0.70	0.67	0.69	335

Accuracy: 64%.



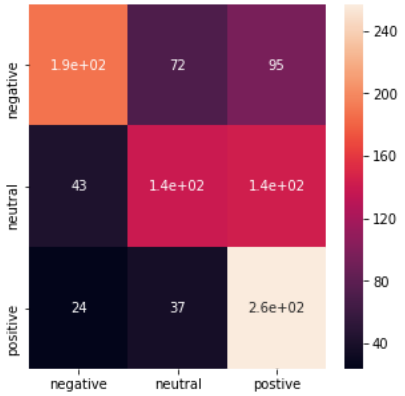
Heatmap of the Results

RCNN with ELMo Embeddings:

Embedding Vector dimension: 1024

class	precision	recall	f1-score	support
0	0.74	0.53	0.62	355
1	0.56	0.43	0.49	327
2	0.52	0.81	0.63	318

Accuracy: 58%.



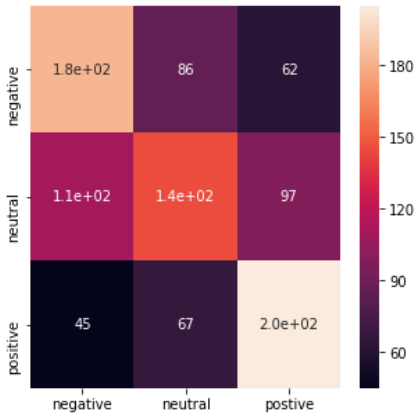
Heatmap of the Results

Seq2seq with Glove Embeddings:

Embedding Vector dimension: 50

class	precision	recall	f1-score	support
0	0.54	0.65	0.55	331
1	0.49	0.41	0.49	352
2	0.56	0.65	0.60	317

Accuracy: 53%.



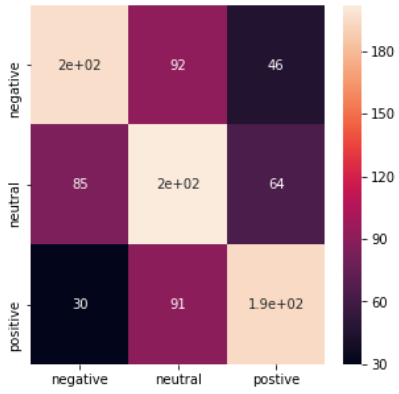
Heatmap of the Results

Seq2seq with Word2Vec Embeddings:

Embedding Vector dimension: 300

class	precision	recall	f1-score	support
0	0.63	0.59	0.61	334
1	0.52	0.58	0.55	351
2	0.64	0.62	0.63	315

Accuracy: 59%.



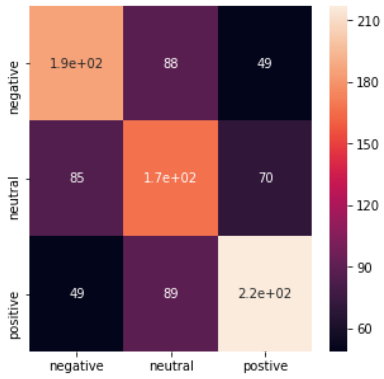
Heatmap of the Results

Seq2seq with ELMo Embeddings:

Embedding Vector dimension: 1024

class	precision	recall	f1-score	support
0	0.58	0.58	0.58	323
1	0.49	0.52	0.50	322
2	0.65	0.61	0.63	344

Accuracy: 57%.



Heatmap of the Results

7. Ablation Analysis

1. We observed high accuracy for unbalanced than balanced data due to data being skewed, with only 9 % of whole labels in class 1.
2. Training time model were in the order TextCNN < RCNN <<< Seq2Seq with attention.
3. The order of training time with respect to embeddings was observed to be Glove(50) < Word2Vec(300) < Elmo(1024). This can be attributed to the increasing length of vectors.
4. We observe that F1- score of RCNN is higher than TextCNN in negative and positive sentiment predictions. This can be attributed to the sequential structure of RCNN which is not present in TextCNN.
5. Also we observe that using context embeddings like ELMo didn't increase the accuracy as Elmo doesn't incorporate medical context.
6. Words that were given high attention by the Attention model with corresponding counts of sentences in which they occurred.

Negative		Neutral		Positive	
bleeding	98	feel	12	love	374
pain	80	started	8	life	367
my	65	bleeding	6	my	211
feel	61	like	6	works	209
bad	56	eat	39	great	182
horrible	47	started	34	love	89
		UNK	34	years	87
		bad	34	great	69

8. References

[1] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, October 25-29, 2014, Doha, Qatar.*

[2] Siwei Lai, Liheng Xu, Kang Liu, Jun Zhao. Recurrent Convolutional Neural Networks for Text Classification. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence 2015.*

[3] Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. Neural Machine Translation by jointly learning to align and translate. *Published as a conference paper at ICLR 2015.*

[4] Jeffrey Pennington, Richard Socher, Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[5] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems 2013.*

[6] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. Deep contextualized word representations. In *NAACL 2018 arXiv: 1802.05365.*

[7] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality.