Research Paper Summary/Notes

Enriching Word Vectors with Subword Information, Facebook.

Laxman Singh Tomar

October 2019

Research Paper Summary/Notes

Enriching Word Vectors with Subword Information, Facebook.

Laxman Singh Tomar

October 2019

Contents

Dedication

I would like to dedicate this work to teachers whom I've never met in real but their valuable contribution have made this work possible.

Acknowledgement

(I encountered several resources which helped me in getting through this work)

I would like to express my sincere gratitude to:

- Tomas Mikolov et. al.
- Adrian Colyer

Abstract

Continuous word representations aka word vectors have proven to be quite successful in a wide array of NLP tasks. Popular models don't take morphology of words into account and hence are not well enough for languages having huge vocabulary and rare words in them. We're introducing a new approach based on the skip-gram model where each word is represented as a bag of character n-grams. It is fast and works well on words which are not part of the training data. Our vectors outperform previous architectures on various tasks to achieve state-of-the-art.

Keywords: Skip-gram; Character n-grams; Subword Model

Chapter 1

Introduction

For the last two decades in Natural Language Processing, many approaches to calculate vector representations of words aka word vectors were introduced. Although, the problem with them was, it was computationally expensive to calculate word embeddings with them. And that was until in 2013, T. Mikolov came up with a simpler and efficient way of calculating word embeddings using a computationally cheaper model: Word2Vec with skip-gram or CBOW approach[1].

Prior work in vector representations of words and phrases ignores the characters within the words, which means that these models ignore regularities such as prefixes and suffixes, roots, and compound words. Intuitively, it should be possible to build a pretty good representation of many words based on morphology, even those that don't show up in the training corpus. It should also be possible to gain model accuracy by exploiting similarities across verb conjugations and noun declensions, especially for strongly morphological languages like Turkish or Russian.

In this paper, we propose to learn representations for character n-grams, and to represent words as the sum of the n-gram vectors. Our main contribution is to introduce an extension of the continuous skip-gram model[2], which takes into account subword information. We evaluate this model on nine languages exhibiting different morphologies, showing the benefit of our approach.

1.0.1 Related Work

In recent years many methods to take morphological features into account such as: Factored NLMs where words are represented as set of features, Composition functions to derive representation of words from morphemes, Joint-learning of embeddings for words characters, Character-4-grams through SVDs, Character level models for NLP and CNN trained over characters which were applied to POS-tagging.

1.1 Motivation

Prior approaches didn't take morphological features of words into account and hence were inadequate for languages having rare words and huge vocabulary.

1.2 Objectives

To propose an extension to the existing skip-gram model to learn representations of character n-grams and to take subword information into consideration.

Chapter 2

Methods

2.1 General Model

Briefly reviewing the continuous skip-gram model[1]: Given a word vocabulary size of W where each word is identified by it's index $w \in 1, ..., W$, the goal is to learn a vector representation for every word w. Word Vectors are trained to predict the words which appear in their context.

The approach is simple: we setup a dictionary with index for each word. Next, we define a window size (say, 2) which gives us context words for every target word. The objective of the skip-gram model is to maximize the average log probability:

$$\frac{1}{T} \sum_{c \in C_t} \log p(w_c|w_t)$$

where the context C_t is the set of indices of words surrounding word w_t. This term signifies the probability of w_c being the context word for word w_t. The values here are just word vectors calculated beforehand.

Let's consider that we are given a scoring function which maps pairs of (word, context) to produce a score in (i.e. single real dimensional space) to signify the numerical score of context between target word and context word. Further, we can define the probability of a word occuring as a context word using softmax function as follows:

$$p(w_c|w_t) = \frac{e^{s(w_t,w_c}}{\sum_{j=1}^{W} e^{s(w_t,j)}}$$

But this probability distribution only considers the probability of occurrence of a single context word to that of the whole vocabulary. Thus, this cannot be used to define our objective function to train the model.

The problem of predicting context words can be seen as a set of binary classification problems. The goal is to independently predict the presence (or absence) of context words. Therefore, we use the binary logistic loss to get the following negative log-likelihood as the objective function:

$$log(1 + e^{-s(w_t,w_c)}) + \sum_{n \in N_{t,c}} \log(1 + e^{s(w_t,n)})$$

where $N_{t,c}$ is a set of negative examples sampled the vocabulary. By denoting the logistic loss function: $l \mapsto log(1 + e^{-x})$, we can re-write the objective function as:

$$\sum_{t=1}^{T} \left[\sum_{c \in C_t} l(s(w_t, w_c) + \sum_{n \in N_{t,c}} l(-s(w_t, n)) \right]$$

A natural parameterization for the scoring functions between a word w_t and a context word w_c is to use word vectors. Let us define for each word w in the vocabulary two vectors u_w and v_w in R^d. These two vectors are sometimes referred to as input and output vectors in the literature. In particular, we have vectors u_{wt} and v_{wc}, corresponding, respectively, to words w_t and w_c. Then the score can be computed as the dot product between word and context vectors as $s(wt, wc) = u_{wt}^T v_{wc}$. The model described in this section is the skipgram model with negative sampling[2].

Clearly, the current approach only considers the neighboring context words for calculating the features and completely ignores the structure of the word itself.

2.2 Subword Model

To address this issue, we represent the word "w" as a bag of all possible character n-grams in the word. The word is padded by a set of unique symbols like the angled brackets: $< WORD >$. This helps in distinguishing the suffixes and prefixes from the rest of the character sequences. We also add the complete word w itself in the set of its n-grams, to learn a representation for each word (in addition to character n-grams).

Say, for example n = 3 the word "*where*" will give us the following n-grams bag represented as G_w:

$$G_w = \Big[< wh, whe, her, ere, re >, < where > \Big]$$

Note that the sequence $< her >$, corresponding to the word her is different from the tri-gram her from the word where. In practice, we extract all the n-grams for n greater or equal to 3 and smaller or equal to 6. This is a very simple approach, and different sets of n-grams could be considered, for example taking all prefixes and suffixes.

Suppose that you are given a dictionary of n-grams of size G. Given a word w, let us denote by $G_w \subset \{1, ..., G\}$ the set of n-grams appearing in w. Every character sequence g in the n-grams bag is denoted by a vector representation z_g. Word vector $< w >$ is given by the sum of the vectors of all the n-grams in that word. We also change the context score function to:

$$s(w, c) = \sum_{g \in G_w} z_g^T V_c$$

It allows the sharing of representations across words, thus allowing to learn reliable representation of rare words. This way, the subword information is utilized in the learning process of calculating word embeddings.

This model can be memory-wise costlier. So, we used Fowler-Noll-Vo hashing function (FNV-1a variant) to hash the character sequences. Ultimately, a word is represented by its index in the word dictionary and the set of hashed n-grams it contains.

Chapter 3

Results

3.0.1 Optimization

Given, the negative log-likelihood as the objective function before, we can optimize to minimize it (and hence maximize the likelihood) by an optimization method. We have used Stochastic Gradient Descent with Linear Learning Rate decay of step size:

$$\gamma_0(1 - \frac{t}{TP})$$

where T is training set containing words, P is number of passes over data, t is time and γ_0 is fixed parameter for all our experiments. The same optimization method has been used by Mikolov et al [2] in their word2vec model.

3.0.2 Implementation Details

We have tried to maintain maximum similarity between our approach and Mikolov et. al.'s approach[2]. Word vector size=300; Negative Sampling Size=5 samples per positive sample and Word rejection criteria = If the word occurs less than 5 times in the corpus, it is removed from the dictionary.

This implementation is 1.5x slower than Mikolov et al's SkipGram[2] implementation.

Sentiment Analysis was done over two datasets: Stanford sentiment treebank database in which each example was having a single sentence and IMDB database in which each example had several sentences. We also tested our method on an information retrieval task, where the goal is to decide if a document should be retrieved given a query.

3.1 Experiments

Datasets that we've taken up: Wikipedia dumps in nine languages: Arabic, Czech, German, English, Spanish, French, Italian, Romanian and Russian.

3.1.1 Human Similarity Judgement

For evaluating the quality of the vector representations, we're computing Spearman's rank correlation coefficient between human judgment and the cosine similarity between the vector representations.

Chapter 4

Discussion

4.1 Discussion

4.2 Conclusions

List of Figures

List of Tables

Bibliography

[1] Tomas Mikolov, G. C., Kai Chen & Dean, J. Efficient estimation of word representations in vector space. *ICLR* (2013).

[2] Tomáš Mikolov, K. C. G. S. C.-r., Ilya Sutskever & Dean., J. Distributed representations of words and phrases and their compositionality. *NIPS* (2013).

Appendix A

Appendix A

Appendix B

Appendix B