TP3-IFT6285 Traitement Automatique des Langues Naturelles

Gustavo Alonso Patino Ramirez

Abstract: In this work we use the word2vec model implemented in the *gensim* library of *python* to transform the words of a corpus into vector entities. The training of the word2vec model was carried out in two ways: Initially the word2vec model was trained using the *train_ post.csv* corpus of words and subsequently a richer pre-trained model with a larger corpus from Google News dataset was used. The objective of the training in the two approachs is to determine the closest neighbors to the most frequent words of the corpus *train_ post.csv*. The proximity criterion was defined based on the similarity between the words. Subsequently, the relationship between the most frequent words of the corpus and its closest neighbors was analyzed using different tags: (MORPHO) for morphological relation, (HYPHO) for hyponym, (SYN) for synonyms, (ANTO) for antonyms, (PARTOF) for meronyms and holonyms and (COHYPO) for cohyponyms.

1. Introduction

Word to vector (word2vec) is a model presented by [1], [2] that embeds words in a vector space using a shallow neural network. There are two versions for the word2vec both implemented in the *gensim* library of *python*: the skip gram model and continuous bag of words (CBOW). The Skip-gram model of Word2Vec, takes pairs of words obtained from the text and trains a neural network of one hidden layer, whose input receives a single word, in this way the hidden layer provides a predicted probability distribution of words close to the input. A one-hot coding goes from the projection layer to the hidden layer, and it is precisely these projection weights that are interpreted as the word embeddings. So, if the hidden layer has n neurons, this network will give us word embeddings of n-dimensional words. Being the number of neurons (or in other words the size of the vector space) a hyperparameter that can be tuning in the training process. On the other hand, the CBOW model is very similar to the skip gram model, because it also uses a one hidden neural network layer. Nonetheless, the main difference is that the training process uses the average of multiple input words, instead of a single word as in skip gram. Similarly, projection weights are interpreted as the word embeddings.

2. Results

2.1. Defining the neighbors

We work with the corpus *train_ posts.csv* used in TP2. This file contains 512629 tagged comments. In the first step, we train a word2vec model using the *gensim python* library and the aforementioned corpus. The word2vec class implemented in *gensim.models* requires three training hyperparameters: *sentences*: The phrases of the corpus, *min_ count*: number with which the internal dictionary is constructed, discarding words that appear in the corpus less than *min_ count* times, and size: number of dimensions of the vector space on which word2vec will map each of the word vectors. The training process may become too slow due to the computational cost, therefore, we only built the vector representations of the words seen in the corpus at least 5000 times. This number was chosen according to the frequency table provided on TP2, where we conclude that the first 1000 most viewed words in the corpus appear at least 8000 times. The size of the vector space was defined by 500. After training the model, we can use the *most_ similar* method which allows to find the most similar set of words. Initially, the *most_ similar* method of *gensim* requires as parameters the word to which we will look for the closest words, and *topn* which defines the size of the set of closest words. In our case, we implemented a method which allows us to find the closest words using a distance of similarity instead of *topn*, as a radius of similarity we define the value of 0.4. In other words, all the words whose similarity is greater than 0.4 were stored in a dictionary, whose *key* is the word and the values are the number of words within the radius of similarity and the set of words. A small sample of the results can be seen in the Table.1. The closest word are written in decreasing order of similarity.

Word	number neighbors	closest words
Day	13	week, weekend, time, evening, afternoon, night, morning, monday, today
Love	6	loving, loves, loved, hate, trust, miss
Home	5	back, house, lunch, dinner, hospital
Bad	5	good, terrible, horrible, weird, sad
Computer	3	internet, desk, machine

Table 1: Closest words using similarity. Model trained with the corpus *train_ posts.csv*, vector representation using word2vec

Unfortunately the trained model is quite limited. Since the training corpus is restricted to a universe of 512629, some words have very few neighbors within the radius of similarity. In order to use a richer corpus, we use a pre-trained model in a Google News dataset corpus that contains approximately 3 million words and phrases. This model is already trained in the *gensim* API and takes a few minutes (about 4 minutes) to load. After loading the model, we can use the code implemented for the previous item and obtain the vector representation of the most viewed words in the *train_ posts.csv* corpus, again we made the vector representation and calculate the closest neighbors of the words seen in the corpus at least 5000 times. The radius of similarity was defined again as 0.4. In Table.2, we can see some results for the same words shown in Table.1, noticing a great difference in the richness of the model. A more complete table was submitted to studium with the name *voisins.csv*. The head of the document is defined by: *word*, *number neighbors* and *closest words*. In *closest words*, we print just the first 10 words of the set of neighbors in order to reduce the file size. Furthermore, the first 10 neighbors words are more informative since the words in *closest words* are defined in descending order of similarity.

Word	number neighbors	closest words
Day	58	week, days, morning, month, hours, afternoon, hour, weekend
Love	500	loved adore loves passion hate loving Ilove affection undying_ love
Home	40	house, Superfast_ WiFi, homes, Home, residence, bedroom, apartment, lakehouse
Bad	211	good, terrible, horrible, Bad, lousy, crummy, horrid, awful, dreadful horrendous
Computer	500	computers, laptop, laptop_ computer, laptop_ computers, PC

Table 2: Closest words using similarity. Model trained with the corpus Google News dataset , vector representation using word2vec

2.2. Analysis of the neighbors

In this section we make a qualitative analysis of the relationship between a word and its closest neighbors. The relationships explored are as follows:

- Morphological (MORPHO): When words have the same root. To find this relationship we lemmatized the word and its neighbors by verb, adjective, noun and adverb and observe the coincidences.
- Hyponym (HYPHO): When the neighbor is a specific concept of the word. We import the *wordnet* class from *python* and define the words and neighbors as *synsets* objects ($word = wordnet.synsets(word)[0]$) with this definition we can obtain the set of hyponyms of the word and see if any neighbor (defined as a synset objects too) is found within the set of hyponyms.
- Synonyms (SYN): After defining the word and the neighbors as synset objects, we can obtain the lemmas and thus obtain all the synonyms of the word in all the contexts, later it is observed if any of the neighbors coincide with any of the synonyms of the word.
- Antonyms (ANTO): We proceed in a similar way to synonyms up to the point of obtaining the word lemmas, later we take advantage of the fact that wordnet lemmas has the method lemma.antonyms (), therefore we can see if any neighbor element matches the antonyms of the word.
- Meronyms and Holonyms (PARTOF): Meronyms and Holonyms represent the part-whole relationship. The meronym represents the part and the holonym represents the whole. We define the word and its neighbors as synset objects and obtain the Meronyms and Holonyms and thus we can detect if the neighbor is a part of (PARTOF) the word.

- Cohyponyms (COHYPO): In this case we look at the set of hypernyms (the general concept of a word) of the word and its neighbors, in case the word and some neighbor have hypernyms in common we can define them as cohyponyms

In Table.3 we can see the relationship between the words of the corpus shown in Table.2 and its closest neighbors, as defined above.

Word	**number neighbors**	**closest words**
Day	58	week[COHYPO], days[MORPHO], morning[COHYPO], month[COHYPO] hours[PARTOF], afternoon[COHYPO], hour[PARTOF], weekend[COHYPO]
Love	500	loved[MORPHO], loves[MORPHO] passion[SYN] hate[ANTO] loving[MORPHO] Ilove affection[COHYPO] undying_ love
Home	40	house[SYN], Superfast_ WiFi, homes[MORPHO], Home[MORPHO] residence[COHYPO], bedroom[COHYPO], apartment[COHYPO], lakehouse
Bad	211	good[ANTO], terrible[SYN], horrible[SYN], Bad[MORPHO], lousy, crummy
Computer	500	computers[MORPHO], laptop[HYPO], laptop_ computer[HYP0] laptop_ computers[HYPO], PC[HYPO]

Table 3: Relation between the words and their neighbors

A more complete table was submitted to studium with the name *annotations.csv*. The data is organized in the order: word,number of neighbors, label words.

3. Conclusion

It was presented a way of embedding the word of a corpus using the word2vec method. Initially, the word2vec model was trained with a corpus of 512629 texts, and we retained the vector representation of the most frequent words seen in the corpus, using this vector representation, we can calculate the similarity with other words and thus define the set of closest neighbors for the most frequent words in the corpus. Since the training corpus is not very robust, some words had few neighboring words within an arbitrary radius of 0.4, we did not want to extend the radius because there is a risk of defining as neighboring words that really do not have much relation. For this reason, we decided to use a pre-trained model in a much larger corpus using the Google News dataset, and calculate again the neighboring words within a radius of 0.4, observing a considerable increase in the number of neighbors since the training corpus is richer. Finally, we qualitatively define the relationship between each word and its closest neighbors using the concept of hyponyms, synonyms, antonyms, meronyms and holonyms, and morphological relationships.

References

[1] T. Mikolov, K. Chen, G. Dean, Efficient estimation of word representations in vector space., arXiv:1301.3781.

[2] T. Mikolov, K. Chen, G. Dean, Distributed representations of words and phrases and their compositionality., Adv. Neural Inf. Process. Syst. 26 (2013) 3111–3119.