DEEP LEARNING

MASTER IN ARTIFICIAL INTELLIGENCE

Universitat Politècnica de Catalunya

PRACTICAL WORK 3: WORD EMBEDDINGS AND CLUSTERING IN IMDB MOVIE REVIEWS

Albert Espín

April 24th, 2019

Table Of Contents

1. Introduction

Identifying the meaning of words in texts is one of the main goal in the field of Natural Language Processing (NLP), since it is a stepping-stone of a large variety of tasks, such as automatic summarization, question answering, or, hypothetically, the acquisition of world knowledge by an artificial agent. Different approaches exist to represent the meaning of words, such as knowledge bases resembling dictionaries (such as WordNet) where words are mapped to different senses, disambiguated by the context of the words (e.g. the sentence where the words appear).

In recent years, a different technique has gained great popularity due to its effectiveness when applied to the aforementioned NLP tasks: word embeddings. Theoretically, a text can be represented in a high-dimensional space of sparse binary features with as many dimensions as different words are found in the text, but just with this it is not possible to induce any semantic information of the words. Word embeddings are models that represent words in a lower-dimensional real-valued space, in such way that words that are close to each other are semantically similar, since the model has learned that they appear in analogous, replaceable contexts.

One of the interesting applications of word embeddings is the possibility of dividing words in semantic groups by applying a clustering algorithm to the word vectors (such as K-Means). This procedure can help to identify semantic blocks of words within the domain of the explored corpus, and the results can be mapped to a two-dimensional space for visualization of the results.

The aim of this work is twofold. Firstly, to build a Word2Vec-powered word embedding model for the well-known Internet Movie Database (IMDB) reviews data set [4], testing a large number of parameter configurations. The embeddings produced by each configuration are used to form word clusters that are evaluated using internal clustering metrics. Secondly, the word embeddings model that is determined to have the best-separated clusters is used to visualize a set of key words in the domain of the data set (movies), along with the words that the model considers of higher semantic similarity to them. The objective is to prove in a visually simple way how the model is able find relations between words, making it obvious even for an audience not familiar with machine learning. The developed code has been uploaded to a Github repository [1].

2. Data set and preprocessing

The IMDB data set contains 50000 plain-text movie reviews, half of which (25000) are labeled as positive and the other half are negative ones; this labeling information, however, is not used to build the embeddings, neither for clustering, which is an unsupervised task.

Word2Vec learns embeddings from lists of words (or tokens) that define sentences, not whole texts such as the reviews of the raw data set. For this reason, an essential preprocessing step is to divide each review in sentences, which is done by using the sentence tokenizer of Python's Natural Language Toolkit (NLTK). This utility locates sentence boundaries (most of the times in locations where dots are identified as periods, not part of abbreviations or other linguistic units). However, before locating the sentence boundaries, the Beautiful Soup package is used to remove the HTML tags present in some of the reviews, such as "<br>". Once the lists of plain-text sentences have been obtained (532125 in total), NLTK's word tokenizer is used to separate the tokens, producing a list of tokens for each sentence.

To appropriately learn word embeddings the only tokens that should be used are words, so non-alphanumeric tokens are discarded. For instance, "%" is not an alphanumeric character, that the tokenizer conveniently separates from words, e.g. "3%" would be divided into two tokens, "3" and "%", with the second one being discarded. It may be argued that some of these characters may actually give semantic information (such as specifying that a number is a percentage, in the previous example), but in many other cases they represent noise (in informal opinion texts it is often to find strings like "!!", "?!", or even constructions such as "sh*t"). As an exception, tokens whose only non-alphanumeric character is a hyphen are accepted, since useful combinations of words can be expressed with this character, such as some adjectives (e.g. "well-known") or compound surnames (e.g. "Zeta-Jones").

Additionally, NLTK's 179 English stop-words are removed. Stop-words are common words in texts that do not provide semantic information by themselves (e.g. "a", "the" or "we"). Initially, this action was going to be avoided, since it was considered that deleting stop-words would artificially alter the context of sentences were embeddings are learned, and some sequential information would be lost. However, after testing different configurations of embeddings for word clustering (explained in Chapters 3 and 4) with the presence and absence of stop-words, it was seen that removing them made clusters more meaningful, by clearly grouping words that a human would consider similar (e.g. "film" and "movie"; "actor" with "actress" and "performant"). When using stop-words to learn the embeddings, the later clusters were more noisy, mostly combining words lacking a clear relation (e.g. "10" with "pitch", "sea" and "hehe"). The justification of this phenomenon is the polyvalent role of stop-words in sentences (e.g. "the" or "a" can be followed any noun or noun phrase), contributing to make the model believe that any two words that commonly follow a stop-word are replaceable and similar, which often is not true. Even when a reasonably high downsampling coefficient was used to mitigate the impact of common words when learning the model (e.g. 0.05), the clusters were not as semantically good as those built without using stop-words. In the end, the total number of accepted words of the sentences is 5791488; there are 129702 unique words, which become the vocabulary for the word embeddings model.

3. Experimental methodology

Different configurations of word embeddings models have been tested. Each configuration is defined by assigning a certain value to every parameter in the following list, that shows the tested values[1] (all the combinations have been tested, for a more thorough evaluation of their influence):

- Number of dimensions of the real-valued vector that defines a word in the embedding space. The higher it is, the more precisely can words be defined (up to a point where too much irrelevant, noisy information is considered), at a higher computational cost, also requiring more storage capacity. The tested values are 200 and 300 dimensions.
- Window size, which is the maximum number of words surrounding a word (as a context, in the same sentence) being analyzed by the model at any particular moment during the learning process. Therefore, the higher it is, the broader the context that is taken into account when studying words in sentences, but a too large window size can incorporate far-away words, that may be not relevant for the current observation. The tested values are 5 and 10 words.
- Minimum word count, the minimum number of times that words need to appear in the corpus of sentences to be at all considered for learning. A very low value such as 1 can be suitable to include very uncommon words that can enrich the vocabulary, but is also sensitive to adding typos. However, many typos are far more common that any normal minimum word count, such as "Im" instead of "I'm" or "I am", so a conservative value does not solve this problem completely. Raising the minimum count too much can cause the loss of a significant number of words that are interesting to learn despite of

1 The values tested for each parameters have been set based on those that were reported to yield the best results in experiments of other researchers that have also applied Word2Vec to the IMDB data set [2]. Additionally, other alternative values that seemed reasonable to compare with the mentioned ones are also tested.

being somewhat rare (such as the names of secondary actors). The tested values are 1 and 20.

- Model architecture, that can be based on skip-grams or continuous bag of words (CBOW). These techniques differ in how the internal model learns the embeddings when given an individual word and its context (window). While the skip-gram model is trained to estimate how likely it is to have a certain context given a word, the CBOW methodology consists in estimating the probabilities of finding a word given the context. Both architectures have been tested.
- Output layer function. Both hierarchical softmax and negative sampling have been tested. Whilst the normal softmax function would compute the probability of every word of the vocabulary to be the word being predicted, hierarchical softmax uses a binary tree to reduce the number of computations to a logarithmic complexity of the vocabulary, obtaining a substantial boost in time performance. Negative sampling uses a noise probability distribution to compare, using logistic regression, real data with noise, and by learning this contrast refine its capability of predicting the correct words.
- Initial learning rate of the model, that gets reduced as the learning process progresses. Both 0.025 and 0.1 values have been tested.
- Downsampling coefficient, used to reduce the impact or relevance in the model of common words, proportional to their frequency in the corpus. Preliminary tests in a small number of runs have shown that 0.001 produced better results than 0.01, that was seemingly reducing too much the influence of common words rich in meaning, such as the adjectives "good" and "bad", to the extent of almost ignoring them. Therefore, only 0.001 was used in the final set of runs.

The set of 64 word embedding models, based on the configurations of the mentioned parameters, were used to divide the words in clusters. For this task, the K-Means algorithm was used, with the K-Means++ initialization, that is well-known to produce better results (in general) than random initialization, by placing initial centroids far away from each other, to try to avoid difficulties in convergence in later iteration (that can make the algorithm getting trapped in poor local optima). After testing some different values for the number of clusters in a few number of fixed embedding configurations, the most informative clusters with this data set were obtained by dividing the data into 10 groups. It was not clear what most clusters shared in common when using a lower (2 or 5), since very different words were together. Higher numbers (20) created many irrelevant groups (e.g. different prepositions, conjunctions and interjections, mixed together).

Since the vocabulary size of the embeddings is considerable (129702) and K-Means has quadratic memory complexity (due to the representation of the matrix with the distance between every pair of word embeddings), batch-based K-Means is used, with a batch size equal to half of the vocabulary size, as a compromise between time efficiency and storage feasibility. The computation is multi-threaded, dividing it in 8 threads, to make the process faster. Early stopping is used too, in such way that if there are 5 iterations without meaningful changes in the cluster centroids (not below a threshold), the algorithm stops. Otherwise, it runs up to a maximum of 50 iterations (in practice it converges in 10 to 12 iterations most times).

The clusters of every word embedding model are evaluated using an internal cluster metric; external ones are not feasible since they would imply having a labeled version of the vocabulary classifying the words in 10 semantic groups. The metric that has been used is the Calinski-Harabaz score, also known as variance ratio, since it is the result of dividing the variance between the elements of a same cluster by the variance between the points of different clusters. It is considered to be better the lower it is, since low values imply that the elements are more similar to those in their cluster than to those of other clusters. A very high value can lead to think that instances are poorly clustered, since they are very similar to those of other groups. The Calinski-Harabaz score is therefore used to determine how suitable each embedding model is to achieve high separability between clusters (semantic groups of words in this case), given the specified K-Means clustering described in the previous paragraphs. Therefore, this metric should not be considered as a general measure of the goodness of an embedding model for any arbitrary task, but only for this clustering task, and only if the objective is to minimize the intra-cluster variance and maximize the inter-cluster variance. As a last step, the model minimizing the variance ratio is used to visualize key words of the movies domain in two dimensions, which requires to reduce the dimensionality of the embeddings in from 300 to 2, which is achieved by using the t-distributed Stochastic Neighbor Embedding algorithm (t-SNE), taking an example as base code [5].

4. Results and Discussion

4.1. Word embedding parameters and variance in word clusters

Table 1 shows the variance ratio (Calinski-Harabaz score) obtained by each of the 64 tested configurations of models that learned word embeddings, sorted in ascending order of variance ratio. That is to say, those configurations in the first positions depict the best parameter values for minimizing the intra-cluster variance and maximizing the inter-cluster variance of the word embeddings (and by extension of the meanings of the words that they represent), for this particular clustering problem.

By analizing the first positions of Table 1, it can be seen that the combination with the lowest variance ratio has a value of 72.853, more than 100 times lower than that of the highest value, 7707.028. To determine which value for each parameter is better to minimize the variance, the top 7 configurations with less variance are analyzed (all bellow the value of 100, so with less than 30% of difference between the seventh and the first).

While the first 3 models use 300 dimensions for the real-valued features describing the embeddings, the next 4 use 200. It can be said that having 300 dimensions allows to represent with more detail each embedding, obtaining more information for precise clustering, but given that models with 200 dimensions also score good, it does not seem a determinant parameter. A similar situation happens when it comes to the size of the context window for words. In the top 7 models, 4 use 5 words for the window, while 3 use 10 words, including the first model, but not the next two. Therefore, it can be induced that both values are almost as good for the problem. While 10 words capture more information, 5 focus in the most closely related context, at the expense of being short-sighted. For these two parameters, the worst models also combine the different values, there is not a single one always present in models with high variance ratio. There are other parameters whose influence (at least with the tested values) is also not clearly beneficial or detrimental, in terms of variance ratio: the model architecture (using CBOW or skip-grams; while CBOW appears in the first 3 models, both options alternative very often throughout the rest of configurations) and the output layer functions (hierarchical softmax, that appears both in the few first and last models, or negative sampling, but they alternate in the rest).

When it comes to the minimum count of words, it can be observed that 20 (present in almost the first half of the models) occurrences guarantees much lower variance ratio than 1 (present in all the worst models). It can be justified by the fact that 1 occurrence is not filtering a high number of words for which there is not enough information (context examples) to infer its meaning and cluster it with a reasonable accuracy. The best initial learning rate is 0.1, the value for the first 7 configurations, while 0.025 is present in the worst configurations. This implies that, since the learning rate decreases with time, when the learning rate starts being 0.025 it decreases to a very low value too soon, before having learned the core structure of the embeddings.

Table 1: Tested configurations of word embedding models with the variance ratio of their clusters.

Feat. dim.	Window size	Min word count	Model arch.	Output layer funct.	Learn. rate	Downsampling	Variance ratio
300	10	20	CBOW	hierarchical softmax	0.1	0.001	72.853
300	5	20	CBOW	hierarchical softmax	0.1	0.001	75.131
300	5	20	CBOW	negative sampling	0.1	0.001	79.549
200	10	20	skip-gram	negative sampling	0.1	0.001	80.877
200	5	20	skip-gram	negative sampling	0.1	0.001	90.712
200	10	20	CBOW	negative sampling	0.1	0.001	94.059
200	5	20	CBOW	negative sampling	0.1	0.001	97.836
300	10	20	CBOW	negative sampling	0.025	0.001	100.878
200	10	20	CBOW	hierarchical softmax	0.1	0.001	110.448
300	10	20	skip-gram	hierarchical softmax	0.1	0.001	124.301
200	5	20	CBOW	hierarchical softmax	0.1	0.001	126.938
300	5	20	CBOW	negative sampling	0.025	0.001	138.583
200	10	20	skip-gram	hierarchical softmax	0.1	0.001	140.223
200	10	20	CBOW	negative sampling	0.025	0.001	148.639
300	5	20	skip-gram	hierarchical softmax	0.1	0.001	150.92
300	10	20	skip-gram	negative sampling	0.025	0.001	166.81
200	5	20	skip-gram	hierarchical softmax	0.1	0.001	180.723
300	5	20	skip-gram	negative sampling	0.025	0.001	183.909
300	5	20	skip-gram	negative sampling	0.1	0.001	186.162
200	5	20	CBOW	negative sampling	0.025	0.001	190.222
200	10	20	skip-gram	negative sampling	0.025	0.001	196.715
300	10	20	skip-gram	negative sampling	0.1	0.001	213.967
200	5	20	skip-gram	negative sampling	0.025	0.001	230.922
300	10	20	CBOW	negative sampling	0.1	0.001	236.348
300	10	20	CBOW	hierarchical softmax	0.025	0.001	297.624
300	5	1	skip-gram	negative sampling	0.1	0.001	317.049
200	10	20	CBOW	hierarchical softmax	0.025	0.001	324.665
300	5	20	CBOW	hierarchical softmax	0.025	0.001	362.949
200	10	1	skip-gram	negative sampling	0.1	0.001	374.114
200	5	20	CBOW	hierarchical softmax	0.025	0.001	395.489
200	5	1	skip-gram	negative sampling	0.1	0.001	450.795
300	10	20	skip-gram	hierarchical softmax	0.025	0.001	465.336
200	5	20	skip-gram	hierarchical softmax	0.025	0.001	478.924
200	10	20	skip-gram	hierarchical softmax	0.025	0.001	479.31
300	5	20	skip-gram	hierarchical softmax	0.025	0.001	509.876
300	5	1	CBOW	negative sampling	0.1	0.001	515.817
200	10	1	CBOW	negative sampling	0.1	0.001	599.23
200	5	1	CBOW	negative sampling	0.1	0.001	706.001
300	10	1	skip-gram	hierarchical softmax	0.1	0.001	708.675
300	10	1	CBOW	hierarchical softmax	0.1	0.001	711.213
300	5	1	CBOW	hierarchical softmax	0.1	0.001	788.456
200	10	1	CBOW	hierarchical softmax	0.1	0.001	840.424
300	5	1	skip-gram	hierarchical softmax	0.1	0.001	887.317
200	10	1	skip-gram	hierarchical softmax	0.1	0.001	911.462

200	5	1	CBOW	hierarchical softmax	0.1	0.001	1010.822
200	5	1	skip-gram	hierarchical softmax	0.1	0.001	1043.091
300	10	1	CBOW	negative sampling	0.025	0.001	1190.213
300	10	1	skip-gram	negative sampling	0.1	0.001	1389.958
200	10	1	CBOW	negative sampling	0.025	0.001	1453.398
300	10	1	skip-gram	negative sampling	0.025	0.001	1484.105
300	5	1	CBOW	negative sampling	0.025	0.001	1560.608
200	10	1	skip-gram	negative sampling	0.025	0.001	1582.6
300	5	1	skip-gram	negative sampling	0.025	0.001	1679.73
200	5	1	skip-gram	negative sampling	0.025	0.001	1797.761
200	5	1	CBOW	negative sampling	0.025	0.001	1970.554
300	10	1	CBOW	negative sampling	0.1	0.001	2700.927
200	10	1	skip-gram	hierarchical softmax	0.025	0.001	4935.38
300	10	1	CBOW	hierarchical softmax	0.025	0.001	5037.967
300	10	1	skip-gram	hierarchical softmax	0.025	0.001	5046.942
200	10	1	CBOW	hierarchical softmax	0.025	0.001	5227.157
300	5	1	skip-gram	hierarchical softmax	0.025	0.001	5774.686
200	5	1	skip-gram	hierarchical softmax	0.025	0.001	5789.536
300	5	1	CBOW	hierarchical softmax	0.025	0.001	7342.024
200	5	1	CBOW	hierarchical softmax	0.025	0.001	7707.028

4.2. Topic identification in word clusters

Table 2 shows an extract of the word clusters of the model with the lowest variance. A set of topics can be easily identified, and words that often appear together in similar semantic contexts tend to appear together. For instance, the first cluster contains common foreign words without strong meaning, mainly articles and prepositions (from French, such as "des", "mon" or "au"; from Italian, such as "il"; from German, such as "herr"; from Asian influence, such as "fuji" and "lama"). Another cluster contains common interjections present in informal texts (e.g. "ewww" or "grrr"). There are some clusters about social aspects, such as one cluster about family and daily life ("family", "father", "house", "school"), and another that can be considered conceptually overlapping since it talks about people, mixed with America ("men", "women", "America", "america"), making obvious that the personalization of American ideas in individuals make the topics not separable for the algorithm. The rest of clusters are purely related with the domain of movies, as it would be expected for IMDB: action scenes ("shot", "killer", "car", "fight"), a movies and genres ("movie", "films", "horror"), directing and music ("director", "cinematography", "score", "song"), film ratings ("good", "bad", "well", but also "film", due to being highly linked to the adjectives that qualify films), female roles and stereotypes ("queen", "heroine", "blonde") and actors and actresses, both with common words ("actor", "actress", "cast") and proper names, some of which may be those of fictional characters ("Michael", "James", "Jack"). It can be said that the clusters are remarkably accurate from a semantic perspective, building clusters that make sense, with the mentioned overlaps being reasonable due to the polyvalent nature of some words and small phrases.

Table 2: Word clusters with the embedding model that minimizes the variance ratio.

Cluster main topic	**Words in cluster (extract)**
Common foreign words	les, el, del, en, du, des, un, harp, il, grady, hai, ka, chronic, dil, herr, mon, au, si, fuji, lama
Action scenes	guy, shot, black, head, white, killer, car, fight, blood, etc, body, bunch, monster, rock, cops
Interjections	ewww, gracias, doh, puh-lease, puh-leeze, blech, grrr, bleah, yakitate, hehehe, -ap3-, crikey
America and people	life, us, world, true, american, war, men, women, human, become, evil, history, art, america
Movie	movie, would, see, first, made, movies, watch, films, seen, best, ever, better, say, horror
Directing and soundtrack	director, beautiful, written, style, works, brilliant, directed, score, song, cinematography
Family life	man, old, family, girl, woman, goes, house, father, help, home, friends, couple, school
Film rating	film, one, like, good, even, time, really, much, well, could, people, bad, make, way, think
Female roles	female, sweet, sexy, affair, queen, heroine, blonde, lesbian, obsessed, naive, nurse, princess
Actors and actresses	great, character, cast, young, role, performance, played, actor, play, job, plays, john, excellent, michael, actress, daughter, roles, david, james, robert, talent, career, jack

4.3. Visualization of key words in the similarity space

Finally, the embedding space of the words is visualized in a simplified way in Figure 1 (using t-SNE to reduce the dimensionality from 300 to 2, as explained in Chapter 3). Since the vocabulary is very large, only a set of key words visualized, in different colors, along with the terms with the top 10 most similar meaning to those (according to the model, i.e. using computing embedding similarity with the model with lowest variance ratio). The visual clusters of words make sense, such as "effects" surrounded by "special", "puppetry", "FX", "CGI"; clusters with common numbers ("1", "10", "ten"; also, "007" is linked to James Bond films such as "Octopussy" or "Goldeneye", which was also a popular video game, so it appears next to console names such as "N64" or "PS1" and its game genre, "FPS"); years ("1984", "1965"; "odyssey" is close to years because of the film "2001: A Space Odyssey"). Adjectives are clustered together when similar ("good" and "half-decent" are closer between them than to "awful" or "horrible", which are together). There is a large region of the space occupied by actors (near "actor" and "actresses") and characters, since they are related with many other topics (the films where they participate, such as "Al" and "Pacino", related to movies such as "Scarface" or "Godfather"). It also happens, in a lower space, with directors ("Coppola", "Scorsese" and others appear next to "filmmaker" and "director"). Names and surnames are close when names are uncommon, otherwise the linking is not so clear ("Al" appears next to "Capone"). The similarity groups are meaningful, especially remarkable taking into account the great simplification in distances between elements when moving from 300 to 2 dimensions.

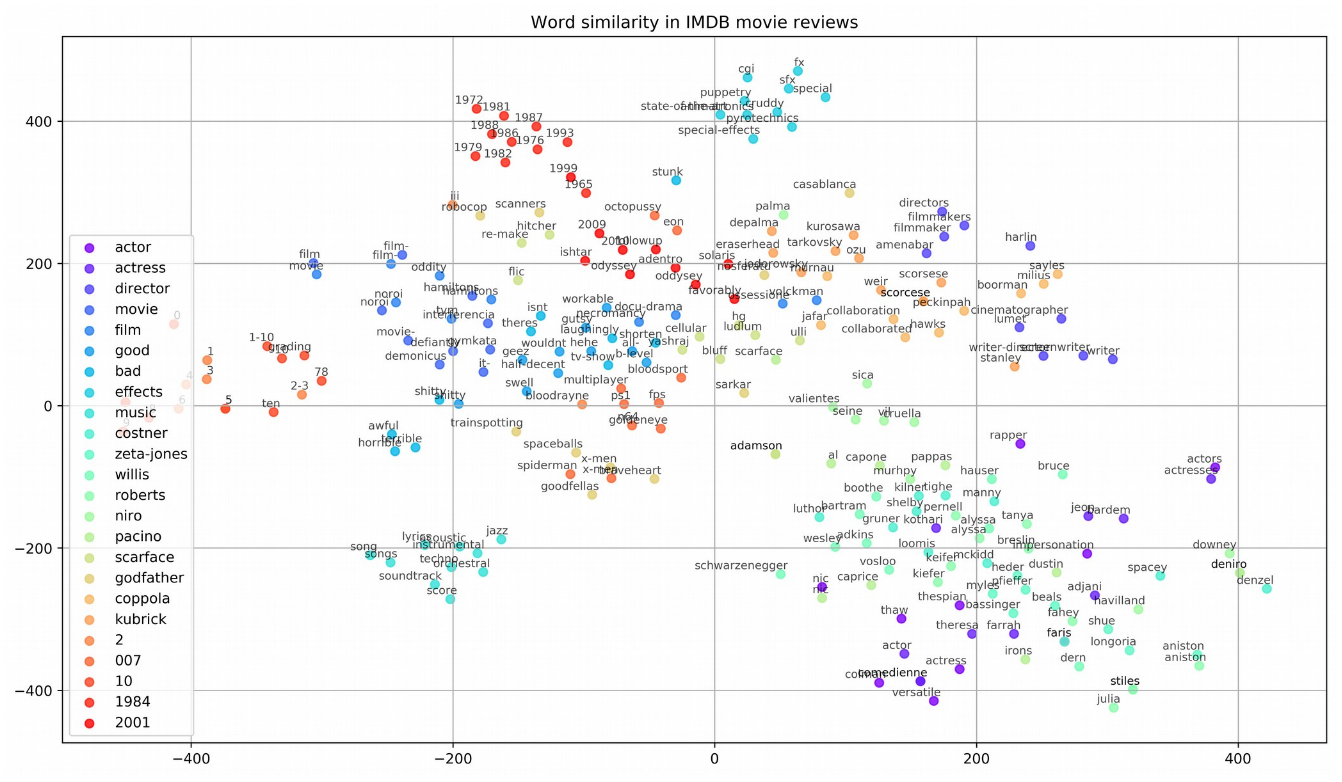

Figure 1: Representation of key words surrounded by their most similar ones.

5. Conclusions and Future work

This work has experimentally shown, by using the particular case of the IMDB, that word embeddings can be applied to learn meaning representations of the words in a corpus. The embeddings can be used to group words in meaningful clusters that separate them by semantic topics.

As future research, it would be interesting to apply the learned models of word embeddings to other tasks, such as sentiment polarity classification in this same data set (e.g. using a deep neural network architecture with fully-connected and convolutional layers, with these embeddings as input), or prediction of word sequences in reviews of other data sets (using these embeddings and a recurrent neural network, for instance based on the Long Short Term Memory architecture or using Gated Recurrent Units). Additionally, other natural language data sets (e.g. Twitter analysis data sets) may be considered to add to the corpus for learning word embeddings for a greater number of words, with more examples as context, potentially enabling more accurate learning of their semantic representation.

6. References

Developed code:
[1] Github repository

Other sources of information:

[2] Kaggle Team. Bag of Words Meets Bags of Popcorn: Introducing Distributed Word Vectors. Kaggle (kaggle.com). Jul. 2015.

[3] Edward Ma. 3 silver bullets of word embeddings in NLP. Towards Data Science (towardsdatascience.com). Jul. 2018.

[4] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011). 2011.

[5] Sergey Smetanin. Google News and Leo Tolstoy: Visualizing Word2Vec Word Embeddings using t-SNE. Towards Data Science (towardsdatascience.com). Nov. 2018.