Abdelwahid Benslimane
wahid.benslimane@gmail.com

NLP: word embedding using the Word2Vec method

I) Introduction

When doing NLP (natural language processing), it is important to understand that language models require a numerical representation of the words that make up a text, as these models are still neural networks that need to perform calculations on the objects they manipulate. In fact, words will be represented in the form of vectors. Various approaches exist to produce a vector representation of words, and one can distinguish between methods that produce a representation of each word in a particular context and those that, by "synthesizing" different possible contexts, develop a unique representation for each word. This is the case, notably, with the Word2Vec method, which I will discuss in this document, but one can also mention GloVe or FastText. Please note that this document will not delve into the implementation details of this method but will describe its outline and present it in broad strokes.

II) Lemmatization and other prerequisites

To construct the classic vector representation of texts, several preliminary steps are generally necessary: segmenting the text into words and punctuation marks, lemmatization, and the removal of "stop words." The latter step aims to eliminate very frequent but non-discriminative words from the text, as they are necessarily present in all texts: prepositions, conjunctions, articles, etc. If the identification of locutions (groups of words that form a single entity) is necessary for the analysis objective, this step should be performed before the removal of stop words.

Lemmas are autonomous units of a language and/or a particular domain (e.g., medicine), with some additional information such as possible forms, including roots, suffixes, and prefixes. The different forms of a word are thus treated as a single lemma. The set of lemmas associated with a given language constitutes what is known as the lexicon of that language. After this operation, a text becomes a sequence of lemmas, all of which will be referred to as words in the following sections.

A lemma can consist of one or more words. Among the lemmas that comprise multiple words, we find compound words and locutions. In French we can for example find nominal locutions and verbal locutions. While compound words are simple to identify, the situation is more complicated for locutions, as their detection sometimes requires a more in-depth analysis. The status of a locution for a group of words can vary from one text to another, which is at least the case, again, in French, for example.

III) Word2Vec

Returning to Word2Vec, this method proposes to exploit the context of words to construct more "refined" vector representations, as the context of a word in a sentence characterizes the word quite well both syntactically and semantically. Word2Vec does this based on very large textual resources and the skip-gram model, whose objective is to find representations that best predict the context of words. More specifically, given a sequence of words $w_1, w_2, \dots, w_T$, the aim is to maximize:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{j=-k, j\neq 0}^{j=k} \log p\left(w_{t+j}|w_t\right)$$

In the formula, k represents the number of words before and after the word of interest, w_t. The overall context width will then be $2k$. This value is arbitrary and is referred to as a hyperparameter. The conditional probabilities are defined using the softmax function as follows:

$$p\left(w_i|w_j\right) = \frac{\exp\left(u_{w_i}^T v_{w_j}\right)}{\sum_{l=1}^{V}\exp\left(u_l^T v_{w_j}\right)}$$

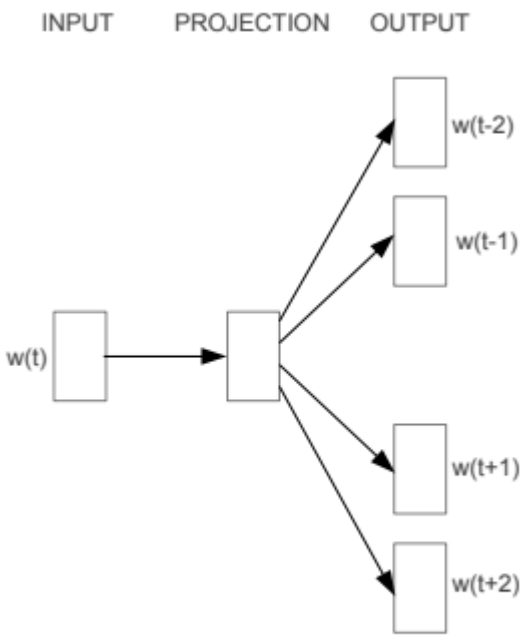
V here is the number of words in the vocabulary of the considered language, $u_{w_i}$is the output representation/vector of the word w_i, and v_{w_j} is the input representation/vector of the word w_j.

IV) Skip-gram model

As previously mentioned, the Word2Vec method is based on the skip-gram model, which uses a very simple neural network with one hidden layer to project words into the target space. The hidden layer contains N neurons, N being an arbitrary hyperparameter, and its activation function is the identity. The output layer contains $2k$ neurons, and its activation function is a softmax that associates each of the $2k$ words of

the context with a vector of size V, containing the probabilities that this word is each of the words in the considered language's lexicon.

The input vector is actually derived from the one-hot encoding of the entire vocabulary, resulting in a vector of length V with a 1 at the coordinate representing the considered word and 0 elsewhere. The output vector is a vector of length $2k \times V$ which is thus a projection of the corresponding input vector that will best represent the considered word. Below is a simplified schematic representation of the skip-gram model (source: https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-skip-gram.html):



Skip-gram