Lab Word Embedding n°1

Algorithms comparision and input quality importance

Word2Vec using srt files

1. Download the zip file at this address: https://github.com/AttilaDSA/IntilaqDSAcademy/blob/master/Word%20Embedding%20Labs/Word%20Embedding%20Lab%20n%C2%B01/HIMYM.rar
   These zipped files contain all the subtitles from a season of a TV series. We will try to train our word embedding model using them as input.

2. Using the **srt** python-library, read and parse the srt files.

3. Separate the sentences and clean them (remove the non alphabetical characters).

4. Transform each sentence into a list of words (lower the case to that case won't affect the training algorithm).

5. Create the *input* variable which must be a list of a list of words. (A list of sentences where each sentence is a list of words)

6. Using the **Word2Vec** from the **gensim** python-library, train the word embedding model using our *input*.

7. Using the **most_similar** function of the resultant word embedding model, search for the most similar words of the word "man".

Word2Vec using a complete text

1. Download the xml file from this location: https://github.com/AttilaDSA/IntilaqDSAcademy/blob/master/Word%20Embedding%20Labs/Word%20Embedding%20Lab%20n%C2%B01/ted_en-20160408.rar

2. Following this tutorial, https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c, clean and train your **Word2Vec** model using this xml file.

3. As before, get the most similar words for the word "man".

4. Compare the results obtained with the ones returned by the previous model (srt based training). Which ones are the most relevant?

FastText using a complete text

1. Using the same input as the last model (Word2Vec using complete text), train a **FastText** word embedding model.

2. The **Word2Vec** algorithm trains itself using the complete word token as input. But the **FastText** trains itself after exploding the words in little fragments (apple → app, ppl, ple). So,as before, get the most similar words for the word "man" and see how these two different approaches affect the results.

3. Choose a word that's doesn't exists among the input text then search the most similar word using the **FastText** and the **Word2Vec** models. Compare the results.