**Dictionary Additions**

The following algorithm it has been created to generate new words sentiment scores starting from already existing sentiment scores, where these last scores are taken from VADER lexicon.

First of all a set of valuable additions is created by comparing two words sets: the selected sentences from the articles set and the English language set.

All the procedure uses words contexts as central pivot, namely the underlying assumption is that if two words have a similar context, they should convey a similar sentiment.

***Extracting possible additions:***

In order to extract possible dictionary additions the set of articles selected sentences is compared to the English language set.

The first set is extrapolated from the articles selecting the single commodity related sentences: this set is called . The second set, called is composed by Google N-Grams project *corpus*.

First of all each word in is counted in the same sentences set, then it’s counted in . Now I can compute

where is the relative frequence of the word *i* in the set , is the single word *i* counter in the set and is the total amount of words contained in the set .

By assuming that those sets are big enough to represent the *English commodity-related language* and a *general English language*, those relative frequencies represent the probabilities of encountering word *i* in a random text from one of those two sets.

The last step for selecting possible additions to the dictionary is to extrapolate the words for which

Namely the words that are more probable to appear in the article sentences set than in the general English set are selected for a dictionary addition.

***Words Context:***

The context of a word is defined as the following and preceding words centered on a given word. If we consider

*“The following statement deals with words context”*

In order to extract the context we have to firstly divide the sentences in N-Grams, then study the N-Gram related to a certain word. In this easy example by considering Tri-Grams and the words *“statement”* and *“words”* their contexts will be respectively *“following, the, deals, with”* and *“deals, with, context”*.

In a more complex setting a *corpus* can be used to extract a word context, namely the most present Tri-Grams words can be extrapolated from a given word using the *corpus*.

The context can be used to compute the *Semantic Similarity* between two words, namely two words have a similar context if they have a similar meaning.

In order to compute the *Semantic Similarity* I need to compute the word count for each word *i* defined as

where is the counter and is the word *i*.

Then I define

where is the word *j*, id the word *i* and defines the N-Gram interval . Practically defines how many times the given word *i* co-occurred with a given word *j*.

Now The *Pointwise Mutual Information Criterion* can be computed as

The PMI computes the discrepancy between the probability of the coincidence of two words given their individual and joint distributions assuming independence, namely two words get an high PMI if they have an high probability of co-occurrence because if in case of words independence.

In order to compute two words *Semantic Similarity* have to be created two sets for , of contexts words in descending order by their PMI values, namely

These two sets have to be looked for words co-occurrences, namely words which appear both in and . Once found those words it has to be computed for both , the sum of those words PMI, which is defined as

where , are the PMI scores of the co-occuring words and is an arbitrary parameter defined as : in particular the greater , the greater the emphasis on words having high PMI scores.

Finally the *Semantic Similarity* between two words , can be defined as

,

where is the PMI sum and , are defined as

where is word counter, is the total amount of words in the corpus and is an arbitrary parameter that should be proportional to corpus size: the bigger the corpus, the bigger .

The aim of all this procedure is to combine existing VADER sentiment scores with *Semantic Similarity* measures to get new words sentiment scores.

***The input***

The inputs used are commodity sentences extracted using the Sentences Selector and VADER lexicon. In particular this test version uses 11341 wheat-related sentences from *amis\_articles\_27\_11\_2016\_sentences\_wheat.jsonl*, while the unmodified VADER lexicon counts 7503 words.

***The output***

At the moment the output is composed by a list of possible dictionary additions named *selected\_words­*: those words are the ones representative of the given sentences set (in probability of appearance terms) that are not already contained in VADER lexicon set.

Both selected words and lexicon contexts are computed: they are named *contexts* and *contexts\_lexicon*. Both of them are computed on the sentences set for consistency.

Using the contexts, Pointwise Mutual Information statistics are computed between each lexicon or selected word and their same context words: this computation is necessary for computing *Semantic Similarity* between words.

Once *Semantic Similarity* will be computed, it will be used to derive new words sentiment scores from VADER lexicon existing ones.

***The algorithm***

Algorithm main script is *dictionary­\_additions.py* which uses the auxiliary scripts *word\_counter.py* and *lexicon\_modification.py*.

*dictionary.py* loads needed modules, reads the lexicon file from VADER sentiment analysis tool and the set of commodity sentences. At the moment the sentences set is restricted to just wheat sentences, but the same methodology will be applied for all the other commodities.

Once the data sets are loaded, *dictionary.py* calls the function *words\_extraction(sentences, lexicon)* from *word\_counter.py* and delivers a set of possible dictionary additions: per each selected word, namely for each possible dictionary addition, it’s then extracted the word *context* using as *corpus* the same commodity related sentences. As next step the script takes each lexicon word and extracts their context always through an analysis based on the same set of commodity sentences. The contexts are extracted always from the same set of sentences because, if sentiment similarity is assumed to be based on context similarity, would have been wrong to base the context analysis on different sets because given a certain word context on one hand its associated sentiment is dependent on the general context in which is used, while on the other hand many contexts are present just in a few general contexts, therefore by basing the analysis on two different general contexts could have leaded to selected words contexts not mirrored by lexicon contexts. However both the words selection for addition and the contexts extraction processes will be explained successively in this paper.

The Pointwise Mutual Information statistic is computed for each context word once each selected word for addition and lexicon words contexts are extracted. As before the PMI statistic is computed for both the sets of words and their contexts.

The auxiliary script *word\_counter.py* is composed by a set of functions which estimate the presence of a certain word type in the sentences set or the English language set (Google NGrams Project). In particular the function *wordcounter(sentences)* computes the amount of times that a certain word appears in the sentences set, while *wordcounter\_google(wordfreq)* does the same using Google NGrams set for the English language. The two functions *probabilities\_sentences(wordfreq)* and *probabilities\_google(wordfreq)* compute the relative frequencies, namely the probabilities, of each word appearance in both sentences and Google NGrams set. The functions *avoid\_divide\_by\_zero(wordfreq)*, *convert(wordfreq)* and *wordfreq\_to\_dic(sentences)* are functions used for formatting reasons: in particular *avoid\_divide\_by\_zero(wordfreq)*, which gives a minimal value to if , it’s needed because a few lexicon words are not present in the sentences set, therefore when computing the ratio

the algorithm could stop in those cases. At last it’s important to underline the functions *load\_total\_counts* and *retrieve\_absolute\_counts* because they are used for Google NGrams information retrieval: while first one connects to the API and retrieves year by year the words total counts , the second one computes each single word . Google NGrams information retrieval is set on the English language corpus (also other languages are available) for the year spanning between 2007-2008, namely the nearer to our data.

All those functions are clustered in the function *words\_extraction.py(sentences, lexicon)* which gives back the set of selected words for dictionary addition.

The second auxiliary script called *lexicon\_modification.py* contains the functions which read VADER lexicon, extract each word both word sentences and lexicon contexts and compute the Pointwise Mutual Information statistics per each selected word for dictionary addition. The function *contexts\_extractor(sentences, selected\_words)* defines each selected word and lexicon word context from the same sentences set: it finds the word in the sentences set, finds the pertinent trigram, per each word in the trigram computes the absolute frequency in the corpus and gives back such number as definition of each word context. Following the function *pointwise\_mutual\_information(wordfreq, contexts)* computes each context word PMI referred to the selected keyword by applying the formula described into the theoretic part.

***Results and Next Research Topics***

At the moment there is still some work to do on the model, as soon as some a few issues with Google NGrams API will be solved.

* set computation as
* , derivation
* New sentiment scores computation

At the moment the model is unfinished because it cannot be tested properly on a small sentences sample because, being context-based, derived contexts would be not representative and contain words which misdirect the analysis.

However it computes a set of selected words for possible addition, both *selected\_words* and lexicon contexts and PMI values. Trying the script on a set of 1100 wheat sentences (from the first one to the 1100 one) and looking at the available results, it’s possible to underline a few interesting results.

First of all are underlined as possible dictionary addictions the following words: *limited*, *third-biggest*, *whose*, *rise*, *sellers*, *four*, *western*, *@*, *Beijing* and *risk*. By logic not all of these words convey sentiment, as words like *four* and *@* should be dropped, but interestingly enough words like *rise*, *limited* and *risk* are selected and they, always by logic, convey sentiment. Moreover the lack of the word *rise* in VADER lexicon for an economic context has been underlined many times.

Moreover if we look at the context of the word *rise* we can find many interesting words as *price*, *wheat* and *harvest* amongst others: this means that the word *rise* is used in the context of those words and, always by logic, may be connected to both prices and production.

As soon as the sentiment scores will be computed and added, the long term objective is to make the model completely self-adapting and machine-based. At the moment is machine-based because it relies on just VADER lexicon the sentences set which can be modified over time to fit new sentences sets, while for being self-adapting a system of words deletion has to be created. This last point can be achieved by using the *Semantic Similarity* by the contrary, namely by erasing the words which show low , in the same lexicon: in this way lexicon consistency is maintained time to time.