Analysis and reproducibility of paper “A Word Embedding based Generalized Language Model for Information Retrieval”, D. Ganguly, D. Roy, M. Mitra, G.J.F. Jones, 2015.

Alessio Lazzaron, Matteo Marchiori, Andrea Oriolo, Fabio Piras, *Group IR1.3*

1 Introduction and Comprehension

With the term Statistical Language Modeling or more simply Language Modeling (LM), we mean the development of probabilistic models that are able to predict a term given a context.

Due to the ambiguity of human language and its intrinsic properties, this process is very difficult to reproduce; however, several models have been developed over the years in order to estimate word representations. Some examples are: Latent Semantic Analysis (LSA), where the terms are represented in a space of reduced dimensionality; Latent Dirichlet Allocation (LDA), which goes to represent the dependencies of the terms assuming that each of them is generated by a set of latent variables called topics.

The main problem is that these models care only about the occurrences of words at the level of individual documents, leading to results that are not always very reliable. In this paper the Generalized Language Model (GLM) is proposed [2], an information retrieval model that allows you to directly model the dependencies of a term using the Word Embedding technique, thus allowing you to store both semantic and syntactic information of words in a space vector. The closer the vectors are the more the words are semantically similar, that is, they occur in the same linguistic contexts.

Unlike the basic Language Model, which assumes that each term is independent from each other, the Word Embedding technique takes into consideration the local context of the word in order to create a better model for predicting the dependencies among terms and thus significantly improving the quality of the retrieval. With the Generalized Language Model we want to generalize the basic Language Model paradigm by transforming a term t' of a document into a term t of the query through a noise channel.

In detail, this transformation process can take place in 3 ways:

- Direct term sampling: no transformation takes place, it represents the standard Language Model;
- Transformation via document sampling: the term t is generated starting from a term t' belonging to the document;
- Transformation via collection sampling: the term t is generated starting from a term t' belonging to the collection.

Once the terms have been calculated using these three approaches, the generalized model involves the combined use of all three methodologies. The following figure shows how is it possible to build the model:

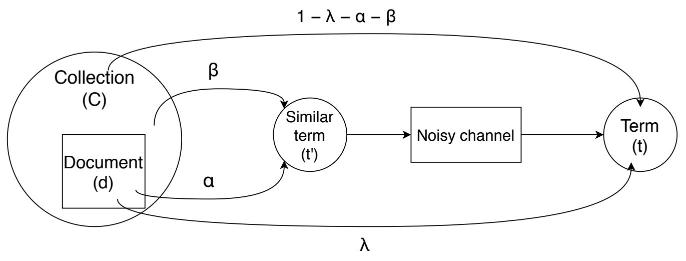


Fig. 1. Schematics of generating a query term t in our proposed Generalized Language Model (GLM). GLM degenerates to LM when $\alpha = \beta = 0$.

From our point of view, this transformation process calculates a similarity value between the term t' and t or how semantically similar the two terms are in the representation of the Word Embedding.

According to the GLM, the probability of the query term t, given a document d in collection C, is provided by the following equation:

$$\begin{aligned} P(t|d) &= \lambda P(t|d) \\ &+ \alpha \sum_{t' \in d} P(t, t'|d) P(t') \\ &+ \beta \sum_{t' \in N} P(t, t'|C) P(t') \\ &+ (1 - \lambda - \alpha - \beta) P(t|C) \end{aligned} \tag{1}$$

In detail:

- the first term in lambda represents the probability of observing a query term t without the transformation process;
- the probability of observing a term t obtained by transforming a term t' sampled from the document is represented by the second term in alpha in the formula;
- the third term of the formula in beta represents the probability of observing the term t obtained from the transformation of t' from the entire collection;
- the last term represents the remaining probability of observing a term t from the collection without the transformation process.

This model takes into account the correlation of terms using the transformations mentioned above.

Analyzing in detail every single probability:

$$\lambda P(t|d) = \lambda \frac{tf(t,d)}{|d|} \tag{2}$$

is the probability of the term t of the query within the document d, calculated as the frequency of the term with respect to the number of words in the document. α is the weight assigned to the probability of transformation of the term t into the term t', with t' belonging to the document d.

$$\begin{gathered}\sum_{t'\in d} P(t,t'|d)P(t') = \\ \sum_{t'\in d} \frac{sim(t,t')}{\sum_{t''\in d} sim(t,t''))} \frac{tf(t',d)}{|d|} P(t')\end{gathered} \tag{3}$$

is the sum of the probabilities of transforming the term t of the query into a term t' of the document d.

β is the weight assigned to the probability of transforming the term t into the term t', with t' belonging to the collection C.

$$\begin{gathered}\sum_{t'\in Nt} P(t,t'|C)P(t') = \\ \sum_{t'\in Nt} \frac{sim(t,t')}{\sum_{t''\in Nt} sim(t,t''))} \frac{cf(t')}{cs} P(t')\end{gathered} \tag{4}$$

is the sum of the probabilities of transforming the term t of the query into a term t' belonging to the collection C.

$$\begin{gathered}(1-\lambda-\alpha-\beta)P(t|C) = \\ (1-\lambda-\alpha-\beta)\frac{cf(t)}{cs}\end{gathered} \tag{5}$$

is the probability of the term t of the query to be inside the collection C.

The Cosine similarity is used to calculate similarity, a technique used to measure the similarity of two terms through the cosine of the angle between the vectorial representations of the two. In detail, given two vectors A and B, which respectively represent the terms t and t', the score of similarity can be calculated as follows:

$$\begin{aligned} sim(A,B) = \cos(\theta) &= \frac{A * B}{||A||||B||} \\ &= \frac{\sum_{i=1}^{n} Ai * Bi}{\sqrt{\sum_{i=1}^{n} Ai^2} * \sqrt{\sum_{i=1}^{n} Bi^2}} \end{aligned} \tag{6}$$

The peculiarity, in our case, lies in the fact that the vectors are represented in the form of embedding. Among the existing techniques, Word2Vec was used for creating the embedding of words. This methodology, based on the use of neural networks, allows you to obtain predictions about words, assign a probability distribution to terms and also allows you to better capture analogies and similarities between semantically related words. In this case Word2Vec is used to extract the embeddings of the terms or their vectorial representation, leaving out the tasks of the respective neural networks.

Taken each term t' from the collection or document, we calculate the similarity between these and the term t of the query. As the similarity value increases between a term t of the query and the terms t' of the document and the frequency of each similar t' within the document, the value of the second term (α) in Formula 1 also increases. This means that the terms inside of the document are semantically similar to the term t of the query and therefore the term t fits well with the document. If, at the same time, the similarity between the term of the query t and the terms t' taken from the collection is high, the third term of Formula 1 will be higher. To avoid taking all the terms in the collection, a list of neighbors of the word is precalculated in order to make the calculation more efficient. This model favors documents containing terms that are semantically similar to the query term t. In fact, semantically similar words tend to contribute more to the increase in the score of the term t, which will better adapt to the context of a document with terms closer to it. The higher the similarity value, the more relevant the correspondence between the terms will be.

2 Aspects reproduced

To reproduce the evaluation results reported in the paper, we decided to use the most recent version of Lucene available at the date of paper's publication, that is Lucene 5.1.0. This choice was made because the Lucene framework has very different aspects between one version and another, many implementations change and it is not backwards compatible. The aspects reproduced concern the indexing of the documents present in the TREC 7 collection. The indexed TREC documents are in XML format, described in DTD specifications. In fact, each type of document has a structure different from the others. There are four types of documents:

- FR94: Federal Register 1994;
- FT: Financial Times Limited 1991, 1992, 1993, 1994;
- FBIS: Foreign Broadcast Information Service 1996;
- LATIMES: Los Angeles Times 1989, 1990.

Although each structure is different, we have noticed the similarity with standard HTML code, also as regards the entities, for this reason we decided to use an HTML parser to obtain the documents in text format. To create the transposed index we used the *Lucene Standard Analyzer*, which executes the following steps [1]:

- Tokenization: Split the document into tokens (words, connected together in the case of an apostrophe and in other particular cases);
- Stopwords removal: removal of a standard stoplist;
- Creation of the transposed index: the index is saved on the disk.

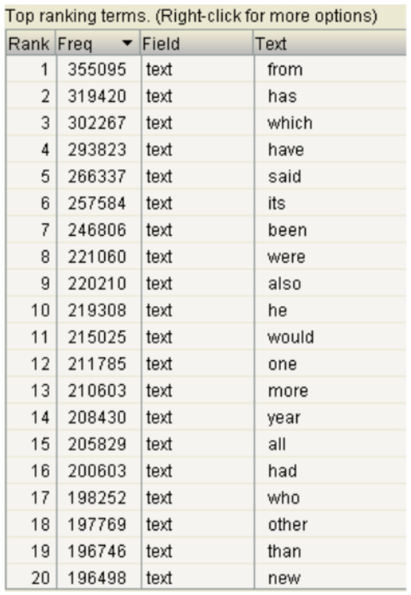
At this point we have tried to reproduce the results obtained with the basic language model, i.e. with Jelinek-Mercer and parameter λ set to 0.2 (without transformation into similar terms).

The reproduction of the GLM turned out to be more difficult than expected. A first approach was trying to reproduce the probability described in the various terms reported above into functions, extending the LMSimilarity class made available by Lucene to represent a similarity, but obtaining an incomplete and inefficient implementation just for the calculation of the probability of the term without transformation. Subsequently, using the GitHub repository of Debasis Ganguly [3], one of the authors of the paper, and having adapted it to our needs, we managed to obtain a better implementation, even if not yet sufficiently performing to get to the results reported in the paper. This repository has many classes useless for the purpose of reproduction, that we deleted. We later adapted some improperly used Java constructs, in order to obtain a more readable and performing code (functional programming). The classes we used are:

- LMLinearCombinationTermQuery: contains the implementation of second and third term of the GLM formula;
- TRECQuery: contains a method for obtaining the w2v representation of the query represented, based on the GLM;
- WordVecIndexer: index documents to get the words of the same. With a list of nearest neighbors, it expands the index with similar terms in the document and collection. To calculate the nearest neighbors use a list of terms in w2v representation;
- WordVecSearcher: runs a GLM run.

3 Results obtained

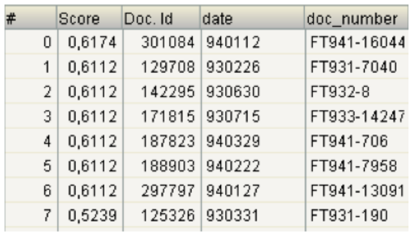
THE first result we managed to obtain is the index created by Lucene. For visualization of it, we used the tool called Luke that allows you to show the content.


Top ranking terms. (Right-click for more options)

Rank	Freq	Field	Text
1	355095	text	from
2	319420	text	has
3	302267	text	which
4	293823	text	have
5	266337	text	said
6	257584	text	its
7	246806	text	been
8	221060	text	were
9	220210	text	also
10	219308	text	he
11	215025	text	would
12	211785	text	one
13	210603	text	more
14	208430	text	year
15	205829	text	all
16	200603	text	had
17	198252	text	who
18	197769	text	other
19	196746	text	than
20	196498	text	new

Fig. 2. Index shown with Luke

The following table illustrates the first 8 documents relating to the term *'from'*, term with rank 1 in the transposed index. The documents in the example are sorted by decreasing value of tf * idf.



#	Score	Doc. Id	date	doc_number
0	0,6174	301084	940112	FT941-16044
1	0,6112	129708	930226	FT931-7040
2	0,6112	142295	930630	FT932-8
3	0,6112	171815	930715	FT933-14247
4	0,6112	187823	940329	FT941-706
5	0,6112	188903	940222	FT941-7958
6	0,6112	297797	940127	FT941-13091
7	0,5239	125326	930331	FT931-190

Fig. 3. Simplest query ever with Luke

The results of the basic Language Model are as follows:

Topic Set	MAP	GMAP	RECALL
TREC-6	0.1612	0.0843	0.5382
TREC-7	0.1734	0.0843	0.5444
TREC-8	0.1899	0.0906	0.5484
Robust	0.2021	0.1016	0.6052

TABLE 1
Results of basic LM

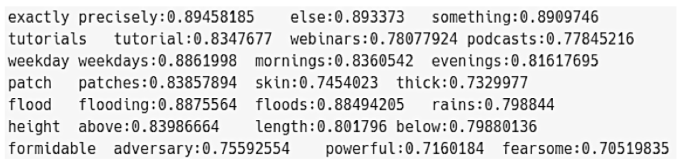
We found minimal differences between the values in the LM paper and the values obtained from our implementation of the basic language model (Jelinek-Mercer). We hypothesized the following reasons:

- Our indexing uses Lucene's standard stoplist. The paper does not specify whether the removal of the stopwords has been performed and with which stoplist;
- we used Lucene's standard parser to get usable queries to get statistics related to the Jelinek-Mercer

language model. This implementation may have changed but we have not studied in depth on GitHub (Lucene is open source and it is possible to perform a check), or in the paper they may have used their own implementation for parsing queries;

- We don't know if a stemming was done in the base language model. We didn't do it.

As far as GLM is concerned, we were unable to reproduce the results due to the excessive time needed to extrapolate the k nearest neighbors from the used word2vec pretrained dataset (glove.6B.50d.txt). Although one of the smallest, it has a too high dimensionality for the time available.



```
exactly precisely:0.89458185    else:0.893373   something:0.8909746
tutorials   tutorial:0.8347677  webinars:0.78077924 podcasts:0.77845216
weekday weekdays:0.8861998  mornings:0.8360542  evenings:0.81617695
patch   patches:0.83857894  skin:0.7454023  thick:0.7329977
flood   flooding:0.8875564  floods:0.88494205   rains:0.798844
height  above:0.83986664    length:0.801796 below:0.79880136
formidable  adversary:0.75592554    powerful:0.7160184  fearsome:0.70519835
```

Fig. 4. Example of found k-nearest-neighbors

Leaving aside the aspect of computationally long times, the final formula for the calculation of probabilities has been reproduced in order to have a complete implementation at the code level.

4 Final observations

The General Language Model studied has been useful to understand that, using the Word Embedding technique as a representation of words, it is more efficient than the basic Language Model, because it allows you to effectively find similar terms with each other through, for example, the Cosine Similarity. Furthermore, through this technique, the semantically similar terms are also considered (i.e. "ship" and "boat") while in the basic Language Model the terms are compared without any transformation. The lack of experience with the Lucene Java library and the difficulty in finding a correct implementation to adapt to our case, meant that it took us a long time to understand the most appropriate way to carry out the different steps necessary to reproduce the results of the paper. The reason why we have not used another library nor another language lies in the fact that we have decided, from the beginning, to faithfully reproduce the work done in the paper. However, this choice does not exclude possible future works in which the work could be implemented through other languages or libraries (e.g. gensim in python).

References

[1] M. Agosti and G. Silvello, *Slides* Padua, Italy: academic year 2019-2020.
[2] D. Ganguly, M. Mitra, D. Roy and G. Jones, *A Word Embedding based Generalized Language Model for Information Retrieval* Dublin, Ireland; Kolkata, India: SIGIR 2015.
[3] D. Ganguly, *GLM GitHub repository*