Web Information Retrieval **Spring 2018**

WIR Final Relation

Members: *Angelo Catalani, Valerio Colitta, Alessandro Lo Presti*

1.1 Introduction

The vector space model is characterized by the representation of documents and queries as vectors and the cosine similarity to compute the similarity score between them.
This means that if the distinct terms in a given collection are n, each document is a n-dimensional vector.
The VSM assumes that for each term t_i there exists a vector $\vec{t_i}$ in the vector space the represents it, and these vectors form the space basis.
This leads to the fact, each document (or query) is a linear combination of the vectors t_i : $\vec{d_k} = \sum_{i=1}^{n} w_i \vec{t_i}$, where w_i is the tf-idf score for the term t_i in the document : d_k .
The underlying assumption is the pairwise orthogonality of terms vector that brings to the conclusion there can not be any semantic correlation between terms.
The GVSM extends the VSM by non considering terms vectors orthogonal and computing the similarity score between terms according the following formula : $score(\vec{d}, \vec{q}) = \frac{\sum_{j=1}^{n'} \sum_{i=1}^{n'} \vec{t_i} \cdot \vec{t_j} \cdot d[i] \cdot q[j]}{||\vec{d_k}|| \cdot ||\vec{q}||}$.
It is interesting to note this formula reduces to the classical cosine similarity assuming pairwise orthogonality of the terms vectors because that product is always zero except when they are the same($i = j$).
The dot product of the terms is their semantic correlation.

1.2 Generalized Vector Space Model

The paper[LM89] taken into consideration, considered documents to be $n' = n \cdot (n-1)/2$ dimensional vectors where, each dimension stands for a distinct pair of terms and all the distinct terms are n.
In particular for a given document d_k, the entry $d_k[i][j] = d_k(t_i, t_j) = (tf-idf(t_i, d_k) + tf-idf(t_j, d_k)) \cdot \vec{t_i} \cdot \vec{t_j}$.
The dimension is not $n * n$ because it is assumed the similarity function between terms is symmetric, so that the entry $d_k(t_i, t_j) = d_k(t_j, t_i)$.
Finally, the score function has been defined as : $score(\vec{d}, \vec{q}) = \frac{\sum_{j=1}^{n'} \sum_{i=1}^{n'} d_k[i][j] \cdot q[i][j]}{||\vec{d_k}|| \cdot ||\vec{q}||}$

1.2.1 Implementation

The first task has consisted of parsing the dataset and build a matrix of tf-idf scores where each row index is the documentID and each col is the termID.
This function is not performance critical because our dataset is relatively small.
The second task has been the implementation of the function, that takes as parameters 2 rows of that matrix and a term-to-term similarity function and computes the score function described above.
The first problem was the fact that the original formulation consists of a nested loop iterating over all the possible terms couples with a complexity of $O(n^2)$ that executed over all the queries and documents, causes a serious performance penalty.

To deal with this issue, an approximated function has been introduced that iterates only over the terms that appears in the document or in the query.
This approximation results in raising the final score with respect to the original one, because:

1. the numerator is the same since it is different from zero only when in the couple of terms considered, at least one belongs to the document and the other, but also the same, to the query.

2. the denominator is different from zero even if one terms belongs to the document (or query), and the other does not.

The approximated solution performed, in less time, with coherent results.
However, the second bottle neck is the computation of the term-to-term similarity function, whose description is delayed to the next section.

1.3 Similarity Function

1.4 Dataset and Result

The first dataset is made of

1. Documents : 11429

2. Queries : 90

3. Distinct terms : 11945

This dataset has associated a ground truth file with the relevant documentIDs associated to each query.
Since the computation of the score function is performance demanding the tests have been conducted after resizing the dataset.
The precision-recall plot is calculated as an average of the precision for a given recall level for the number query considered.
In particular 10 recall levels have been considered : $0.1, 0.2, ..., 1.0$.
In addition to this, it has been executed a test also on the classical cosine similarity where it has been possible to consider the whole dataset since each computation has a linear complexity.
AGGIUNGERE TEST 10 QUERY CON VELERIO SIM , CON UN ALTRA SIM

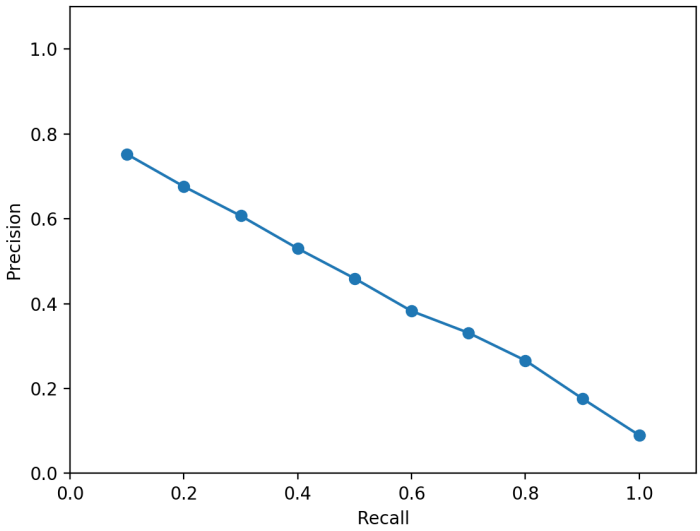

Figure 1.1: Precision-Recall Cosine-Similarity

1.5 Future work

In order to make the score function faster, each computation could be done in a multithreading context. In addition to this the term-to-term similarity scores could be precomputed and stored in a global dictionary.

References

[1] George Tsatsaronis and Vicky Panagiotopoulo, *A Generalized Vector Space Model for Text Retrieval Based on Semantic Relatedness.*
Department of Informatics Athens University of Economics and Business, 76, Patision Str., Athens, Greece, 2009

[2] Pedregosa, F. and Varoquaux, G. and Gramfort, A. and Michel, V. and Thirion, B. and Grisel, O. and Blondel, M. and Prettenhofer, P. and Weiss, R. and Dubourg, V. and Vanderplas, J. and Passos, A. and Cournapeau, D. and Brucher, M. and Perrot, M. and Duchesnay, E.
Scikit-learn: Machine Learning in Python.
Journal of Machine Learning Research, volume: 12, pages: 2825–2830, year: 2011