Web Information Retrieval
Final Project Results
A.Y. 2017/2018

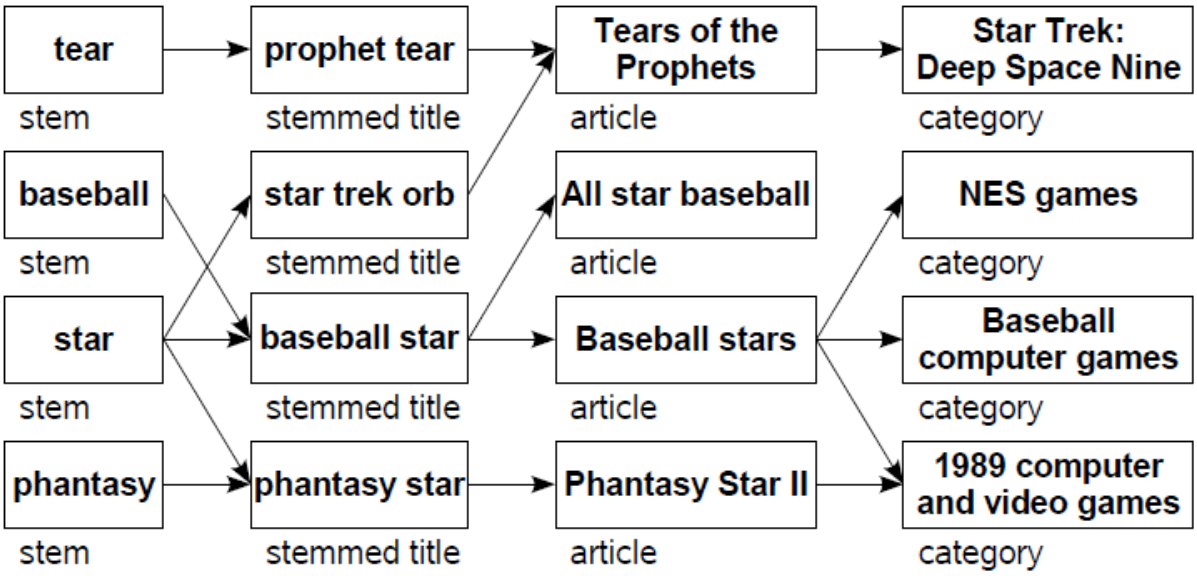
Team ESA
Federico Boarelli
Luca Carbone
Marco Cuoci

Concept:

Can we understand to which matter a document belongs just operating on its text? This is the aim of the paper written by Peter Schonofen, *Identifying document topics using the Wikipedia category network*, that, as the title says, take advantages of the most known and accessible information database: Wikipedia. The paper shows that after some operations on the text of the document, we are able to say to which Wikipedia's categories (its not limited to one) it belongs, taking for good the authoritativeness of the source. However, twelve years have passed since the publication of the paper: are the result obtained still valid under the same assumptions? This is the goal of our work: demonstrate that despite the time, this kind of operation is still valid and maybe became more reliable too. Here we quickly discuss the steps that we made and the results that we have obtained. First, let discuss which corpora we are going to use: after twelve years, the Wikipedia's corpus grew up in an exponential way since we pass from 800k documents to more than 10 million of documents, which was too large for be used directly without selection. So we decide to use the DBPedia API to build a random corpus of 30k documents, which, as we will see in the next sections will have influence over the final result. For the corpus of documents to be categorized we decide to use the same of the original experiment, the 20 Newsgroup one, trying to maintain the most similar conditions with the original paper. In the following steps we prepare the two corpora throw some operations (like removing stop words, stemming, lower casing) and then we finally start the categorization process, taking in consideration one document at time.

Creation of the random Wikipedia Corpus:

The wikipedia corpus as a set of html pages provides a lot of informations. But in the assigned paper all this informations are just partially used. In particular, we will use only titles of wikipedia articles and the list of related categories and redirectories, which will be used as additional titles of that article. For our experiments, we first need to build a special wikipedia corpus with just this informations, discarding links, text of articles, category hierarchy etc. First, we need to make a query to DBPedia to get all the informations we need, in particular starting from an article title we ask the list of categories and redirectories of that article. So our initial dataset will be composed of 30000 objects that are structured like: title of article, list of categories, list of redirections. After constructing this dataset, we build our corpus like this:
First, we have all the set of categories from the dataset, and we link each article to each category in which that article appears in the dataset. Then we build the list of titles. To do so, we remove stopwords and perform stemming on the title of the article and the redirections, obtaining a set of titles. Then we link each title to the article from which we generated this set. Note that due to stemming and stopwords removal, some titles of articles can collide, in that case the stemmed title will point to all the articles pointed before the stemming/stopwords removal operations. Finally, we build an inverted index on the terms from the whole set of stemmed titles. While constructing this corpus, we also update some other informations regarding each category. In particular, for each category, we save the number of articles that point that category. In addition, for each category, we maintain its vocabulary, namely the set of words in the inverted index that 'points' (indirectly) to the category.
The final schema of the corpus can be visualized in the following image:



After doing so, some categories are discarded, in particular the categories pointed from too much or too few articles, because are considered to be irrelevant.

Analysis of the 20 Newsgroup corpus

As said before, we have decided to use the 20 Newsgroup dataset. We need of course some preprocessing to be ready for the classification step. Given the objective to categorize the entire dataset, we create unique *while* cycle where we analyze one document at time. For each document, we perform the following steps:

- Removing of the stop word
- Apply stemming to each word
- Calculate Tf/log(icf) (inverted category frequency)
- Vectorize in a unique dictionary that will be reinitialized at each iteration

The definitions of each points are good enough for explain them self quite good, but lets go deep in the inverted categories frequencies definition: as said, we compute the term frequency for each word in each document, then, instead of applying the classic tf/idf, we divide the first term by a new element that is the result of the division of the total number of the categories by the number of categories that contain the word w in their vocabulary, all contained in a base

10 logarithm. A pure variant of the Tf/idf that use the categories as second element instead of the number of documents.

Categorization

In our project two categorization methods are used, in accordance to the methods used in the original paper. The "Soft" categorization method uses just the word's weights to determine the right category for the documents. While this runs faster, the results yielded are broad. The more complex "Hard" cathegorization method does some post-processing on the categories to achieve narrower results at the cost of time.

Soft Categorization

1: Remove stop words and perform stemming, remove words not occurring in Wikipedia article titles
2: Collect words of the document and weight them by $R_w = tf_w \times \log \frac{N}{cf_w}$
3: Collect Wikipedia titles whose words (with the possible exception of one) are all present in the document, and weight them by $R_t = \sum_{w \to t} R_w \times \frac{1}{t_w} \times \frac{1}{a_t} \times \frac{S_t}{L_t}$
4: Collect Wikipedia articles pointed to by the titles and weight them by $R_a = \max_{t \to a} R_t$
5: $R_c = \sum_{a \to c} R_a$
6: Select categories with the highest weights

Soft categorization algorithm

Step 1's focus is stopword removal and stemming of the words in the documents. Words not present in our wikipedia's vocabulary are also ignored. In step 2, each word w of the document is weighted (Rw). In the formula, tf is the term frequency, N is the number of categories and cf is the "category frequency": how many categories contain the word w. Note that the log N/cf is similar to the idf. Step 3 aims to weight all the words of the titles of the wikipedia articles whose titles appear in the document with the exception of at most one word. Similarly to step 2, step 3 also assigns weights Rt to titles t where Rw is the weight of a "supporting word", tw is the number of wikipedia titles containing word w, at is the number of articles pointed by title t, Lt is the title length in words, and St

is how many of these words are present in the document (at,least Lt - 1). In step 4 each wikipedia article is given the maximum weight calculated among the titles pointing to them. Step 5 assigns weights to the categories by summing all the weights of the articles in the category. Step 6 only serves to return the results.
In this phase no filter on the number of articles per category and no optimization of category weights are assumed. This is done because it has been noticed experimentally that applying these filters would greatly diminish the result set, favouring broader, bigger classes. The result set for each document showed some related categories and other widespread categories among the first ranks. This result is inline with the original study, which used the whole 2006 Wikipedia corpus as its dataset.

Hard Categorization

1: Remove stop words and perform stemming, remove words not occurring in Wikipedia article titles
2: Collect words of the document and weight them by $R_w = tf_w \times \log \frac{N}{cf_w}$
3: Collect Wikipedia titles whose words (with the possible exception of one) are all present in the document, and weight them by $R_t = \sum_{w \to t} R_w \times \frac{1}{t_w} \times \frac{1}{a_t} \times \frac{S_t}{L_t}$
4: Collect Wikipedia articles pointed to by the titles and weight them by $R_a = \max_{t \to a} R_t$
5: Collect Wikipedia categories assigned to the articles and weight them by $R_c = \frac{v_c}{d_c} \times \sum_{a \to c} R_a$
6: Decrease weights of categories sharing its supporting words with other categories of higher R_c values
7: Select categories with the highest weights

Hard categorization algorithm

While step 1 through 4 are the same, further steps are added in order to better refine and narrow down the result set. Step 5 assigns weight to the categories by taking the sum of their articles weight and multiplying it by a scaling factor composed of the number of supporting words per category c (vc) and the number of words in the category c (dc, c's vocabulary). In Step 6, common supporting words are pruned in order to reduce their influence. Step 7 is just like the "Soft" algorithm's step 6. In this phase both preprocessing and post-processing is done, such as: filtering on the number of articles per category and optimization of categories weights.

The result set shows for each document some related categories, less widespread categories but introduce unrelated categories into the results.
This is due to the smaller size of the corpus which is used in respect to the whole Wikipedia corpus, which in turn doesn't allow to take all the categories existing within Wikipedia, and to a difficult interpretation over the "supporting word" definition. The result obtained during this phase is not inline with the original study, due to the considerable reduction in size of the dataset used.

Results

We show now the results obtained over two documents that demonstrate what we have explain in the previous section. The first document analyzed is a short email entitled *Political atheists*, while the second one is a list of *Atheist Resources* (organization and favourite books). The results are in descending order.

Political atheist:

Soft categorization:

CATEGORY	WEIGHT
Socialism in Canada	31031.3480
Political movements in Canada	31031.3480
Social democracy	31031.348
Political history of Canada	31031.348
Metaphysical theories	23454.591

Hard categorization:

CATEGORY	WEIGHT
Epistemological theories	0.285
American male novelists	0.122
20th-century American novelists	0.073
Political history of Canada	0.055

Kantianism	23454.591		

Atheist Resources:

Soft Categorization: Hard Categorization:

CATEGORY	WEIGHT	CATEGORY	WEIGHT
Søren Kierkegaard	31031.3480	Statistical distance	3.0
Philosophy books	31031.3480	Abortion in New Zealand	1.34
Works about Søren Kierkegaard	31031.348	Communication	0.79
Atheism	31031.348	Economy and religion	0.66
Religious demographics	23454.591	Anthropology	0.624

First, we notice how the Hard categorization method outputs a weigh normalized, as we do not have extremely high result like in the Soft categorization method. But while the second method produce good results, very good in some cases like in the second document analyzed, the Hard method suffers a lack of contextualization due to an ambiguous definition of the supporting word definition in the original paper. The good results obtained with the Soft Categorization method underline how critical is the understanding of such definition, which if not correctly interpreted could bring on the wrong path the analysis result.

Conclusions

We want to understand if the categorization of the 20 Newsgroups corpus made twelve years ago with the whole Wikipedia's corpus was still valid with the Wikipedia of today, which ten times bigger just only in document number, and furthermore if those assumptions made were still valid with a random generated corpus. The first method implemented has produced an acceptable result, considering the randomicity of the corpus created, while the specialized categorization did not work due to an ambiguous definition over the set of the supporting word, which produces a useless result.

The implementation was done in Python using the NumPy and SciKit. The random corpus generated with DB Pedia counts 30k documents, while the 20 Newsgroups corpus is the same used in the paper, made by 18k documents.

References

- *Identifying document topics using the Wikipedia category network*, by Peter Schonofen – Hungarian Academy of Science, December 2006