SAPIENZA UNIVERSITÀ DI ROMA

Identifying document topics using the Wikipedia category network

Web Information Retrieval project

Fiorella Artuso (1602113), Andrea Migliori (1607771)

1. Introduction

This project aims to repeat an experiment presented in the paper "*Identifying document topics using the Wikipedia category network*" written by Peter Schönhofen. The goal of such an experiment is to show that it is possible to identify quite well the Wikipedia categories most characteristic of a document even with a simple algorithm that exploits only titles, redirections and categories of Wikipedia articles.
In fact, each Wikipedia article consists of:

- A **title**
- A **set of categories** which are organized hierarchically into sub-categories and super-categories
- A **set of redirections** that are other titles through which the main article is accessible

Our entire work is organized into three main steps:

1. Creation of the dataset
2. Implementation of the algorithm which takes a document as input and assigns a weight to each Wikipedia category; the top ranked Wikipedia categories are the most characteristic of the document.
3. Validation of the results

2. Experimental setting

In our work we made two main choices that differ from the paper:

1. reduced size of the dataset
2. reduced number of categories

2.1 Reduced size of the dataset

The Wikipedia snapshot used in the paper is related to 2006 and contains 878.710 articles, whereas the one related to the current year (2019) contains 5.789.574 articles because the number of Wikipedia articles grows exponentially in time.
Since the time needed to process each article so as to collect data that need to be added to the dataset is 30 seconds on average on our machines, it is infeasible for us to process all the currently available articles because it would takes more than 5 years. Therefore we decided to work on a restricted dataset consisting of 33.500 articles: since this process would require about 11 days of work, we decided to split the computation between us (half of the dataset to each of us) and then to merge the data we had individually collected.

2.2 Reduced number of categories

As a consequence of the reduced size of the dataset, the risk was that of selecting a high number of very heterogeneous categories. This would lead to the fact that each category is not supported by a sufficient number of articles so as to make it characteristic enough for labeling a document taken as input: the algorithm would then poorly identify document topics.
The solution we propose consists in:

1. taking one category among the list of Wikipedia's major topic classifications (such as arts, business, sport, religion, etc…)
2. selecting at random either the previously chosen category itself or one the its subcategories up to 3 level of indirection
3. randomly picking one Wikipedia article having the selected category among its categories set.

3. Creation of the dataset

Since Wikipedia articles in their original form (e.g. HTML pages) are not suitable for our purposes, once each article is selected as described above, all its necessary information (redirections, categories) is retrieved from **DBpedia.** In fact, it is a project aiming to extract structured content from the information created in the Wikipedia project and such extracted information is represented through the Resource

Description Framework (**RDF**), so we accessed data by using an SQL-like query language for RDF called **SPARQL**.

In order to add an article to the corpus, each article undergoes the following steps:

1. extract redirections and categories
2. perform stop word removal and stemming on article title and redirections. In addition, we removed information between parenthesis (disambiguation) since it does not necessarily appear directly in the document.
3. remove categories corresponding to Wikipedia administration and maintenance
4. remove categories containing less than 5 articles
5. merge stub categories with regular ones

The resulting dataset is structured as follows:

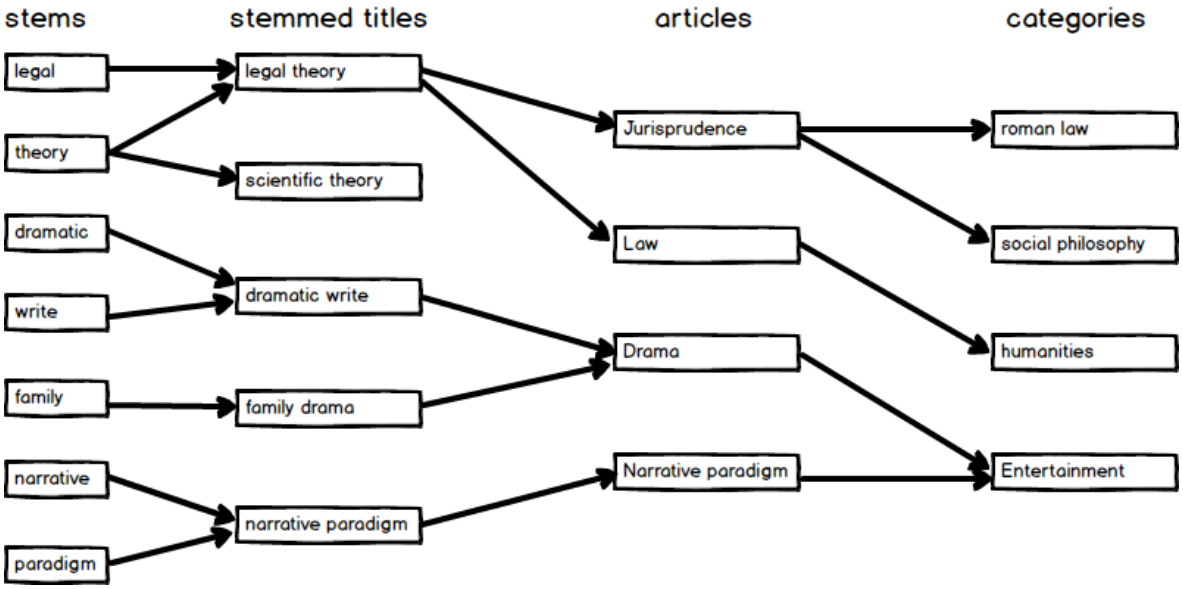


Figure 1

This structure has been implemented with four python dictionaries:

- **stems dictionary** – it represents a mapping between stems (key) and stemmed titles containing such a stem (value)
- **stemmed titles dictionary** – it represents a mapping between stemmed titles (stemmed redirections and stemmed article title) and their corresponding articles
- **article dictionary** – it represents a mapping between articles and their associated categories
- **categories dictionary** – for each category it keeps track of the number of articles in that category and the vocabulary.
  Note that the *vocabulary of category c* is the set of words occurring in the titles of articles in category c.

Notation

From now on, as already presented in Figure 1, we are going to adopt the following the notation:

- **stems** – they represents all the words occurring in the stemmed titles
- **stemmed titles** – they represents all the redirections and the titles of Wikipedia articles after stop words removal and stemming
- **articles** – they represent the actual titles of Wikipedia articles

4.identifying document topics

After we have prepared the dataset we are able to identify topics of a document taken as input through a series of steps.

In step 1, we perform stop words removal and stemming on the input document in exactly the same way as we did during the preparation of the dataset.

Note that in all the following steps we are going to consider only words that are present both in the document and in the stems of the dataset; all others words are ignored.

In step 2, we assign a weight R_w to each word w:

$$\boldsymbol{R_w = tf_w \times \log \frac{N}{cf_w}}$$

where tf_w is term frequency (i.e. number of times the word occurs in the document), N is the number of Wikipedia categories and cf_w is the category frequency (i.e. how many categories contain word w in their vocabulary).
The first factor gives importance to words occurring many times in the document because if a word appears many time in the document it is probably informative of the document topic.
The second factor prefers words which are related only to few categories thus avoiding to give a too high weight to very common words that obviously will appear in many titles and so being linked to very sparse categories. Note that this second factor looks like the inverse document frequency (idf), but we did not use it because otherwise we would have lost the concept of the importance of a word with the respect to categories associated to it.

In step 3, we collect stemmed titles supported by words present in the document. A word w supports title t if w occurs in t, and out of the other M words of t, at least $M-1$ are present in the document. Such a single word mismatch between the title and the document is allowed to properly handle documents that refer to persons, places or technical terms in an
incomplete way, for example “Vincent van Gogh” may appear as “van Gogh”.
Now stemmed titles we just collected are weighted according to this formula:

$$\boldsymbol{R_t = \sum_{w \to t} R_w \times \frac{1}{t_w} \times \frac{1}{a_t} \times \frac{S_t}{L_t}}$$

where w is a supporting word for title t, R_w is its corresponding weight, t_w represents the number of stemmed titles containing word w, a_t is the number of articles pointed to by title t, L_t is the title length in words, and S_t specifies how many of the title words are present in the document.
The first factor gives more or less importance to a title t according to the weights of the supporting words constituting it.
The second factor reduces the weight that common words pointing to many titles would have and the same does the third factor with respect to titles pointing to many articles.
The last factor simply measures how much percentage of the title words occur actually in the document so that titles having few words in common with the document receive a lower weight.

In step 4, we collect articles pointed to by the titles found in the previous step and we weight them according to the following formula:

$$\boldsymbol{R_a = \max_{t \to a} R_t}$$

Note that for the computation of articles weight we do not sum weights of all the previously collected titles referring to such articles since the fact that an article is pointed by many titles does not reflect the importance of that article but the structure of Wikipedia.

In step 5, we collect all the categories associated to the articles above and we weight each of them according to the following formula:

$$\boldsymbol{R_c = \sum_{a \to c} R_a}$$

However this formula has a limit because a category can have an high weight due to the presence of many titles pointing to articles in that category. In order to smooth this effect we implemented a **first improvement** by modifying the formula as follows:

$$\boldsymbol{R_c = \frac{v_C}{d_c} \times \sum_{a \to c} R_a}$$

Where v_C is the number of supporting words of category c whereas d_c is the number of words of the vocabulary of c. Note that supporting words of category c is the set of supporting words contained into the supporting titles which refer to the articles in category c.

We can now introduce a **second improvement**. Since a supporting word may appear in the vocabulary of many categories it would contribute to their weights in exactly the same way, however it would be better for such supporting word to contribute differently to each of the categories whose vocabularies contain it.
First, for each word of the document, we set up a d_w decay value, initially 1. Then, we sort the categories in descending order according to their weight. For each one, we recompute its weight and then we recompute the decay values for its set of supporting words (B_c).

$$\boldsymbol{R'_c = R_c \times \frac{\sum_{w \in B_c} d_w}{|B_c|}}$$

$$\boldsymbol{d'_w = \frac{d_w}{2}, w \in B_c}$$

In step 6, we simply select the top n categories, i.e. the ones with the highest weight because they will be considered the most characteristic topics of the document.

5. Validation of the results

In order to validate our experiment we have performed two kinds of tests.

The first one simply aims to test whether our algorithm is good in predicting the central topic of a given document by observing the top 20 categories assigned to it.
Such input documents are selected from the 20 Newsgroups dataset which is a collection of approximately 20'000 newsgroups documents partitioned heavily across 20 different newsgropus.
Figure 2 shows the results of our algorithm on the document "49960" which is about atheism:

PRE-OPT	OPT1	OPT2
Religion: 37.05451373682791	Atheism: 2.6921802698543633	Atheism: 2.6921802698543633
Philosophy: 35.12936723774073	Skepticism: 2.3905004638483325	Skepticism: 1.6932711618925689
Language: 35.01543176590475	Philosophy: 1.7717419998164892	Philosophy: 1.6190056205219643
Life: 32.789054970281136	20th Century Fox films: 1.7617171507354547	20th Century Fox films: 1.4814898813002688
Entertainment: 25.216036981868772	Criticism of religion: 1.627509063266141	Religion: 0.8520139465581635
Mathematics: 24.606303401321014	Humanism: 1.590082513244776	Humanism: 0.7232666987328669
Culture: 23.998614592309835	Religion in science fiction: 1.5543366334022737	Language: 0.6744504458044537
Science: 23.219079162113353	Søren Kierkegaard: 1.4654113367217068	Criticism of religion: 0.6200034526728156
Business: 20.718455778628066	2010s thriller films: 1.4614737164603169	Søren Kierkegaard: 0.54952925127064
Living people: 19.851070046393627	Religion: 1.4519527609985272	2010s science fiction films: 0.4784983196730488
Law: 18.706224847939733	Agnosticism: 1.387523534425303	Religion in science fiction: 0.47185219228283304
Politics: 17.48351855466089	2010s science fiction films: 1.3199953646153 07	Life: 0.46452479944021596
Sports: 16.723041425670747	Films about religion: 1.316822925946283	Culture: 0.3984840607316937
Technology: 16.292607938792003	Philosophical movements: 1.3142527675337132	2010s thriller films: 0.3653684291150792
Society: 16.02152394280103	Space adventure films: 1.2934186919975765	Space adventure films: 0.33851192329624075
History: 14.613003560780115	Irreligion: 1.2904276650775248	Mathematics: 0.27537697449825915
Concepts: 14.09205077445818	Language: 1.2679902191431995	American films: 0.26457150155370435
Reference: 13.422759302194454	Secularism: 1.1910682016011611	Philosophical movements: 0.2592568935955176
Nature: 12.901784001880241	Life: 1.1815875664966176	Science: 0.23822797895210296
Education: 12.892386207195324	Culture: 1.1752969797866306	Collective rights: 0.229677263171798

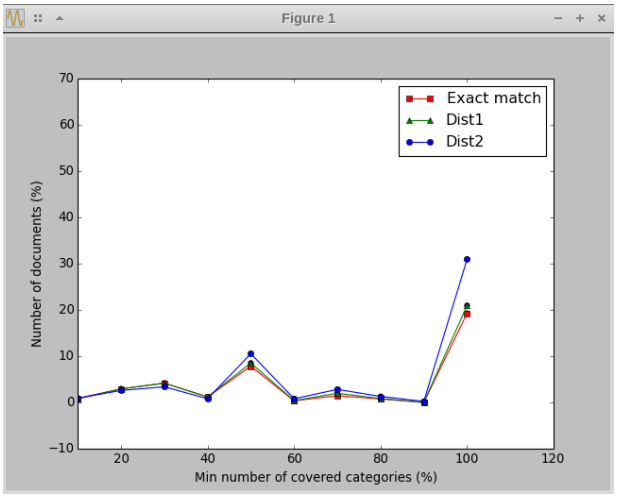
Figure 2

As we can see, the pre-optimized algorithm is able to predict the general concept present in the document sufficiently well (*religion, philosophy*) but most of the top 20 categories are not related to the document. In fact such categories received an high weights due to the fact that they belongs to the list of Wikipedia's major topic classifications and thus they have an extremely large vocabulary.
On the other hand, the two optimizations explained in step 5 aim to smooth this effect, pushing down categories with a large vocabulary and not related to the document and pushing up at the beginning of the top 20 the most characteristic ones. In fact *atheism*, which is the category assigned by the 20 Newsgroups dataset, now is the first one and most of the following are strictly related to the document except for some little mistakes that are still present (*20th century fox films*).

In the second experiment, in order to measure how well our algorithm can predict the original categories, we run it on the body of 1775 Wikipedia articles not contained in the dataset and randomly selected from Wikipedia in exactly the same way as we have described in chapter 2.2.
We kept valid the same experimental setting of chapter 2.2 because otherwise we might have collected articles whose categories are not present in the dataset and obviously our algorithm would have been unable to predict them.

NB: all the categories of Wikipedia articles that are not present in the dataset are ignored while computing the percentage of official categories present in the top 20 categories.

The results of our experiment are shown in the figure below:



The curve "**exact match**" on Figure 1 shows the number of documents for which the top 20 categories contained at least a specific percentage (indicated on the x axis) of the official categories. We see that the proposed method predicted the 50% of the official category for approximately 7% of the documents, and there were about 20% for which all official category was discovered.
If we do not insist that official categories appear directly among the top 20 categories, and allow their substitution by their immediate sub or super categories, we have that according to curve "**Dist1**",accuracy nearly uniformly improves by approximately 2%.
If we further relax our requirements, and instead of one level of indirection, we allow two, accuracy again increases, as shown by the curve labeled "**Dist2**". The improvement is larger this time, which is no surprise, since n levels of indirection mean roughly a^npossible substitutions, where a is the average number of immediate sub- and super-categories for a category.

As we can see from the results, thanks to the experimental setting choice and despite the small size of our dataset and the reduced set of categories, this method is able to predict the original categories of a document quite well.