Final Report

Citation Prediction

Group Number
2

Group Name
AGAS

Group Members:
Animesh Jain

Abstract

Citation Prediction for papers. Our goal is to predict the top-10 references for a given paper published in 2013, based on the information such as abstract, authors, venue, and title of the paper.

Introduction of Background

The goal of the project is to find the top 10 references for all the papers, which got published in 2013. This tool helps researchers to discover relevant references for the particular paper, which is published in the present time. In previous years lot of tools were launched such as Web of Science and JSTOR, offer only simple based hierarchy and keyword-based categorization. These tools often yields poor result when variation in terminologies and re-use of technical terms lead to the poor ranking, many irrelevant results and omission of important references.

We have organized the references in a way that leverages the research work that researchers have already done in a) building their own list of relevant citations, and b) writing with the goal of drawing interest from those in related fields; we used textual analysis to develop a system capable of using substantially more information than simple phrases, i.e. an entire abstract, to better refine results.

Problem Definition and formalization

The problem can be defined in the following way:
Given a set of query papers without references, we need to recommend a set of relevant 10 references for the paper by applying data mining techniques to the candidate training set AP_train.txt.
To formalize the problem, let us consider the case of a single query paper that does not have any references. The query paper q can have a set of 10 candidate papers 'c' ordered from the highest to lowest in terms of relevance to the query paper's context.
So, there is a function **f(q, c)** that maps the candidate paper to the query paper, which is what we are trying to determine through the solution for this problem.
The candidate papers for each of the query papers are considered from the training set based on the following conditions:

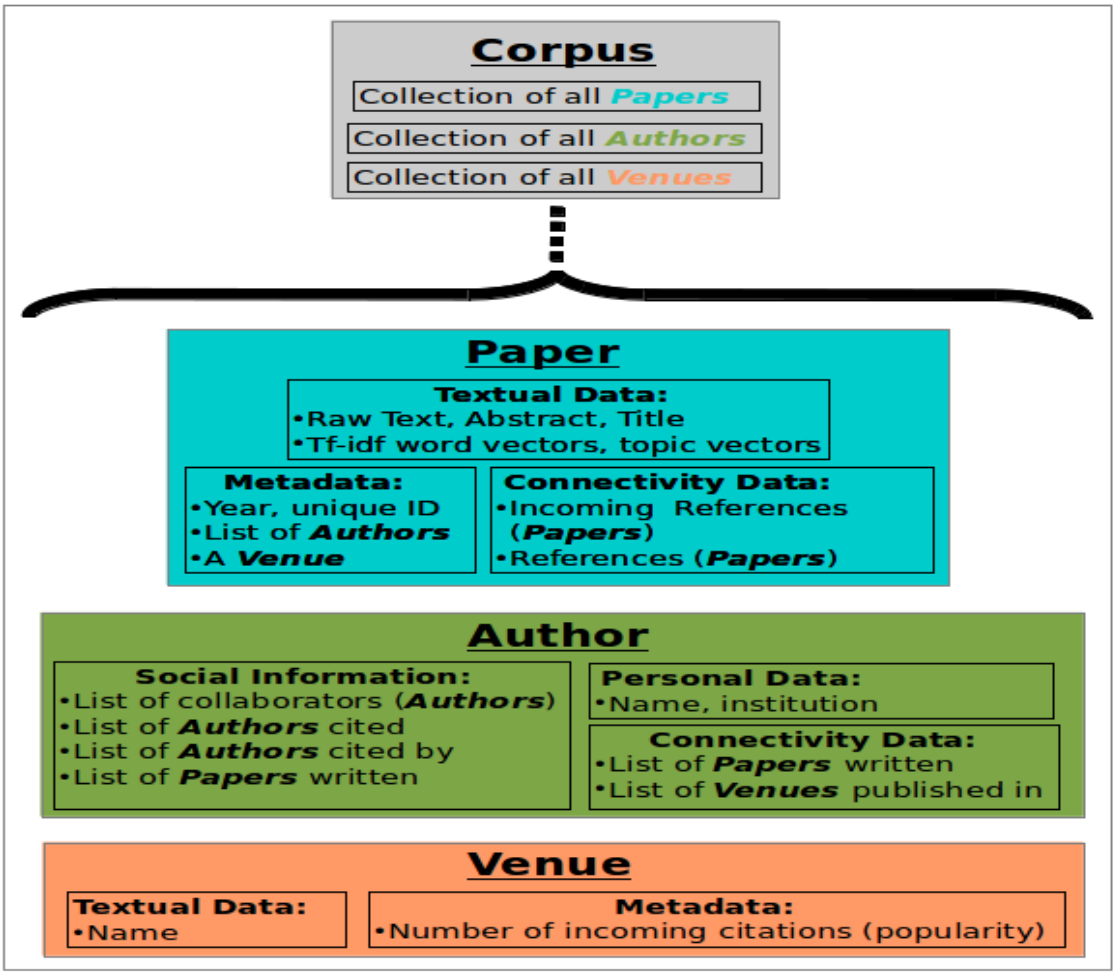
a. The candidate papers must not be dated after the date of publication of the query paper, which the given training set already satisfies.
b. The candidate papers must have a level of context similarity to the query paper.
c. It should be possible to rank the candidate paper based on its "mined" features in order to show its relevance to the query paper.

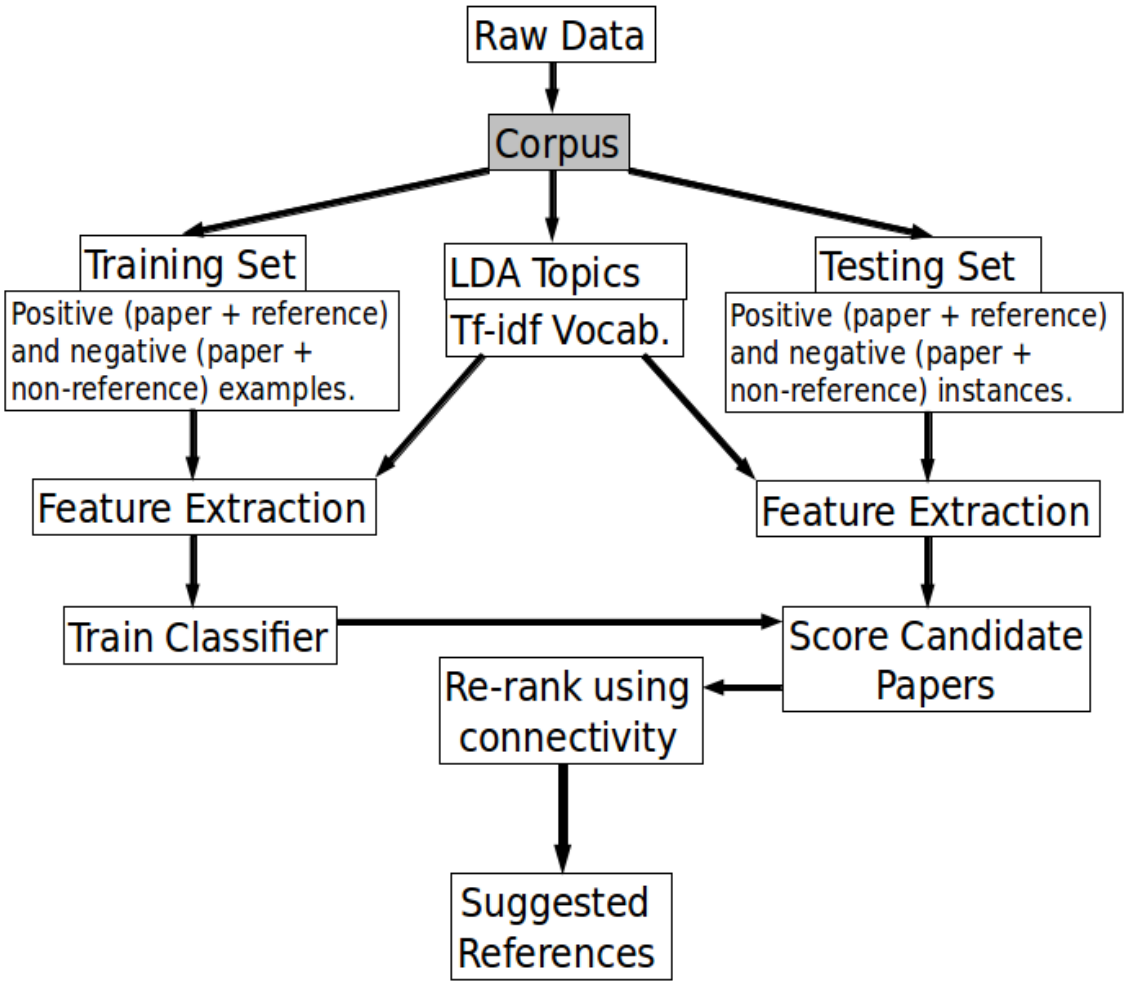
Data Description and Pre Processing

We use the Arnet Miner as our data source. This data set consists of raw text data describing a large set of papers-including authors, venue, publication year, and raw text- and their citations consisting of other papers. In total, the data set contains 19,647 paper, 16,152 authors, and 94,973 citations. A significant portion of our work was concerned with reading the data, cleaning it, and transforming it into usable form.

Steps executed in pre-processing

1. We have used titles, abstract, references and year of publication for each index in the sample data, and created a hash map which composed of key as an Index and values as a object having information on titles, abstract, references and year of publications.
2. We removed all the stop words from the abstract and titles content before setting it into the hash map
3. We converted the text of abstract and titles into lowercase, so it will be easy to implement the cosine similarity on it.




Raw Data
Corpus
Training Set
Positive (paper + reference) and negative (paper + non-reference) examples.
LDA Topics
Tf-idf Vocab.
Testing Set
Positive (paper + reference) and negative (paper + non-reference) instances.
Feature Extraction
Feature Extraction
Train Classifier
Score Candidate Papers
Re-rank using connectivity
Suggested References

Method Description

After preprocessing the data, we go the hash map in the memory, which consists of all the data present in AP_train.txt. After that we create a vector for each hash map key value entry. Now we create the vector for each publication present in the AP_test_par and do the cosine similarity with the each of the vectors entry present in the AP_train.txt. Out of all the similarities we will be going to take top 10 of them.

Working of Tf-Idf and Cosine Similarity

Tf-Idf term weighting

A commonly used term weighting method is tf-idf, which assigns a high weight to a term, if it occurs frequently in the document but rarely in the whole document collection. Contrarily, a term that occurs in nearly all documents has hardly any discriminative power and is given a low weight, which is usually true for stop words like determiners, but also for collection-specific terms; think of apparatus, method or claim in a corpus of patent documents. For calculating the tf-idf weight of a term in a particular document, it is necessary to know two things: how often does it occur in the document (term frequency = tf), and in how many documents of the collection does it appear (document frequency = df). Take the inverse of the document frequency (= idf) and you have both components to calculate the weight by multiplying tf by idf. In practice, the inverse document frequency is calculated as the logarithm (base 10) of the quotient of the total number of documents (N) and the document frequency in order to scale the values.

Vector Space Model

The tf-idf values can now be used to create vector representations of documents. Each component of a vector corresponds to the tf-idf value of a particular term in the corpus dictionary. Dictionary terms that do not occur in a document are weighted zero. Using this kind of representation in a common vector space is called vector space model (Salton et al., 1975), which is not only used in information retrieval but also in a variety of other research fields like machine learning (e.g. clustering, classification). Each dimension of the space - we are far beyond 3D when it comes to text - corresponds to a term in the dictionary. Since documents are usually not of equal length, simply computing the difference between two vectors has the disadvantage that documents of similar content but different length are not regarded as similar in the vector space.

Cosine Similarity Measure

To avoid the bias caused by different document lengths, a common way to compute the similarity of two documents is using the cosine similarity measure. The inner product of the two vectors (sum of the pairwise multiplied elements) is divided by the product of their vector lengths. This has the effect that the vectors are normalized to unit length and only the angle, more precisely the cosine of the angle, between the vectors accounts for their similarity.

$$sim(d_1, d_2) = \frac{\vec{v}(d_1) \cdot \vec{v}(d_2)}{\left|\vec{v}(d_1)\right\|\vec{v}(d_2)|}$$

Documents not sharing a single word get assigned a similarity value of zero because of the orthogonality of their vectors while documents sharing a similar vocabulary get higher values (up to one in the case of identical documents). Because a query can be considered a short document, it is of course possible to create a vector for the query, which can then be used to calculate the cosine similarities between the query vector and those of the matching documents. Finally, the similarity values between query and the retrieved documents are used to rank the results.

Experiments Design and evaluation

Vector selector:

In order to first define the set of features relevant to this analysis, we had to collect **context based feature sets** like citation count to understand a **paper's popularity**, the venue count to get the **popularity of a venue**, the **author count** to see how many papers the author has published and then the **author popularity** i.e. how many references were there for an author's paper. To do the last part, we had to merge the authors' name with that of count of citation and this gave us the **reference count** for each author.
In our design, as we knew that our paper set for the test data was of 2013, **publications earlier than 2000 would have a lesser chance of being relevant** to the current test set. To be on the safe side, we **excluded papers before 1995** from the training set. This training set was still very big to be read into memory or to perform analysis upon. Hence we resorted to considering **textual features** as the next step for filtration as mentioned below:
To get a filtered training set based on text based features, we first filtered the entire set of title + abstract in the training set by removing stop words. We used a stop word-removing algorithm called as **Parser** to do the same. Parser collects nouns, adjectives, verbs and adverbs from the dataset abstracts and titles and returns it in the form of a set of strings. This step helped us in reducing the dataset to a very large extent and also helped us in applying text based feature selection/similarity methods on the dataset for getting better valid feature sets.

Feature evaluation:

The next step was feature evaluation. We took the reduced dataset, loaded this data into a hashmap and started running our feature evaluation algorithm. This step ensured that we were not running into memory issues when we load the entire dataset into memory. But we did run into memory issues and hence we could not proceed with this step.

So instead, we again ran the Vector selection step (step 1) on the test dataset (AP_test_par) and then loaded just that dataset to hashmap memory. In order to follow up our procedure with comparisons like cosine similarity and tf-idf, we started reading the **filtered and reduced** AP_train.txt dataset that we got from the vector selection procedure (step 1), one line at a time, parsed the file to understand whether it was a title, abstract, year, etc. and **ran cosine similarity algorithm** comparing the reduced test dataset with this file based reduced dataset one at a time. We took the cosine similarity measure and it proved to be a satisfactory measure to compare text based features. Taking the values obtained through cosine similarity, we collected the top 10 documents.
For the above 10 documents that we got from cosine similarity measure, we get the references of these documents and form a **hashmap of reference vs count** and rank these references using their popularity (author count, citation count and venue count features).

Conclusions

This project opened our eyes into the real world of data mining. The following are our major takeaways:

1. Data mining is a very large and rigorous process. Much of the effort needs to be put into preprocessing the data and making the raw data usable for further mining and analysis. We had to put in a lot of time trying to find out ways how to preprocess the data quickly and efficiently.
2. We came to know that both context-based features and text-based features need to work in tandem to give us a good mean average precision (MAP). The reason we fell short of a MAP good score was because we underestimated the time we will require to finish this project and had thought that text based cosine similarity would be a whole lot faster, but in fact we were wrong. Also we are pretty certain that had we come up with better context based features like perhaps the co-authorship of a paper or the co-citation between papers, to compare the test set and the main dataset, we would have got a few better results. All in all, we needed more time in the end.
3. We believe that although we did not make much progress in terms of the Kaggle competition, we have learned very good concepts from **Prof. Yizhou** and the **TAs Yupeng and Kosha**. The pre-final submission meeting with the Prof. helped us a lot in understanding what needs to be done for pre-processing as we lacked ideas before that.
4. Owing to this project, we had to read a lot of citation recommendation papers by esteemed researchers in their areas to understand how we can make our algorithm better and faster.

References

1. http://web.stanford.edu/class/cs224n/reports/2013/katesharbeck.pdf
   [Paper describing citation recommendation using both context-based and text-based features.]
2. http://www.cse.psu.edu/~dkifer/papers/citationrecommendation.pdf
3. http://www.sciencedirect.com/science/article/pii/S0012365X04001116
4. http://www.cse.psu.edu/~dkifer/papers/autocitation.pdf
   [A lot of our ideas were similar to what this paper does]
5. http://www.cse.unt.edu/~ccaragea/papers/jcdl13.pdf