Project Proposal: CSE 519 Data Science Fundamentals
Automatically building Book Indices

Objective:

The purpose of the project is to build a tool that applies the principles of data science to automatically build back-of-the-book indices by analyzing the user given text.

Background research:

Creating book indices is a well studied problem and there is existing work in this regard which relies on various forms of keyword extraction and information retrieval. We referred to existing literature on the topic [1], [2], [3], [4] to study the different approaches and considerations taken during the index building procedure.

Although it is easy to think of index building as a keyword extraction problem, there are certain subtleties to address. Consider for instance the important words which are defined in certain parts and later referred in the paragraph, the index must only have page numbers of paragraphs where the indexed word is described or dealt with in detail, ignoring parts which aren't as important.

Another challenge is whether to treat this problem as a supervised learning problem or unsupervised learning problem. A supervised learning algorithm would need a dataset to train on which already has an index generated. Invariably the indices, topics and the style of composition of subjects vary from domain to domain. Thus we would be tied to a specific domain and need considerable dataset on the specific topic. An unsupervised learning approach to this problem is discussed in [1] in which the indexability is modelled around two factors: context-aware informativeness, key phraseness (Likelihood of a keyphrase).

The paper[2] discusses about a supervised way of generating the index list using the term frequency and document frequency as weights. The papers[3][4] talk about the characteristics of keyphrases in the index list and provide different ways of generating the index list based on the importance of a phrase to the document. The paper[3] also discusses about the ways of evaluating the results generated using different methods.

Dataset:

We planned to use arXiv's dataset as it will give latex sources for a good number of papers with indices. This would greatly aid in considering various text formatting as features to learn from. We came across a subset of the arXiv dataset which was released as part of of Citation Prediction Task of the KDD Cup competition, this comprises of all papers in the High Energy Physics theory portion of the arXiv until May 1, 2003. The dataset had 25768 papers in it. However, we found that among these only 167 papers had latex index tag, which means that most of the papers do not contain a index.

Since the pool of papers has such limited data to train on, perhaps an unsupervised approach is more appropriate. Alternatively, downloading the entire arXiv data using the requester-pays subscription model would lend us more insight if we can find sufficient data for a supervised learning approach.

Approach:

The challenge with the back of the book indexing is the generation of multi-word phrases for indexing purposes. This requires keyphrase generation, finding the index-worthy keyphrases, and ranking them based on how important they are to the topic of the paper.

We generate an almost exhaustive list of all the phrases that can be indexed, and filter the phrases that are not good candidates for indexing. We will do this for every input file to the tool. We will n-grams approach with many restrictions[shouldn't contain punctuations, shouldn't cross word borders, shouldn't contain the connectors, shouldn't begin or end with a stopword or a common word etc] that would reduce the number of phrases generated. We will further prune it based on whether they are actually valid phrases. We will also find the main theme of the text file for our initial model. To make the relevance more reliable, we will find the theme of each section and find out how relevant a particular keyphrase is to that section. Once the data is cleaned and the main theme of the paper/section is available, we will create a data set with following features:

- frequency of the keyphrase
- keyphrase is a noun phase [0,1]
- keyphrase is a name entity [0,1]
- keyphrase is part of the heading[0,1]
- keyphrase is part of a subheading[0,1]
- keyphrase has special formatting [0,1]
- relevance of the keyphrase to the main theme of the input file

We have reduced the problem to a binary classification of whether a keyphrase is index-worthy or not. To avoid missing out on good candidates, we are generating an exhaustive list of phrases. We will use a model that predicts the probability of a keyphrase being a good candidate for index. We rank the keyphrases based on this value and generate an index-list from the top n(user input) entries in the list.

Preprocessing is an important stage that decides the accuracy of the prediction. So, much importance and effort is given to cleaning the dataset such that it is suitable for use. First Tokenization will be done on the data. We are going to use the word tokenize from nltk package of python[5]. The process should convert all the words to lowercase and remove the accents before processing. Then parts-of-speech tagger is used so that only key concepts and important names should be indexed. Then Stemming and Lemmatization is done to reduced to the words to its base form. Stop word removal is an important stage in pre-processing. For now NLTK's stopwords corpus is used. Later larger stopwords corpus will be used depending on the results we get.

Handling of special cases should be done. One such case identified so far is consecutive group of words or phrases have to be considered for indexing such as "Computer Architecture and Data science". So when indexing the words, occurrence of consecutive words also has to be tracked[6]. Also words such as "U.S.A" should be matched with alternate form of the word, for example "USA". Choosing the number of index entries has to be taken care. Both over-indexing and under-indexing is bad. Ideally, the size of the index should depend on the size of the document[7]. However, if the user gives a size of the index as input, the model need not take care of this issue.

The pre-processed data is fed into a model as testing set and prediction is made on the passed documents. Pre-processing of text data and model building is tweaked to achieve better results. Plan is to use multiple models for building index. Either the best model will be picked or multiple models will be used to build an Ensemble.

Validation:

One way to validate the book indices is to take a set of publications with pre-existing indices and compare the tool generated results with the existing index. This method suffers from a particular drawback due to the nature by which publication indices are generated; Typically indices are generated by humans, often times by people other than authors of the publication. As such the indices are topics that they perceive to be the most crucial. This "opinion" might vary from one person to another.

Another way to validate the tool would be to compare it against pre-existing tools that already address the index generation problem. Tools such as XYZ can be considered, however we are limited by the accuracy of the tool we are comparing against.

A way to combat the above issues is to distribute a set of papers among people and do a survey asking them to build the index. A single publication can be distributed among multiple people and a common subset among those can be then chosen to be compared against the index generated by the model.

Next steps:

1. Collect data from arXiv site, if needed use book indices as well.
2. Preprocess the data and generate features needed for building model.
3. Perform Feature transformation using NLTK tool in Python.
4. Build a function to generate keyphrases.
5. Build a ranking function and rank keyphrases.
6. Run the scoring system on the test data for further improvisation.

References:

1. http://delivery.acm.org/10.1145/2510000/2505627/p1745-wu.pdf
2. www.aclweb.org/anthology/P88-1025
3. web.eecs.umich.edu/~mihalcea/papers/csomai.flairs07.pdf
4. web.eecs.umich.edu/~mihalcea/papers/csomai.cicling06.ps
5. www.researchgate.net/publication/273127322_Preprocessing_Techniques_for_Text_Mining
6. www.pythonprogramming.net/tokenizing-words-sentences-nltk-tutorial/
7. www.cl.cam.ac.uk/teaching/1314/InfoRtrv/lecture2.pdf