

Proposed solution

The above problem can further granularized into below categories:

1. Extracting the relevant topic (patent filed category) and query extraction
2. Crawling the web and extract results from the search engine
3. Calculating the relevance and extracting the Assignee

Topic and Sub-Topic Extraction:

Topic (field in which patent is filed) is crucial to detect from the document since this effect the search results of the crawl bot. The field of invention section is very helpful to detect the topic and sub-topic of the patent. The following are the steps that can be used to extract topic and sub-topic of the patent.

Step 1:

- Topic is defined as the class a document is classified into (Ex: The sample patented document is of orthopedics class)
- To Extract the topic, we can use Tf-idf (Term Frequency, Inverted Document Frequency) which returns the matrix of term vectors which represents more frequent terms in the document but the terms are present in less number of documents (inverted document frequency)
- But using only Tf-idf may not be enough since tf-idf might return many topics, to further refine and extract relevant topic a Spacy POS tagger or coreNLP POS Tagger can be used to extract Noun Singulars (NNS) from the terms identified by TFIDF (For further filtering stemming, lemmatization, removing stop-words can be used). I used spacy POS tagger in the script.

Step 2:

To extract a subtopic, our task is to find relevant words to main topic. We can use wordnet or word embeddings (Glove or Word2Vec). I used google Word2Vec with 300 dimensions (<https://drive.google.com/file/d/0B7XkCwpl5KDYNINUTTISS21pQmM/edit>) which gives the vectors for a word. Using Gensim similarity we can find which vectors are close to the topic vector and then we can extract the sub topic or query sequence. Then the combination of topic and subtopic are passed to web search which gives results to the search query. Google patents API is preferred but since it is deprecated in 2011, we can use google custom search API to crawl the web pages.

Crawling using Google Custome Search API:

I used Google custom api (<https://cse.google.com/cse/all>) to search for the relevant documents. I created a custom search API, you can find the cx key in the code:

"AIzaSyAD0UzPyyBnhcIPOyuhPiJ_7jQCldtj9FY".

Due to time constraint, I extracted top 10 results, I used BeautifulSoup to crawl the webpages for the tags, headings, meta data etc.

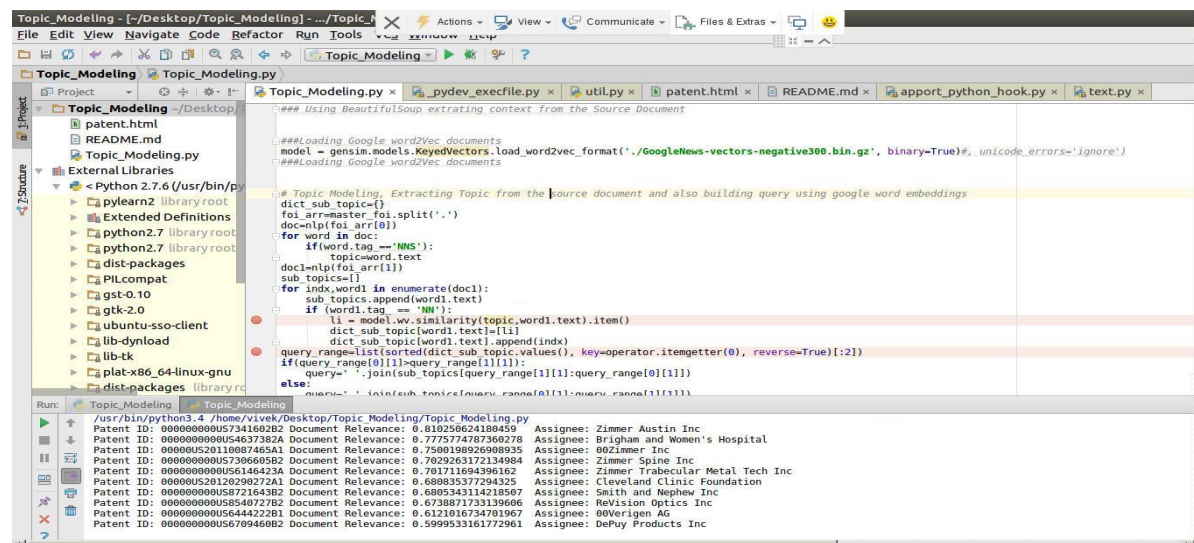
Relevance ranking and Extract published author / Organization:

I used sci-kit learn tool ki to create TFIDF vectors and also get the document similarities, then perform Named entity entity extraction to retrieve the Assignee then use text similarity (Stanford text similarity or python Gensim).

Summary:

With the amount of data presented, my script performs better than the traditional models which are available online. My Neural Language Model takes the advantage of the combination of TFIDF, word embeddings and Spacy tagger. Due to time limitation, I restricted my work to 10 search results, due to data and infrastructure limitations, I used all the default tools available online. This work can be greatly enhanced when more amount of data is available to extract more relevance from the data and retrain word2vec, similarities, TFIDF, Named Entities.

Results:



```
Topic_Modeling - [~/Desktop/Topic_Modeling] - .../Topic_
File Edit View Navigate Code Refactor Run Tools View Communicate Files & Extras
Topic_Modeling.py x pydev_execfile.py x util.py x patent.html x README.md x apport_python_hook.py x text.py x
Project
  Topic_Modeling -/Desktop/
    patent.html
    README.md
    Topic_Modeling.py
  External Libraries
    <Python 2.7.6 (/usr/bin/py
      pylearn2 library root
      Extended Definitions
      python2.7 library root
      python2.7 library root
      dist-packages
      PILcompat
      gst-0.10
      gtk-2.0
      ubuntu-ss-client
      lib-dynload
      lib-tk
      plat-x86_64-linux-gnu
      dist-packages
      library root
    Run:
      Topic_Modeling
      Topic_Modeling
      /usr/bin/python3.4 /home/vivek/Desktop/Topic_Modeling/Topic_Modeling.py
      Patent ID: 0000000005734160282 Document Relevance: 0.810256624180459 Assignee: Zimmer Austin Inc
      Patent ID: 000000000054637382A Document Relevance: 0.7775774787360278 Assignee: Brigham and Women's Hospital
      Patent ID: 000000529110087465A1 Document Relevance: 0.7500198926908935 Assignee: 00Zimmer Inc
      Patent ID: 0000000005730660582 Document Relevance: 0.7029263172134984 Assignee: Zimmer Spine Inc
      Patent ID: 000000000056146423A Document Relevance: 0.70171694396162 Assignee: Zimmer Trabecular Metal Tech Inc
      Patent ID: 000000529120290272A1 Document Relevance: 0.680035377294325 Assignee: Cleveland Clinic Foundation
      Patent ID: 00000000058721643B2 Document Relevance: 0.6805343114218507 Assignee: Smith and Nephew Inc
      Patent ID: 000000000058540727B2 Document Relevance: 0.6738871733139606 Assignee: ReVision Optics Inc
      Patent ID: 000000000056444228B1 Document Relevance: 0.6121016734701967 Assignee: 00Verigen AG
      Patent ID: 00000000056709460B2 Document Relevance: 0.5999533161772961 Assignee: DePuy Products Inc
```

Future work:

Step 1: It can be improved with a dataset clustered into categories as Health, Technology, Electronic. This implies more relevance with respect to each topic

Step 2: The google word2vec can be retrained based on the categories

Crawlers can also be enhanced by adding weight to the specify category results.

Environment Requirements:

- BeautifulSoup – 4.4.1
- Gensim - 3.0.1
- GoogleApiClient
- Spacy – 1.9.0
- Requests – 2.18.4
- Python – 3.5.2
- Scikit-learn – 0.18.2

Note: Possible use of other versions of python might lead to change util.py file of spacy

References:

1. Tf-idf (<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>)
2. Spacy POS Tagger (<https://spacy.io/docs/usage/pos-tagging>)
3. Word2Vec (<https://drive.google.com/file/d/0B7XkCwpI5KDYNINUTTISS21pQmM/edit>)
4. BeautifulSoup (<https://www.crummy.com/software/BeautifulSoup/>)
5. Python Gensim (<https://radimrehurek.com/gensim/index.html>)
6. Scikit-learn (<http://scikit-learn.org/>)