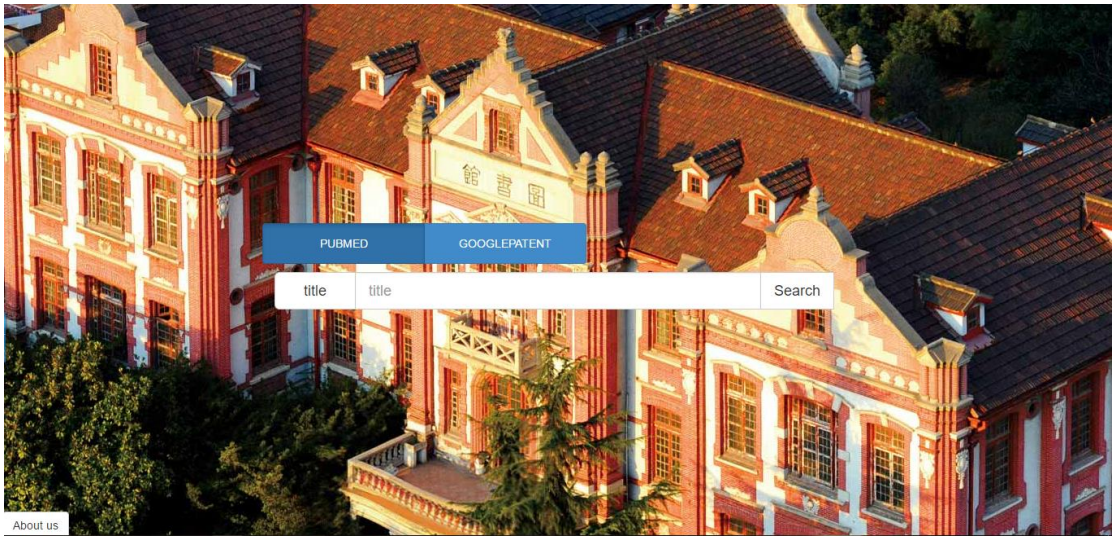
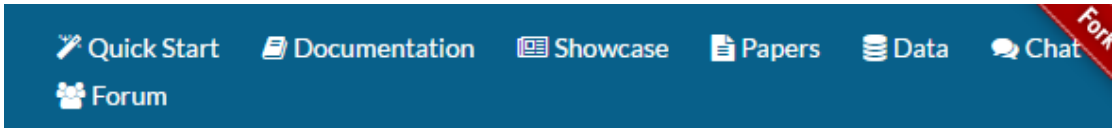
Project 实验报告——任翔远 5130309514

As shown in the powerpoint, I accomplish 3 individual project. They are deepdive, search engine and crawler respectively.

1/



To finish the deepdive, I use the tutorial in the website of deepdive http://deepdive.stanford.edu/ . You can just click the quick start icon to get in.



Besides, the data we handled is shown in the data icon, in which I used Pubmed and Google patent.

PATENT (Google Patents)

Quick Statistics & Downloads

Pipeline	OCR'ed Text > NLP (Stanford CoreNLP 3.5.1)		
Size	428 GB	**Document Type**	Government Document
# Documents	2,437,000	**# Machine Hours**	100 K
# Sentences	248 Million	**# Words**	7.7 Billion
Downloads	Download Full Corpus ▾	Download Small Teaser ▾	

We plan to have DeepDive's PATENT Corpus to contain a full snapshot of patent grants since 1920 from the United States Patent and Trademark Office (USPTO), European Patent Office (EPO), and World Intellectual Property Organization (WIPO), indexed by Google Patents in Feb 2015.

Patent applications we processed belong to the public domain. Information obtained at Jan 27, 2015.

PMC-OA (PubMed Central Open Access Subset)

Quick Statistics & Downloads

Pipeline	HTML > STRIP (html2text) > NLP (Stanford CoreNLP 1.3.4)		
Size	70 GB	**Document Type**	Journal Articles
# Documents	359,324	**# Machine Hours**	100 K
# Words	2.7 Billion	**# Sentences**	110 Million
Downloads	Download Full Corpus ▾	Download Small Teaser ▾	

PubMed Central (PMC) is a free full-text archive of biomedical and life sciences journal literature at the U.S. National Institutes of Health's National Library of Medicine (NIH/NLM). DeepDive's PMC-OA corpus contains a full snapshot that we downloaded in March 2014 from the PubMed Central Open Access Subset.

PMC applies different creative common licenses. Information obtained at Jan 27, 2015.

The main process to tackle it is relatively simple, firstly we need to install it (I do recommend you use the homebrew)

Installing DeepDive

First, you can quickly install DeepDive by running the following command and selecting the `deepdive` option:

```
$ bash <(curl -fsSL git.io/getdeepdive)
```

```
### DeepDive installer for Mac
+ curl -fsSL https://github.com/HazyResearch/deepdive/raw/v0.8.x/util/install/install.Mac.sh
1) deepdive                5) postgres
2) deepdive_examples_tests  6) run_deepdive_tests
3) deepdive_from_release    7) spouse_example
4) deepdive_from_source
# Select what to install (enter for all options, q to quit, or a number)? 1
```

You need to have a database instance to run any DeepDive application. You can select `postgres` from DeepDive's installer to install it and spin up an instance on you machine, or just run the following command:

```
$ bash <(curl -fsSL git.io/getdeepdive) postgres
```

Alternatively, if you have access to a database server, you can configure how to access it as a URL in the application's `db.url` file.

Tips: it may takes days to install it in Linux, but hours in Mac
After installation, we just need the following instructions to complete it, remember deepdive is well designed to handle dark data and do the NLP

```
$ deepdive query '?- articles("5beb863f-26b1-4c2f-ba64-0c3e93e72162", content).' format=csv | grep -v
'^$' | tail -n +16 | head
```

```
$ deepdive query '?- sentences("5beb863f-26b1-4c2f-ba64-0c3e93e72162", _, _, tokens, _, _, ner_tags,
_, _, _).' format=csv | grep PERSON | tail
```

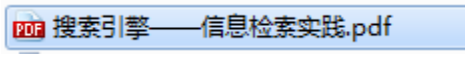
```
$ deepdive sql "
    SELECT p1.mention_text, p2.mention_text, expectation
    FROM has_spouse_label_inference i, person_mention p1, person_mention p2
    WHERE p1_id LIKE '5beb863f-26b1-4c2f-ba64-0c3e93e72162%'
      AND p1_id = p1.mention_id AND p2_id = p2.mention_id
  "
```

The postgre is the SQL database we use. For the reference of the long time setting for the search engine and database, you can look at the project shown by others, it is a quite standard process. And the search engine used for deepdive is Lucene.

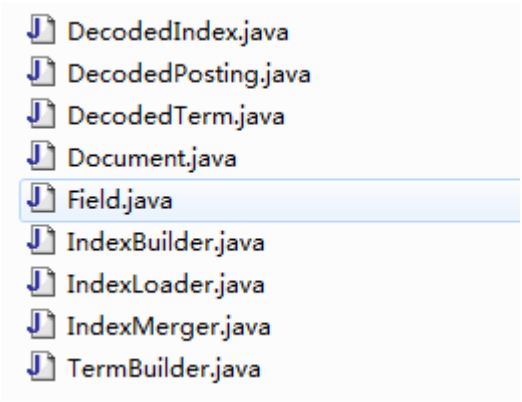
2/

The Search engine we designed for Acemap is named lucid girlfriend. We mainly focus on several field in the paper, the title , author, content, abstract. We can even enlarged it to more field with few changes of the source code.

The main difference in term of code from other search engine is in the indexing and ranking. Indexing is all the invert- indexing. But we need to split it into two part, first which field the term is in, second where it appears. For the ranking algorithm we use the BM25. The information of it quite illustrated in the book < search engine- the information retrieve> Here is its Chinese version.



The code within the PaperSearchEngine is mainly in the src/main.
The analysis is the text transformation part, I do the stemmer, stopwords and tokenizer.
The indexing code is separated quite uniformly.



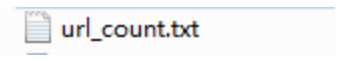
And the Ranking part

Finally, you run the Demo.java to do the search.
All the code designed is very time-consuming, but the algorithm and idea within it can be easy to understand If you read the book mentioned above and use the indexing to understand what a field is like.

P.S. The text transformation part is finished by ZhangXi, a group member of Data Group.

3/

The concept of crawler has been illustrated too many times by our group members. The url.content.txt is the work allocated to each one.



The main part crawler can be separated into 2 parts. The Generator and Parser, which generates the URL we need to crawl and crawl the special content within the URL respectively.

The crawl folder is the code to do something interesting. You can run it to see the effect (surprise included). Besides, the ieee-Parser.py and spider.py are the main code.
Actually, we design the code similar to each other, since we studied together and discuss the uniformed steps together, so I recommend you read the report of ShiZhenfeng, who do the crawler of a Japanese website very very very neatly.

The following code is what I designed for the URL :
But the ip is all forbiddened after crawl it for about 10thousand papers.