TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

-----*-----**

FINAL REPORT

UNIT OF STUDY CS336: INFORMATION RETRIEVAL

<u>Topic:</u>

PROJECT 1: TEXT RETRIEVAL

<u>INSTRUCTOR:</u> ***PhD. Ngo Duc Thanh***

<u>Team Member:</u> **Trương Đình Đức Trí – MSSV: 19522395**

Đặng Xuân Mai – MSSV: 19521820

Lê Văn Trí – MSSV: 19521043

Phạm Nguyễn Thanh Hà – MSSV: 19521458

TP. HCM – 2022

Table of Contents

SECTION 1: INTRODUTION

1.1 Program introduction

Nowadays, with the growth of information and technology it has made a huge impact on every aspect of our life. It has been considered as one of the decisive factors in the activities of the governments, organizations as well as companies, it plays a very important role, can create a breakthrough in power.

With the continuous development of computer technology and electronic networks, as well as the ease of access to data on the internet, many text and images databases have been created in applications that use the Internet, various uses such as educational, industrial, medical, social and many different areas of life. With the development of many databases this has increasing the need for searching and retrieving different types of text information, also the enormous growth of database has caused an overload problem this has been a motivation and led to the development of powerful text search engines that meet almost the maximum needs of people. Effort and time have been devoted to improving the search method.

Stemming from the above awareness, with the same interest and knowledge, the group chose to go in-depth, learn and build an application on the topic ***"Building a text search engine" .***

With limited experience, knowledge and skills, ... the team tried their best to build the most complete text search engine within their capabilities. However, it is still unavoidable to make mistakes and unnecessary mistakes. We are looking forward to your recognition and suggestions to help us improve the above application.

1.2 Dataset

To build a text search engine, having a database of text documents is essential. With instructor support, the group is able to approach the dataset with many different topics to serve the text search engine. Detailed information about the text database that the group has approach in order to build the search engine is as follows:

- Total number of documents in dataset: 43.301

Data collections is available at this link: Links

1.3 Instructions for use

This part is a short introduction that explains how the web works also as help users to clone the repository from Github to run at local device and adjust code.

We use streamlit library to develop our information retrieval system because it is easy to use, supports many widgets to interacts with users [1].

Link to our web: Information Retrieval System

Link to Github repository: IR repository

We assume that all needed code has been prepared [2]. Here are some main files:

- streamlit_app.py: code to display data, interact with users.
- helper.py: contains code to run model, evaluate model, … (execute but doesn't show for users see – like the backend of the web). The details of model will be explained in Section 3.
- file_names_vietnamese_full.npy: because we developed our model on Google Colab, so the order of files (read from Google drive) is different from the order at local folder, so we store the name file to get exact file's name when needed.
- tf_idf_vietnamese_ful.sav: saved tf-idf transformer will help users don't need to wait too long to get the result.
- doc_vector_vietnamese_full.npz: the already transformed vector (vectorized bases on content from whole dataset).

Note: If you want to use the code which we submitted on courses.uit.edu.vn, please add all data (which you gave us) in directory "news_dataset" or else the code won't work. Or you can just clone the complete repository and run it.

If you want to run at local devices, edit code, … please follow these steps:

1. Install streamlit with "pip install streamlit".
2. Clone IR repository.
3. Open terminal in that cloned directory and run "streamlit run streamlit_app.py".

And below is our website and we will illustrate its function.

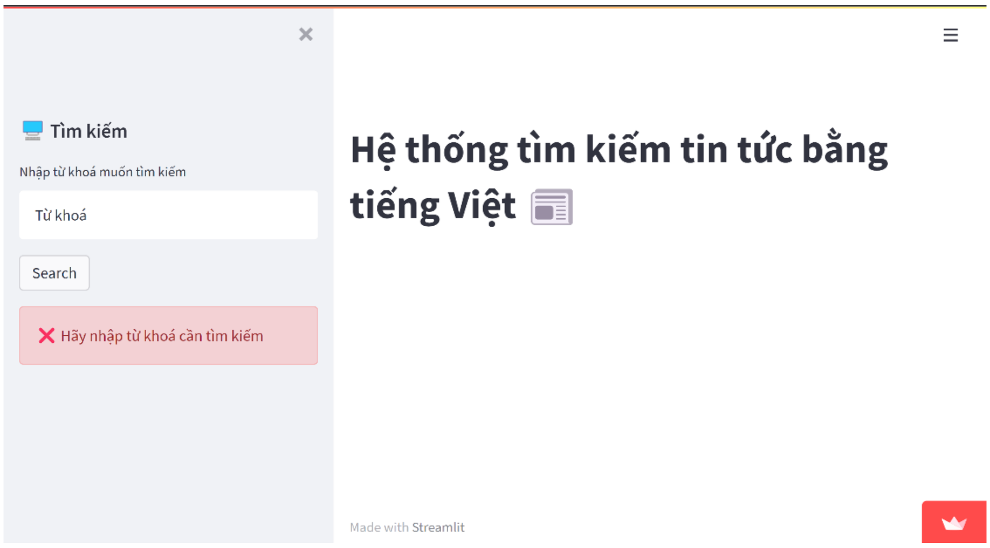

Tìm kiếm
Nhập từ khoá muốn tìm kiếm
Từ khoá
Search
Hãy nhập từ khoá cần tìm kiếm
Hệ thống tìm kiếm tin tức bằng tiếng Việt
Made with Streamlit

Figure 1 - User interface

This is the first apperance of our web, users will enter the *query* on the search box in the sidebar (the left part – and users can also close it after search).

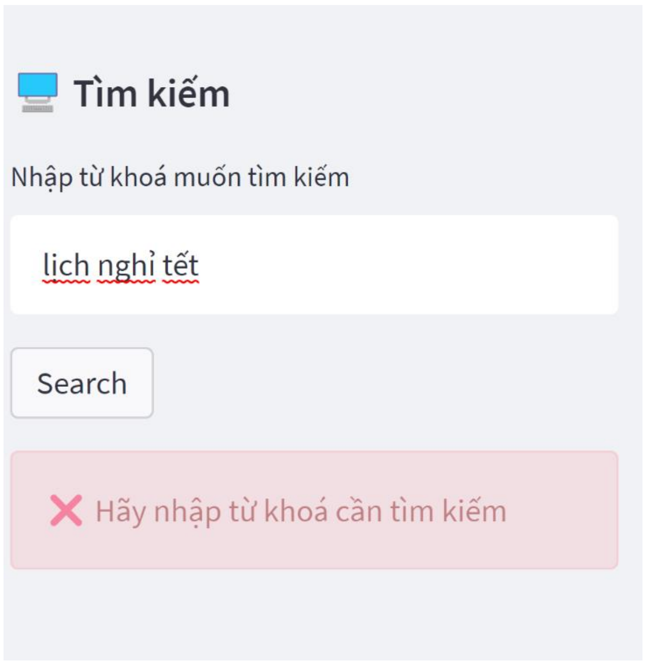
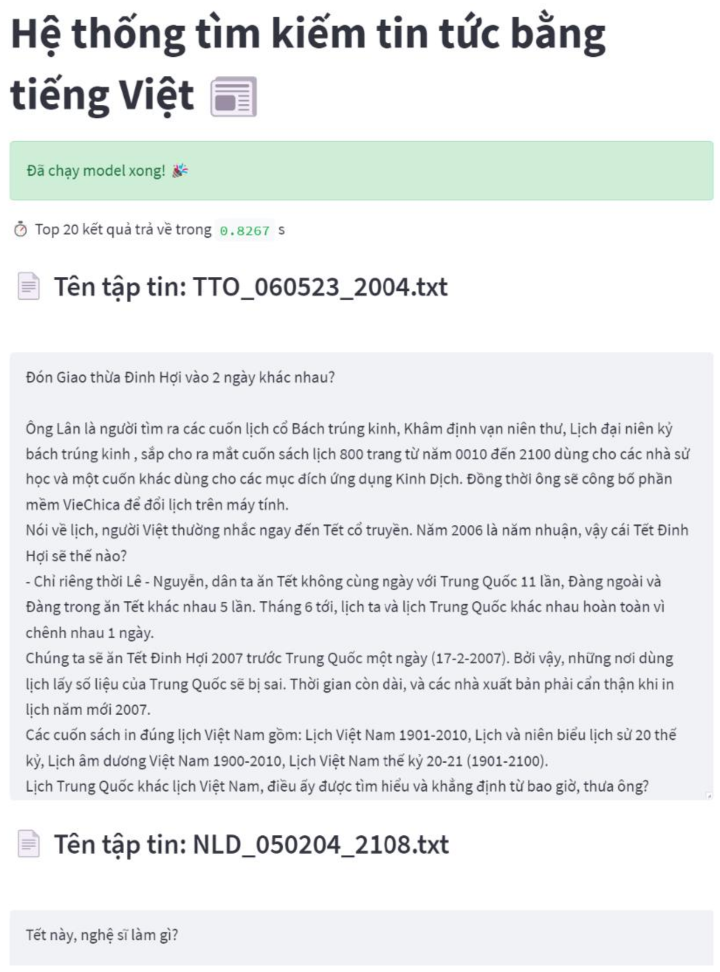


Figure 2 - Enter query

When you want to search, you can click the Search button or press Enter on keyboard. After that, top 20 relevant files will be shown on the right part (Figure 3).


Hệ thống tìm kiếm tin tức bằng tiếng Việt

Đã chạy model xong!

Top 20 kết quả trả về trong 0.8267 s

Tên tập tin: TTO_060523_2004.txt

Đón Giao thừa Đinh Hợi vào 2 ngày khác nhau?

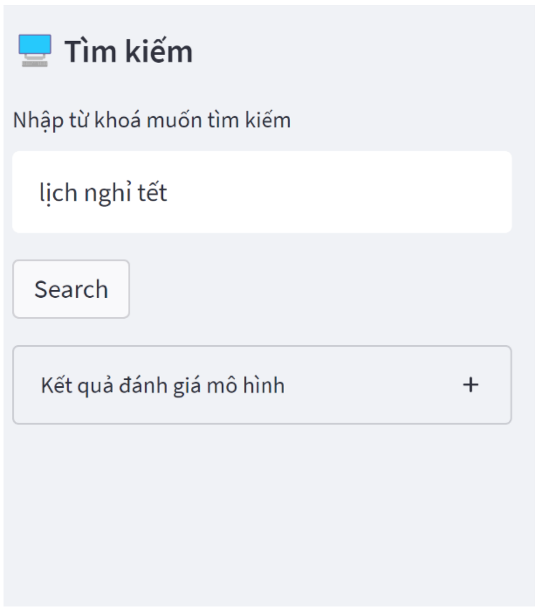
Ông Lân là người tìm ra các cuốn lịch cổ Bách trúng kinh, Khâm định vạn niên thư, Lịch đại niên kỷ bách trúng kinh , sắp cho ra mắt cuốn sách lịch 800 trang từ năm 0010 đến 2100 dùng cho các nhà sử học và một cuốn khác dùng cho các mục đích ứng dụng Kinh Dịch. Đồng thời ông sẽ công bố phần mềm VieChica để đổi lịch trên máy tính.
Nói về lịch, người Việt thường nhắc ngay đến Tết cổ truyền. Năm 2006 là năm nhuận, vậy cái Tết Đinh Hợi sẽ thế nào?
- Chỉ riêng thời Lê - Nguyễn, dân ta ăn Tết không cùng ngày với Trung Quốc 11 lần, Đàng ngoài và Đàng trong ăn Tết khác nhau 5 lần. Tháng 6 tới, lịch ta và lịch Trung Quốc khác nhau hoàn toàn vì chênh nhau 1 ngày.
Chúng ta sẽ ăn Tết Đinh Hợi 2007 trước Trung Quốc một ngày (17-2-2007). Bởi vậy, những nơi dùng lịch lấy số liệu của Trung Quốc sẽ bị sai. Thời gian còn dài, và các nhà xuất bản phải cẩn thận khi in lịch năm mới 2007.
Các cuốn sách in đúng lịch Việt Nam gồm: Lịch Việt Nam 1901-2010, Lịch và niên biểu lịch sử 20 thế kỷ, Lịch âm dương Việt Nam 1900-2010, Lịch Việt Nam thế kỷ 20-21 (1901-2100).
Lịch Trung Quốc khác lịch Việt Nam, điều ấy được tìm hiểu và khẳng định từ bao giờ, thưa ông?

Tên tập tin: NLD_050204_2108.txt

Tết này, nghệ sĩ làm gì?

Figure 3 - Show relevant files [3]

Just scroll down to see files.

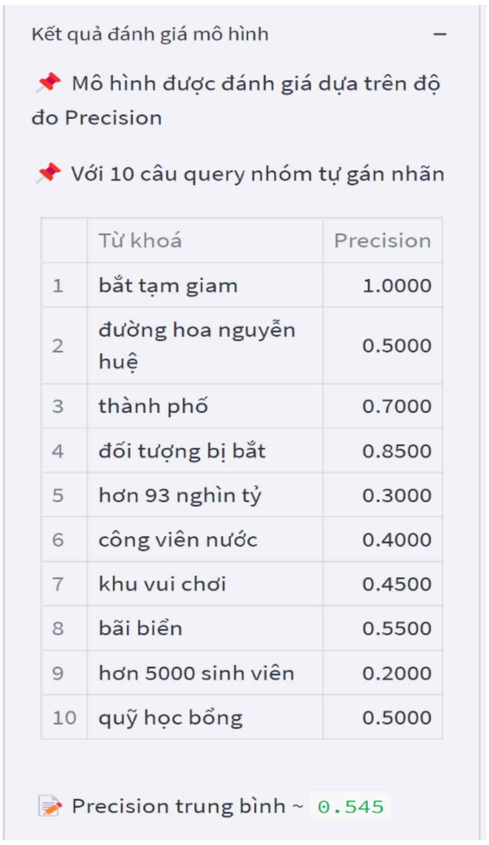

Tìm kiếm

Nhập từ khoá muốn tìm kiếm

lịch nghỉ tết

Figure 4 - Evaluation Result

After run search, on the sidebar will have an extra information – the Evaluation of our model bases on 10 queries we chose and annotated by team’s member – Hà. It’s just an extra so if users want to see it, click on it and the full information will be displayed. If not, users just let it there.


Kết quả đánh giá mô hình –

📌 Mô hình được đánh giá dựa trên độ đo Precision

📌 Với 10 câu query nhóm tự gán nhãn

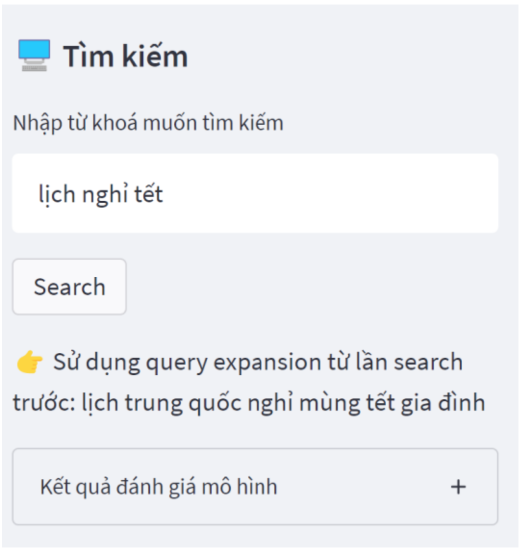
	Từ khoá	Precision
1	bắt tạm giam	1.0000
2	đường hoa nguyễn huệ	0.5000
3	thành phố	0.7000
4	đối tượng bị bắt	0.8500
5	hơn 93 nghìn tỷ	0.3000
6	công viên nước	0.4000
7	khu vui chơi	0.4500
8	bãi biển	0.5500
9	hơn 5000 sinh viên	0.2000
10	quỹ học bổng	0.5000

Precision trung bình ~ 0.545

Figure 5 - Full evaluation result

And to improve system, we use query expansion directly on the second search. When users click on the url to use this website, it will store words which were searched, if users close web tab, and click on url again, all cache is removed.

For example, we are testing on query = ‘lịch nghỉ tết’, if users click (or press Enter) to search again, they will receive this note.


Tìm kiếm

Nhập từ khoá muốn tìm kiếm

lịch nghỉ tết

Search

👉 Sử dụng query expansion từ lần search trước: lịch trung quốc nghỉ mùng tết gia đình

Kết quả đánh giá mô hình +

Figure 6 - Using query expansion

Just keep in mind that if you don’t close the web tab, search history will be saved. So if you search ‘lịch nghỉ tết’, ‘hội hoa xuân’, …. Then comeback to search ‘lịch nghỉ tết’, the expanded query will be used instead, and you still receive that note.

SECTION 2: WORK ASSIGNMENT

After group discussion, it was determined that there are few big things to do in this subject project: designing the interface and coding the program's functions. With four members, the team decided to assign the part of interface design to member Dang Xuan Mai, the part of coding the search function for the program by members Truong Dinh Duc Tri and Le Van Tri. Member Pham Nguyen Thanh Ha was assigned to annotate growth for the program's results. Job details are shown in the table below:

Member's name	ID	Mainly Job	Detail
Trương Đình Đức Trí	19522395	Code Function	- Code the main search function for the program. - Learn and edit the program to suit the processing of Vietnamese. - Manage and assign tasks to team members.
Lê Văn Trí	19521043	Code Function	- Code the main search function for the program. - Learn and edit the program to suit the processing of Vietnamese. -Code query expansion function for the program.
Đặng Xuân Mai	19521820	Design Interface	-Code the entire web interface for the team. -Propose evaluation methods for the program.
Phạm Nguyễn Thanh Hà	19521458	Annotate program's result	Annotate the growth-truth for program's result.

SECTION 3: METHODS OF IMPLEMENTATION

3.1 Tokenizer

Most of documentation on the internet is based around processing English. But in this program we have to deal with Vietnamese language. Vietnamese language is much more complicate than English. We can take "Compound word" in Vietnamese as example:

- Word: "biết bao nhiêu"
    - ❖ "Biết": In English which means as "know".
    - ❖ "Bao nhiêu": In English which can be understood in english as "How much".
    - ❖ "Biết bao nhiêu": But the entire combination has different meaning, it can be understood as "really really much".

→ So just with some simple combinations it already has 3 different meaning. Even if with us can be confused sometimes.

We can't apply the basic tokenizer from NLTK library (***NLTK is a standard python library that provides a set of diverse algorithms for NLP. It is one of the most used libraries for NLP and Computational Linguistics***) to split word in Vietnamese and produce a vocabulary for us to use. More than that we still have to use ***TfidfVectorizer*** or ***CountVectorizer*** to split word for us and create a Vietnamese vocabulary. So due to this problem we will have few options that we can teach scikit-learn vectorizers segments our Vietnamese language. The technique that we use is just to use a custom tokenizer.

We use our tokenizer from the "underthesea" library [4] to token our Vietnamese documents. Then we override a TfidfVectorizer's tokenizer with our custom tokenizer.

```
from underthesea import word_tokenize
from sklearn.feature_extraction.text import TfidfVectorizer
text=['Tôi tên là nam','hôm nay trời thật đẹp']
def tokenize_everydoc(doc):
 tokens=word_tokenize(doc)
 return tokens
vectorizer = Tfidfvectorizer(tokenizer=tokenize_everydoc)
X=vectorizer.fit_transform(text)
```

We apply this process to every document that exist in our corpus and the query. Then we will have all the token and are close to build our library to create a concept space but this is not enough we will have to delete some stopwords in our corpus also our query. This will be discuss in the part below.

3.2 Stopwords

As you already know the stopwords we have to use will totally different from the basic stop word we use from NLTK library. The scikit-learn they don't support Vietnamese stopwords so we have to use ours own. Fortunately, we have someone already built Vietnamese stopwords[5] so we just use them. We create a definiton of stopword than remove them from every documents in our corpus and in our query.

```
def remove_stopwords(docs)
clean_text=[]
for words in docs:
     if words not in stopwords:
                clean_text.append(words)
   return clean_text
```

After we already tokenized all documents and query then we remove those stopwords now we can fit our result into vectorizer to turn our all documents and query into vectors and create a concept space after that is to use consine_similarity and calculates which documents are close to query and display them.

3.3 Query Expansion

3.3.1 Basic ideal: automatic relevance feedback

Clusters: known relevant documents contain terms which can be used to describe a lager cluster of relevant documents.

Obtain a description for a larger cluster of relevant documents automatically: identifying terms which are related to the query terms; synonyms, stemming variations (in Vietnamese language, we ignore it), terms which are close to the query terms in the text, etc.

3.3.2 Global analysis

All documents in the collection are used to determine a global thesaurus-like structure which defines term relationships. Then, this structure can be shown to the user who selects clusters or terms for query expansion.

3.3.3 Local analysis

The documents retrieved for a given query q are examined at query time to determine terms for query expansion. So local clustering and local context analysis are without assistance from the user.

Local clustering operates solely on the documents retrieved for the current query which leads to some advantages and disadvantages. It is valuable because term distributions are not uniform across topic areas: distinguishing terms are different for different topics. Conversely, global techniques cannot take these differences into account. But local clustering requires significant run-time computation so it isn’t suitable for Web search engines due to cost (our project use web interface, not web search). In particular, this solution is useful in intranet environments and for specialized documents collections.

There are 3 types of clusters: association, metric and scalar clusters. After carefully considering the whole situation (simple to code, use less time and able to return diversity), we decide to implement association clusters in local analysis to expand query.

3.3.4 Implement association clusters

Stemming and Lemmatization are the basic text processing methods for many languages as English. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. Fortunately, this problem doesn’t exist in Vietnamese so we can skip it when we use the available solution.

There are some definitions before we build query expansion:

q: query

D_l: local document set (retrieved documents, in our project, we use the first 20 results the system returns)

V_I: vocabulary of D_I

S_I: set of distinct stems for V_I (As mentioned above, the steming isn't necessary in Vietnamese language. So S_I are V_I)

In this section, we apply association clusters solution. Assocciation clusters's idea: terms which co-occur frequently inside documents likely relate the same concept. So we can find the way to implement. Firstly, simple computation based on the frequency of co-occurrence of terms inside documents-correlation between the terms:

$$c_{u,v} = \sum_{d_j \in D_I} f_{s_u,j} \times f_{s_v,j}$$

In above equation, we just compute correlation on a couple term. Representing them as matrices is simpler to deal with the rest terms. Matrix m has $|S_I|$ rows and $|D_I|$ columns, $m_{ij} = f_{s_i,j}$ (frequency of term s_i in document d_j). Matrix m can be gotten from using CountVectorizer in Sklearn library. Next, the correlation matrix s will be completed by under equation:

$$s = mm^T$$

The values in this matrix favors more frequent terms. A normalized correlation or association matrix is more useful because it tends to terms which are more rare. Normalized score is 1 if two terms have the same frequency in all documents:

$$s_{u,v} = \frac{c_{u,v}}{c_{u,u} + c_{v,v} - c_{u,v}}$$

Finally, for each term in query, expand query with the n terms, with the highest value of $s_{u,v}$. Union of all query term clusters created by original and new ones in above process is expanded query. (in our project, we set n=1).

In our test, it takes about 10s to retrieve documents completely instead of instant when we use query expansion. Most of the time is spent on normalization. In order to reduce execution time, we can use the unnormalized form of s. Then, unnormalized factors tend to group terms due to large frequencies.

SECTION 4: EVALUATION AND ANALYSIS

We will use Precision metric to evaluate this model. Team's member will search each query in query list that our team created, and human labeled which document is relevant or not. We can’t use other metrics like MAP or F1 because we can’t define all relevant files in the corpus, and precision is intuitive in the situation that we don’t really know the data well.

To have ground truth files, the annotater has to do like we metioned above, and stored files in order our team decided to get access easier.

The construction of ground_truth directory directory's link is:

```
|-- ground_truth
|   |-- list_query.txt
|   |-- ground_truth
|   |   |-- gt_1.txt
|   |   |-- …
|   |   |-- gt_10.txt
```

With each line in the list_query.txt is one query our team prepared, and each gt_number.txt is ground truth file (contains relevant files’ indices, each index is on a single line) of the number-th query in the list_query.txt file.

Pseudocode for function to evaluate our model, this piece of code is for evaluate one query, for example this is for the first query, gt_1.txt is the correct ground truth file.

```
1 true_relevant = 0
2 ground_truth_index = readline(gt_1.txt)
3 relevant_indices = Run_Model(query)
      # our model will return a list of indices of relevant files in dataset, not
      # the content of relevant files
4 for each_relevant_index in relevant_indices:
      # Iterate through the predicted result by model
5     if each_relevant_index in ground_truth_index:
6             true_relevant += 1
7 precision = true_relevant / 20  # Because we use top 20 results
```

And like the result we put on the web, the evaluation result is:

Table 1 - Evaluation Result

Query	Precision
bắt tạm giam	1.00

đường hoa nguyễn huệ	0.50
thành phố	0.70
đối tượng bị bắt	0.85
hơn 93 nghìn tỷ	0.30
công viên nước	0.40
khu vui chơi	0.45
bãi biển	0.55
hơn 5000 sinh viên	0.20
quỹ học bổng	0.50

And the average of the system ~ 0.545.

The query "bắt tạm giam" is equal to 1, we think because the data from news contents, this query is very common, so it can get this high precision. On the other hand, 'hơn 93 nghìn tỉ' and 'hơn 5000 sinh viên' are too specific, the number part make the system hard to find the correct relevant files (nghìn tỷ and sinh viên are common in news) so there precision aren't high. That is our opinion about this evaluation.

SECTION 5: CONCLUSION

We know how to build an information retrieval system step by step, can applied query expansion to improve system althought this is still not an ideal system. We can use our knowledge to choose suitable metric for our model.

SECTION 6: REFERENCES

[1] Streamlit API references link.

[2] Model Development using Streamlit link

[3] Display Vietnamese text - unicode characters link.

[4] Underthesea library link

[5] Vietnamese stopwords link