Evaluation of IR models

Project Part B

Mihir Kulkarni
Computer Science and Engineering
University at Buffalo
Buffalo, NY-14214
UB Person number: 50168610
mihirdha@buffalo.edu

Ajinkya Thorve
Computer Science and Engineering
University at Buffalo
Buffalo, NY-14214
UB Person number: 50167854
ajinkyay@buffalo.edu

***Abstract*— The project evaluates Information Retrieval models like VSM, BM25 and LM based on various measures like F-measure, MAP and nDCG using Apache Solr. The performance of the models is evaluated using TREC evaluation. While evaluating these models we change the model's hyper parameters and track the corresponding changes in the evaluation measures. Using these changes we then suggest optimal parameters for each model.**

***Index Terms*— VSM, BM25, Language Model, Information Retrieval.**

I. Introduction

Information Retrieval is performed by selecting appropriate documents for the given query. But, in real world IR systems, merely showing the relevant document is not enough. This is because, users of typical IR systems rarely go beyond first few results to get the information. So, it has become very important to rank the relevant documents according to their relevance to the query. This is achieved by implementing various IR models and ranking documents according to them. Vector Space Model (VSM), Best Matching (BM), Language Model (LM) are the IR models implemented in our project. Effectiveness of these models is judged by various evaluation measures. We are using F-Measure, Mean Average Precision (MAP), Normalized discounted cumulative gain (nDCG) in the project to judge the effectiveness. We find these measures by comparing our calculated relevance judgments with the ideal relevance judgments given. We are using TREC_eval script to calculate these measures. The IR models have some parameters which when changed, change the effectiveness according to different measures. We observe the results after changing these parameters and select the parameter set which yields best results according to the evaluation measures.

We are also given test queries for which relevance judgment is not given. We find results for these queries using the parameter set which has yielded best result on the training set.

II. Dataset

We are given a dataset of tweets in json format. These are the tweets about Syrian refugees. We index this data and perform search queries on it.

Along with this, we are also given 14 training queries and their corresponding relevance judgments. We use this training data to train the parameters of our IR models so that the best result is achieved.

We are also given test data without the relevance judgments. We find the results for these queries using our trained IR model and submit the results.

III. Evaluation Measures

We are using F-Measure, Mean Average Precision (MAP) and Normalized discounted cumulative gain (nDCG) to evaluate our models.

F-Measure or F_1score is the harmonic mean of precision and recall. It is given by the formula-

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R} \quad \text{where} \quad \beta^2 = \frac{1-\alpha}{\alpha}$$

Where α or β can be modified to give more importance to either precision or recall.

Mean average precision is an evaluation measure known for good discrimination. It is calculated as-

$$\mathrm{MAP}(Q) = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j}\mathrm{Precision}(R_{jk})$$

Where Q is the document set and $\{1\ldots m_j\}$are relevant documents in Q.

Normalized Discounted Cumulative Gain (nDCG) is calculated as-

$$\mathrm{NDCG}(Q,k) = \frac{1}{|Q|}\sum_{j=1}^{|Q|} Z_{kj}\sum_{m=1}^{k}\frac{2^{R(j,m)}-1}{\log_2(1+m)},$$

Where, Q is the document set and $\{1 \ldots m_j\}$ are relevant documents in Q. Z_{kj} is normalization factor.

IV. Vector Space Model

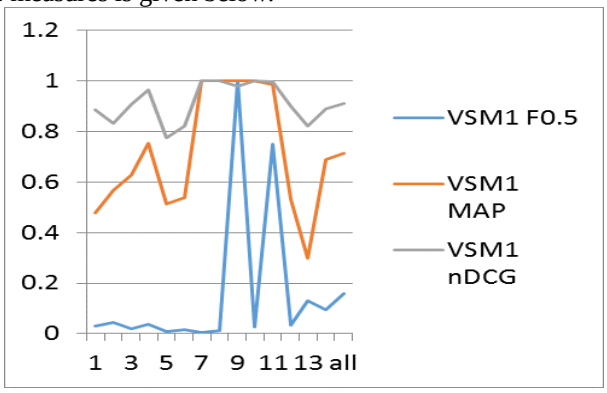
In the vector space model, documents and queries are represented as vectors, and it uses the *tf-idf* weighting scheme. The *tf-idf* weighting scheme assigns to term *t* a weight *tf-idf* in document *d* given by

$$\text{tf-idf}_{t,d} = \text{tf}_{t,d} \times \text{idf}_t$$

In our case, tf-idf assigns to term *t* a weight in document *d* that is

- highest when the term *t* occurs many times within a small number of tweets (thus lending high discriminating power to those tweets)
- lower when the term occurs fewer times in a tweet or occurs in many tweets (thus offering a less pronounced relevance)
- lowest when the term occurs in virtually all documents

Initially we use the queries directly as they are, obtain the search results, and calculate trec_eval on it. We have calculated three evaluation measures: F0.5, MAP, and nDCG. The plot for these measures is given below.

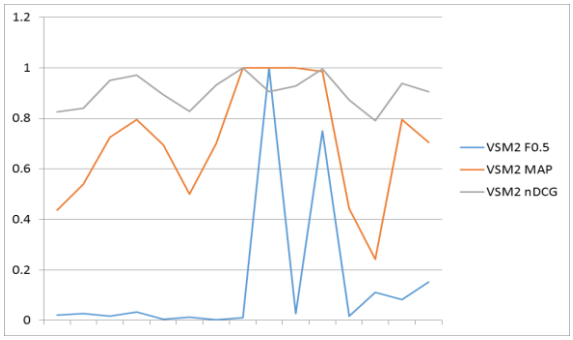


To improve the score further we have applied measures at index time as well as at query time. Index time measures are common to all models and are given below in separate section.

Query time measures taken to improve the score are-

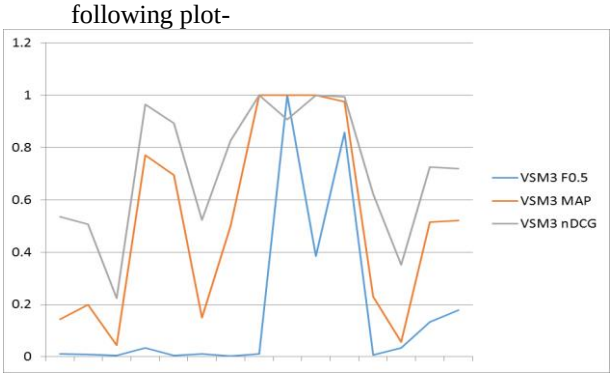
- Using translation on the queries

  The method we used above is a basic way of querying, and it can be optimized. Since the queries and documents are in three languages, it makes sense to specify the text fields while querying. That is, English query can be searched in text_en, German in text_de, and Russian in text_ru, and we can 'OR' all the results obtained for a query. Using this approach and calculating trec_eval on it, we get the following plot:



- Using translation and boost factors on the queries

  Using another approach, we can boost certain queries. That is, English query can be boosted in text_en, German in text_de, and Russian in text_ru. For example, if the original query is in English, we have used a boost factor of 5 when it is searched in text_en, while leaving default boost factors for German and Russian queries. Similar approach is used when we encounter queries in German and Russian. Using this approach and calculating trec_eval on it, we get the following plot-



Thus, it is seen that approach number three (using translation and boost factors) did not yield the expected results. Approach number two (using simple translation on the queries) gives better performance only in some cases. Hence, for VSM, we will use the test queries directly as they are.

V. BM25 Model

BM25 is a family of scoring functions which retrieves documents based on term frequency and inverse document frequency. That means, BM25 treats each document as bag of words and ignores the relationship between query terms in the document.

In BM25 model, score for document D for query set Q i.e. $\{q_1, q_2 \ldots q_n\}$ is calculated using the formula-

$$\text{score}(D,Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})},$$

Where,

f (q_i , *D)* is term frequency of q_i in document D. IDF (q_i) is inverse document frequency of q_i. k_1 and b are parameters which can be varied to adjust the model.

We are implementing BM25 in Solr using similarity class

"org.apache.solr.search.similarities.BM25SimilarityFactory"

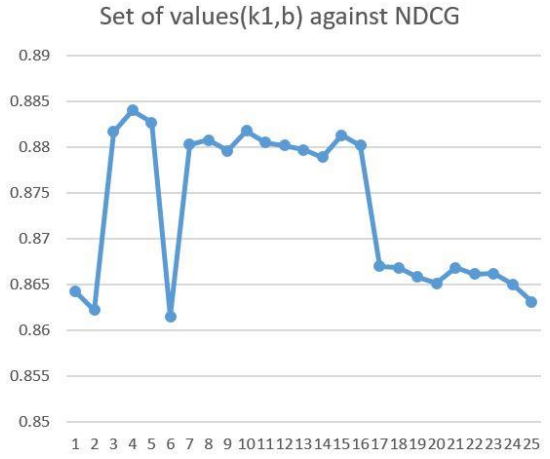
Using the model, we find results for each training query given. This result is then compared with the expected result given in qrel.txt using trec_eval script. Trec_eval script generates score for each query and for the model as whole. This score can be used to find the optimal model using techniques explained below.

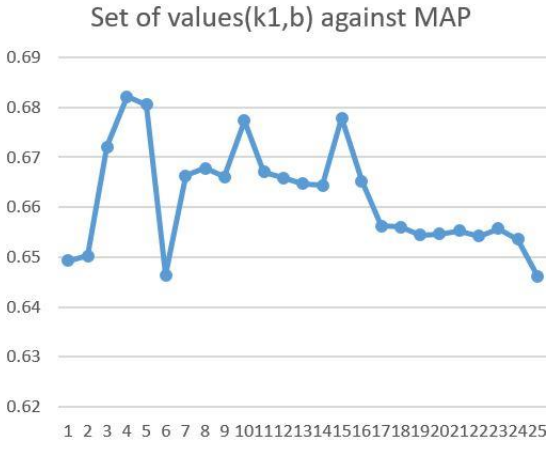
A. Grid Search

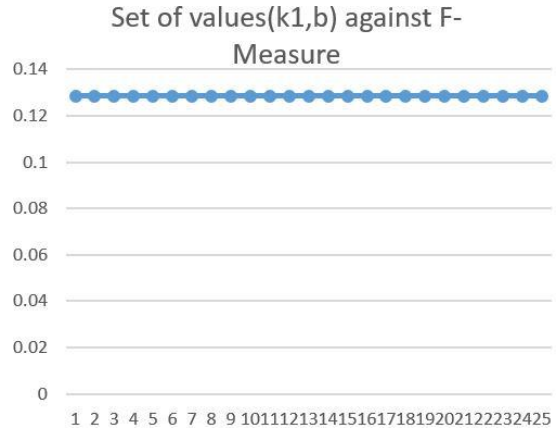
We change values of k_1 and b in order to get better results from our model and in turn get better trec_eval score. We perform a grid search on these parameters.

For each set of values in the grid search, we get the results and evaluate trec_eval score for all three evaluation measures.

We then plot graphs of each of these evaluation measures. Each value on X axis represents a set of values of k_1 and b for which the evaluation measure is calculated.







From the graphs we can see that F-Measure value doesn't change. This is expected because, F-Measure depends n precision and recall and ignores the scoring of the result documents. Our parameters k_1 and b affect the scoring of the retrieved documents.

NDCG and MAP both reach their maximum value for sample 4. In the sample 4 we had used k_1 as 1.0 and b as 0.8. So, we conclude that the maximum score can be achieved by keeping these values of the variables.

So, for further use and for getting results of the test queries, we keep k_1 =1.0 and b=0.8

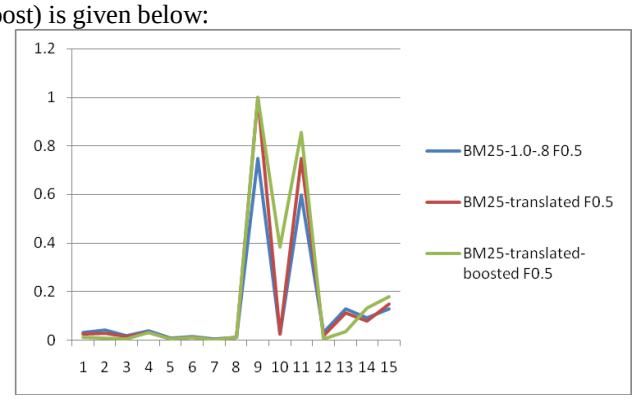
B. Language Translation

Similar to the approach used in VSM, we can try using translated queries to improve the search results. That is, we can translate the queries and English query can be searched in text_en, German in text_de, and Russian in text_ru

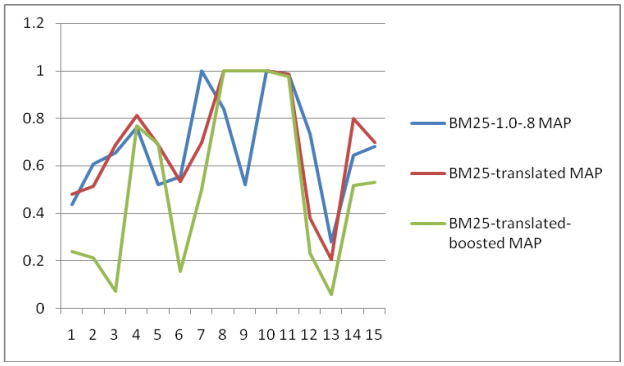
C. Boosting

We can also try to boost certain queries. Here, if the original query is in English, we have used a boost factor of 5 when it is searched in text_en, while leaving default boost factors for German and Russian queries. Similar approach is used when we encounter queries in German and Russian.

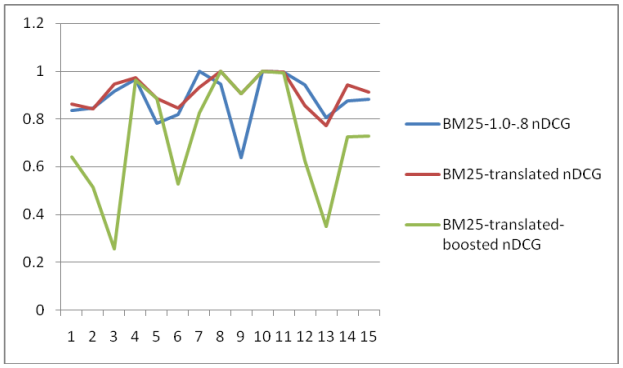
The plot for basic queries (without translation, without boost) is given below:



After using translation, we get the following plot:



After using translation and boosting, we get the following plot:



We can see that there has been an improvement in the results after translation. However, for boosting we did not get much improvement. As a result, we will use translated queries while implementing BM25 on the final test queries.

VI. Language Model

Language model in Information retrieval follows the principle that- A document is best fit for the query if that document is likely to generate the query.

Language model builds a probabilistic language model to find results for the query.

If our query is $q = q_1q_2...q_n$ and $d = d_1d_2...d_m$ id document set, then *p(d,q)* is the posterior probability according to which we need to rank documents. It is probability of document d given a query q. To find this we calculate prior probability *p(q,d)*. It is the probability of generating query q given a document d.

Posterior probability can be found using prior probability by Bayes' rule

$$p(d,q) \propto p(q,d)\, p(d)$$

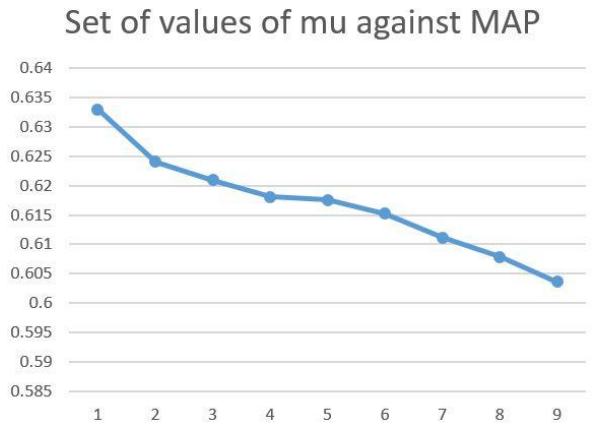
p(d) is the prior belief that document is relevant to any query. This can be assumed uniform so that it does not affect ranking.

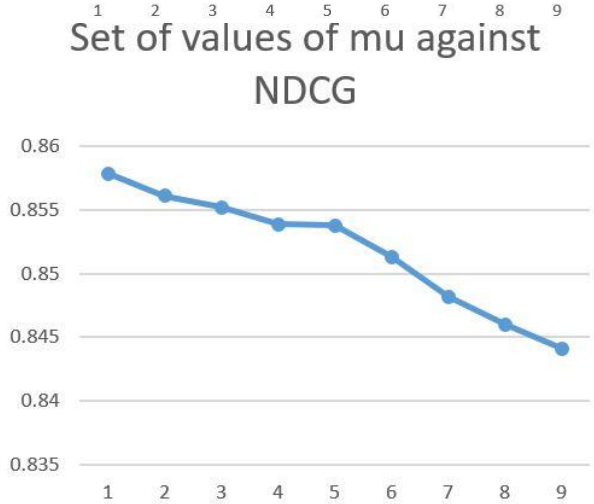
To implement the Language model in Solr, we are using class

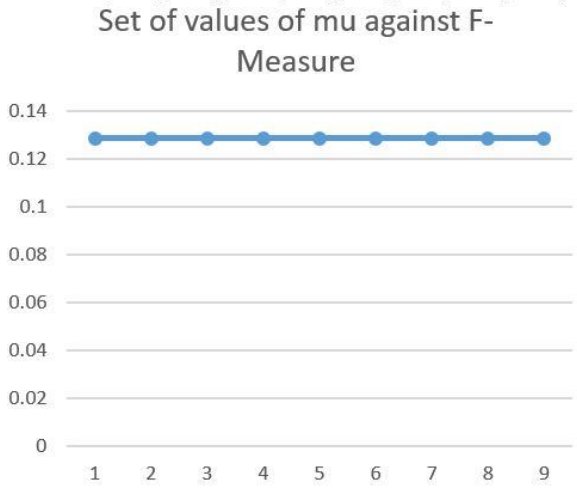
"org.apache.solr.search.similarities.LMDirichletSimilarityFactory"

Using the model, we find results for each training query given. This result is then compared with the expected result given in qrel.txt using trec_eval script. Trec_eval script generates score for each query and for the model as whole. This score can be used to find the optimal model using techniques explained below.

A. Grid Search







From the graphs we can see that F-Measure value doesn't change. This is expected because, F-Measure depends on

precision and recall and ignores the scoring of the result documents. Our parameter mu affects the scoring of the retrieved documents by smoothening.

NDCG and MAP both reach their maximum value for sample 1. In the sample 1 we had used mu as 1. So, we conclude that the maximum score can be achieved by keeping mu as 1.

So, for further use and for getting results of the test queries, we keep mu =1.0

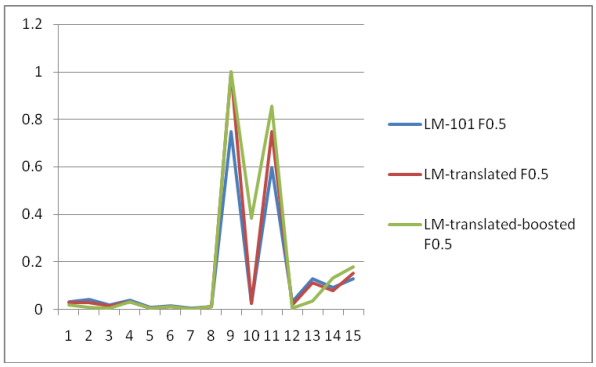
B. Language Translation

Similar to the approaches used in VSM and BM25, we can try using translated queries to improve the search results. That is, we can translate the queries and English query can be searched in text_en, German in text_de, and Russian in text_ru

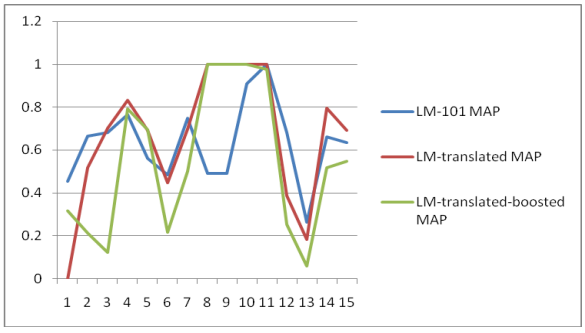
C. Boosting

We can also try to boost certain queries. If the original query is in English, we have used a boost factor of 5 when it is searched in text_en, while leaving default boost factors for German and Russian queries. Similar approach is used when we encounter queries in German and Russian.

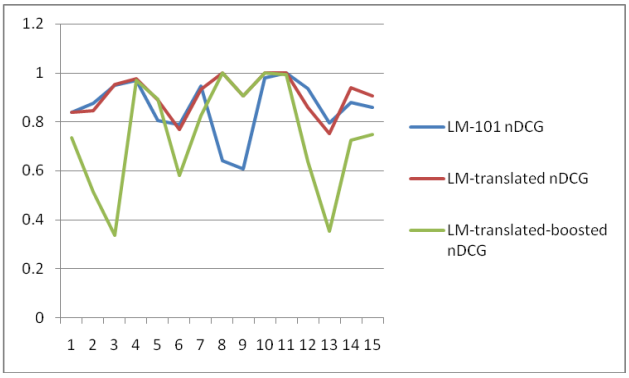
The plot for basic queries (without translation, without boost) is given below:



After using translation, we get the following plot:



After using translation and boosting, we get the following plot:



From the plots, it is observed that there has been a slight improvement in the results after translation. However, for boosting we did not get much improvement. Hence, while implementing LM model on the final test queries, we will use translated queries without boosting.

VII. Index Time Score Improvement

During indexing, we are using the following filters to improve the score-

- StopFilterFactory: Certain words like ‘the’, ’is’, ‘a’ etc do not describe any useful content. During indexing these non-significant words (stop words) are removed from the document vector, so the document will only be represented by content bearing words.
- LowerCaseFilterFactory: Converts any uppercase letters in a token to the equivalent lowercase token.
- PorterStemFilterFactory: A set of mapping rules that maps the various forms of a word back to the base, or stem, word from which they derive.
- UAX29URLEmailTokenizerFactory: Emits "<URL>" and "<EMAIL>" type tokens.

VIII. Analysis

1. Analyze TREC_eval result on different models. Which one performs better overall and why?

We have implemented Vector space model, BM25 model and Language model in Solr.

Each of these models have their merits and demerits. Each of these models give better performance in some or the other circumstances.

For example, BM25 treats documents as bag of words and ignores relationship between the terms. So, it is useful when we are not concerned about the relationship of the terms with each other.

Language model is a probabilistic model and finds out prior probability of generating the query given a single document. This is particularly useful when we are not performing any term-relation specific retrieval. This usually also needs a large dataset to give better performance.

Our dataset is relatively small. VSM is a model which maps documents and queries in vector space and determines how closer a particular document to the query is. We think that VSM is best for our project and works best on our queries.

2. What have you done to improve the performance?

To try and improve the results, we have tried two approaches on each of the models

1. Performed grid search on the hyper parameters of each model to train the model so that it gives optimal performance.
2. Translated queries in languages other than query language.(explained results above under each model)
3. Boosting of certain words on the translated queries. (explained results above under each model)
4. Removing stop words from each language.
5. Used tokenizer other than standard to improve performance.
6. Applied index time score improvement measures explained in the previous section

3. How does the performance change when using different setups? You can use a graph to illustrate how the system performance changed.

When using different setups and approaches, the performance of the system changed considerably. It has been illustrated using different graphs for each individual model above. We have also discussed how we have improved the system based on the changes in the performance.

4. What measure gives a better evaluation of the system and why?

Here, we are using three measures namely- F-Measure, Mean average precision and NDCG.

F-Measure depends on precision and recall and ignores the ranking of the document. So it is not a good measure when ranking is important to the system.

MAP is mean average precision. It is calculated by calculating the precision for each query and taking average.

NDCG is normalized discounted cumulative gain. It is best indicator of the system's performance. This is because statistical definition of NDCG which is mentioned above gives us how much we are closer to expected result and in turn better score.

REFERENCES

[1] "http://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-ranked-retrieval-results-1.html"

[2] "https://en.wikipedia.org/wiki/Okapi_BM25"

[3] "http://nlp.stanford.edu/IR-book/html/htmledition/language-models-for-information-retrieval-1.html"

[4] Class material and TA's help.