COMP6714 Information Retrieval and Data Mining

Project Part 2 Report

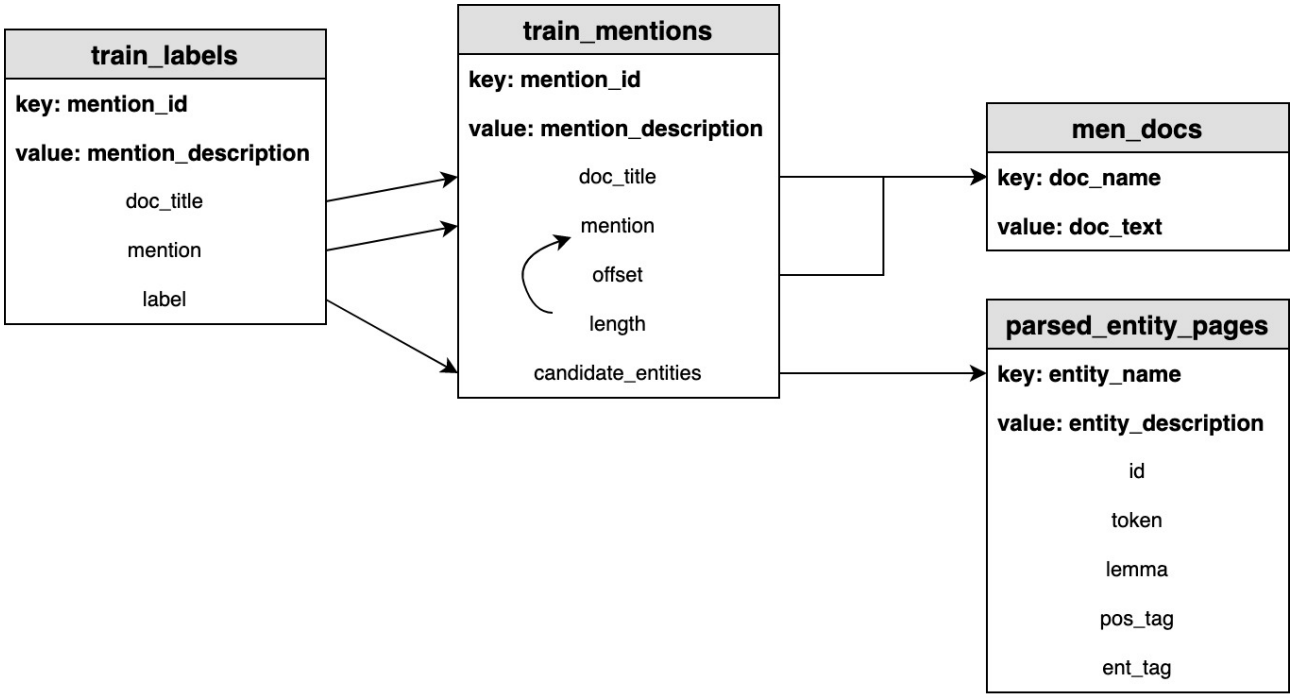
Name: Wenke Yang
ZID: z5230655

Q2: Named Entity Disambiguation/Named Entity Linking

Step 1. Understand the relationship between inputs

Input files:

train_mentions: training examples
men_docs: document base
parsed_entity_pages: parsed wikipedia pages of each candidate entity
train_labels: correct labels of mentions in given documents



Step 2. Calculate tf-idf use project part 1 index

Since the specification recommends to start with using tf-idf as the features, I have choose to calculate tf-idf for **mention, entity name, entity description tokens** in **men_docs**. Each has term frequency count from given doc titles's document, inversed document frequency count from all document base.

Besides the basic tf-idf above, I have added **tf-idf difference in mention and entity name**, **tf-idf difference in mention and entity description** into account. This because the tf-idf for mention indicates the importance of the mention tokens in the given document, while the tf-idf for entity name and description indicates their importance in the given document. If the importance of

mention and entity name and/or description is similar, they would have higher chance to represent similar meanings. Therefore, here I calculated 5 different tf-idfs(3 basic, 2 diffence) in total.

In order to calculate tf-idf, we first build the inverted index by creating the InvertedIndex class from part1 and pass given document base men_docs, and then, calling max_score_query to calculate tf-idf. In part1, we split the query and find the best split, however, this cost too much time to split in part2 due to the high volumn of data, therefore, we skip the find the best split part in part2. For example, when calculating tf-idf for mentions, we split the mentions into tokens by space, and pass only one split to the max_score_query to get the tfidf score.

To try other variations of tf-idf, another parameter **score_type** is added, this parameter indicates the **tf-idf would be calculated as average, maxima or sum**.

Step 3. Train and test xgboost model

In part2, XGBoost model is used to train the dataset. The training dataset consists of three parts: training features, training groups and training labels. The training groups is defined as one group per mention_id, each group has size as number of candidate entities for this mention_id. The training features are the corresponding feature values for each (mention_id, candidate_entity) pair. All features are gathered by get_feature() function with each column as a different feature. The training labels are the corresponding best candidate_entity(label) for each (mention_id, candidate_entity) pair. Here's the dimension of three parts of training dataset:

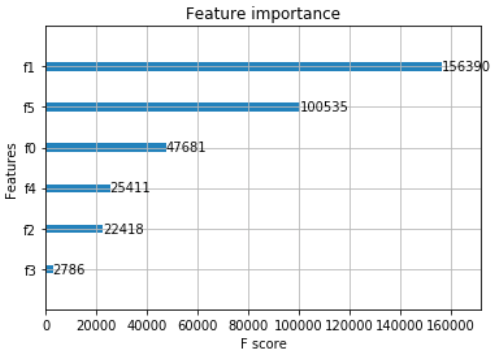
	number of rows	number of columns
training features	# of (mention_id, candidate_entity) pair	# of features
training groups	# of mention_ids	—
training labels	# of (mention_id, candidate_entity) pair	—

Then, train the XGBoost model with the above training dataset, the result is also gathered as scores assign to each (mention_id, candidate_entity) pair. The highest score is selected for each group as the final estimated label, the accuracy is calculated according to the number of estimated label that matches the given training label among all training examples.

Step 4. Select more features, analyse results, train and test, and repeat

At this stage, more features are selected according to the exploration of the given dataset. The matching of mention tokens and the entity name tokens was found useful. If mention tokens and the entity name tokens match exactly, they have a high possibility to match each other. Therefore, a new feature mention_ent_match is added. The mention and entity name are splited by space and underscore respectively.

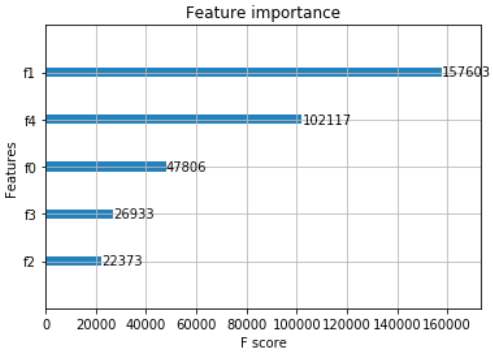
Then, the model is trained and tested by selecting all tf-idf features mentioned in Step 1 and the new matching feature. Here's the accuracy for development datatset 1 and feature importance plotted by xgboost library:



Dev accuracy = 0.6232394366197183

6 Features, each as follows:
f0 - tfidf diff in mention and candidate entity name
f1 - tfidf diff in mention and candidate entity description
f2 - mention match entity name
f3 - tfidf mention
f4 - tfidf entity name
f5 - tfidf entity description

From the above plot, we can see that feature f3, tfidf of mention, has the lowest feature importance: 2786. The feature f3 only shows the relationship between mention tokens and the original document which has no obvious relationship with the candidate entity. Therefore, from the perspective of both logical analysis and results gathered, feature 3 is better to removed. The below feature importance plot is the result after removing the original f3. The accuracy for development set is still ~ 0.62, same as with f3. This proves that f3 is a useless feature.



Dev accuracy = 0.6232394366197183

5 Features, each as follows:
f0 - tfidf diff in mention and candidate entity name
f1 - tfidf diff in mention and candidate entity description
f2 - mention match entity name
f3 - tfidf entity name
f4 - tfidf entity description

Continue doing feature selection

Due to the low accuracy gathered, chosen features need to be improved and more features need to be extracted from the given dataset.

As to the tf-idf results, by doing experiments and exploring the dataset, average of tf-idf is found to be the most reasonable value to be choose. Since the number of tokens are different from one example to another, for example, a description page can have 3000 tokens while another page only has 300 tokens, average can eliminate some effects caused by unbalanced size.

As to the tfidf based on document, it is found that a mention is extracted from the given document, but mention does not have to be the main idea of the document. For most cases, the mention is only relevant to some sentences near the offset of mention in the document. Therefore, local document index is built by selecting the sentence where the given mention is located. However, the accuracy of model by selecting local document tfidf does not perform very well. This might because the rest of document also has some relevant information that local document index cannot reflect. Also, the range of local document index is hard to determine. After trying multiple thresholds of determing local document, the paragraph that the mention is located is chosen. By combining the local paragraph tfidf and the full document tfidf, the accuracy has an obvious increase, especially for development set 2.

Besides, the matching of the mention and entity name feature can be improved by choosing both side as lowered case. This also contributes an increase in accuracy for both datasets.

Furthermore, the F1 score, takes precision and recall into account, is found to be useful for evaluating the degrees of entity description matching the mention.

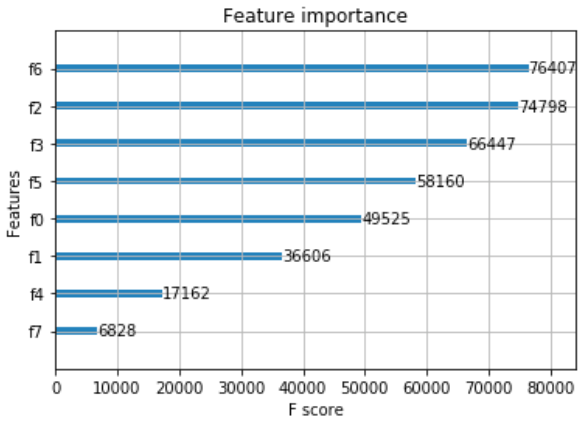
F1 score = (2 * precision * recall) / (precision + recall)
precision = TP / total number of objects classfified
= dscp_TP / len(description_tokens)
recall = TP / total number of relevant/correct objects
= mention_TP / len(mention_tokens)

Here, TP stands for true positives, that is the number of tokens in description/mention classified as relevant and is actually relevant. In our example, since description tokens can have duplications, we take the precision TP as the number of description tokens that appear in mention tokens. For recall TP, we take the number of mention tokens appear in description tokens.

Final selection of features and testing results

Since the full feature selection process is repeating and redundant, here I would skip the process and list only the final features selected and the accuracy of both development set and development set 2.

Dev accuracy = 0.8908450704225352 Dev 2 accuracy = 0.7763975155279503



8 features in total:

f0 - tfidf diff in mention and entity name in doc
f1 - tfidf entity name in doc
f2 - tfidf entity description in doc
f3 - tfidf entity description in local doc
f4 - mention match entity name
f5 - tfidf mention in description
f6 - mention description F1
f7 - mention entity name pos-tag match

Step 5. Tune hyperparameters

The given parameter set = {'max_depth': 8, 'eta': 0.05, 'silent': 1, 'objective': ‘rank:pairwise', 'min_child_weight': 0.01, 'lambda':100}

At step 5, we would like to tune the hyperparameters to increase the accuracy. First, xgboost’s cross validation function xgboost.cv was tried, but it does not support group function. Instead gridsearch is used. It tries all given ranges of combinations of parameters and narrow down the range, finally find the best parameter combination. However, after performing the grid search, the original parameter set is already the best parameter set for the features selected. Therefore, the final hyperparameters are selected same as original set.

Besides, ‘subsample’ parameter is tuned to 0.8 to reduce overfitting. This parameter indicates the percentage of dataset chosen to train the model. In our case, use 80% to train each tree gives the best performance.