Assignment2: Text classification based on the recognized entity URIs

***Abstract*—Text classification is the process of assigning documents (ie: web page, library book, media articles, gallery etc...) to one or more predefined categories. It has widely studied and has been one of the most typical task in supervised machine learning (ML). In this assignment, the classification task is the same with assignment1: classify the cleaned documents extracted from Wikipedia into "Artificial Intelligence (AI)" and "No-AI" categories. The difference between the two assignments is the feature extracting method in data pre-processing process. In this assignment, each document was converted into a bag of words model using recognized entity Uniform Resource Identifier (URI) instead of using ngrams. Then, those entity URIs from each document was ranked by TF-IDF scores. The same classifiers were applied to classify the documents based on the extracted features. We compare the solution using entity URIs to the previous assignment, considering accuracy, precision, recall, F1 score of the validation files classification.**

I. Introduction

TEXT classification is a typical task which has been widely studied by [1], [2], [3], etc.. In these papers, several new approaches has been proposed to improve the performance of complicated text classification (ie: Long-Short Term Memory, Convolutional Nerual Network, etc.).

In this project, the same classifiers as assignment1 were selected to solve the same binary classification problem: classify the documents extracted from Wikipedia into "AI" and "No-AI" categories. The pipeline of this project is the same with previous one except in the data pre-processing part, the recognized entity URIs[4] was used as features for classification task.

The reminder of the report is organized as follows: Section II describes the pre-processing of extracting recognized entity URIs features from documents for classification. The details of extracting recognized entity URIs pipeline is introduced in Section III. The two groups (using ngrams and using URIs as features respectively) of evaluation results of different classifiers algorithms under different benchmarks are provided in Section IV. Section V provides a brief conclusion.

II. Data Pre-processing

In order to classify text, the categorized texts need to be firstly extracted and characterized into the format which could be understood by the ML methods. In this section, the data format was firstly converted from arff to text for sklearn framework. The data format processing is described in subsection II-A. Then, the text documents were pre-processed into characterized vector by bag-of-words model. The bag-of-words models are built by URIs from the text documents, the URI component is described in subsection 4b.

A. Data description

In this project, the data for training and validation is stored in arff format file that contains examples from AI and Not-AI (Biology), and the contents were removed from noise, including stopwords, references and links. We converted the file into two groups of documents. One group has 2383 documents belonged to AI category and the other group has 1747 documents belonged to Biology category. Those two group documents are all in text format.

B. URI extraction

After splitting the arff documents into two groups of text documents, URI were obtained from long text strings based on the DBpedia knowledge base. DBpedia (from "DB" for "database") is a project aiming to extract structured content from the information created in the Wikipedia project. This structured information is made available on the World Wide Web. DBpedia allows users to semantically query relationships and properties of Wikipedia resources, including links to other related datasets. In this assignment, DBpedia Spotlight[5] was used to annotate the content of the documents. This annotation process will be briefly described in section III. Then the extracted URIs were ranked by TF-IDF score.

URI is a string of characters used to identify a resource. Such identification enables interaction with representations of the resource over a network. In this project, the URI interacts the spotted entities from the documents with the DBpedia links. The syntax of generic URIs and absolute URI references was first defined in Request for Comments (RFC)[6], Fig. 1 shows the generic form of URI, it comprises scheme, an authority part, a path, a query, an optional fragment. It also presents an example of URI from a document categorized by Biology in this project. In Fig. 1, the 'http' is the component of scheme, 'depedia.org' refers to the component of authority part, 'resource' refers to the component of path, and 'Biositemap' refers to the component of query.

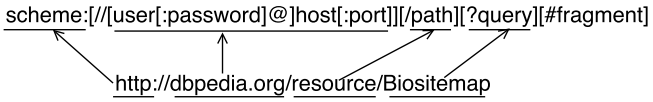


Fig. 1. URI syntax. The upper one is syntax of generic URIs. The lower one is an example of URI from a document categorized by Biology in this project.

III. Semantic Annotation using DBpedia Spotlight

In this section, the process of extracting URIs using DBpedia Spotlight API from documents is described. The semantic annotation process has four main steps:

- Spotting: Identification of entity names substrings of the original input that may be entity mentions.
- Candidate Selection: Selecting a set of entity names from step 1 along with the DBpedia resources that are candidate meanings for those entity names.
- Disambiguation: Deciding on the most likely candidate resource for each selected entity name.
- Filtering: Adjusting the annotations to task-specific requirements according to user-provided configuration.

The details of each processing step will be briefly described in subsection III-A, III-B, III-C, III-D, respectively.

A. Spotting Algorithm

In this project, text string of each document was firstly processed into a lexicon for spotting using LingPipe Exact Dictionary-Based Chunker [7]. The lexicon was a dictionary consisted by the spotted entities that interconnect multiple resources from the document.

B. Candidate Selection

After creating lexicon from documents, the spotted entities were mapped to candidate disambiguations (e.g. Washington as reference to a city, to a person or to a state). In this stage, the DBpedia Lexicalization dataset was used for determining candidate disambiguations for each spotted entity. The purpose of candidate selection is to narrow down the searching space thus reducing the time consumption. The candidate selection can be viewed as a pre-ranking selection procedure that pick the candidates with the highest priority probability for a spotted phrase.

C. Disambiguation

After selecting candidate resources for each entity name, the best matched resource was further matched by each candidate resource Term Frequency-Inverse Corpus Frequency (TF-ICF) score. Similarly, the TF weight is commonly used in IR to measure the local relevance of a term in a document. In this project, TF represents the relevance of a word for a given resource. ICF is introduced to better distinguish the candidate resources for a given entity name and is calculated as Equation 1, R_s represents the set of candidate resources for a entity name s and $n(w_j)$ is the total number of resources in R_s that are associated with the word w_j.

The disambiguation process can be viewed as a ranking problem where the objective is to rank the correct DBpedia resource at position 1 according to the cosine similarity score between their context vectors and the context surrounding the entity name.

$$ICF(w_j) = log|R_s| - logn(w_j) \quad (1)$$

D. Filtering

Finally, to adjust the annotations to task-specific requirements according to user-provided configuration, various configuration parameters (Topic Pertinence, Contextual Ambiguity, Disambiguation Confidence.) are set to make it easier to perform a specific task.

In this project, we used different disambiguation confidence value to test the classifiers performance. The confidence parameter ranges from 0 to 1. It takes into account factors such as the topical pertinence and the contextual ambiguity. If choose a high confidence threshold, incorrect annotations can be avoided as much as possible, but this is at the risk of losing some correct annotations.

In this project, we tested the accuracy and time cost of choosing different confidence parameters (0.2, 0.3, 0.4, 0.5, 0.7) in subsection 2.

IV. Evaluation

In this section, we reported our experiment results in AI and Biology dataset. The dataset is splited into train and validation set based on 10-fold cross-validation. We applies 8 ML classifiers to classify the documents based on the train set bag-of-word sparse vector and evaluates these classifiers performance in classification accuracy, time cost, precision, recall and F1 score. Besides, we also compare the classifiers performance in different feature extraction strategies. One is used ngrams as features, the other is used URIs as features. The general performance of different algorithms and two feature extraction strategies are reported in the following subsections.

All the experiment is implemented with python and sklearn. The code was ran in a 2.5 GHz Intel Core i7 with MacOS systems.

A. Features comparison

In this subsection, two features extraction methods are compared with each other in AI and Biology dataset. The classification accuracy, time cost, precision, recall and F1 score are used to evaluate the classifiers.

The accuracy, precision, recall, f1-score of different algorithms using ngram features in AI and Biology are shown in Table I, the results using URI features when chose confidence parameter equals to 0.4 are shown in Table II.

The threshold of result is selected based on Zero-R. In our case, since the number of documents in AI and Biology is 2383 and 1747, the threshold is equaled to 0.577. In Table I, the best result in this dataset using ngram features was obtained from Random Forest and Linear-SVC classifer, the accuracy is 98.7%, and precision, recall, F1-score are all 99%, whereas the worst result was obtained by Bernoulli Naive Bayes, the accuarcy is 93.3%, the precision is 94%, the recall and F1-score are both 94%. In Table II, the best result using URI features was obtained by the same classifiers, and the accuracy is 98.6%, precision, recall, F1-score are all 99%. Whereas the worst result was obtained by Nearest Centroid classifier, the accuracy is 95.9%, the precision, recall and F1-score are all 96%.

Besides, the training and testing time of different feature extraction methods are shown in Fig. 4. Fig. 4a shows the classification accuracy and time cost of different classifiers between AI class and Biology class using ngram features. Fig. 4b shows the accuracy and time cost of different classifiers in the same dataset but using URI features, the time cost of

TABLE I
RESULTS IN AI AND BIOLOGY DATASET USING UNIGRAM FEATURES

	Accuracy	Precision	Recall	F1
Ridge	0.987	0.99	0.99	0.99
Perceptron	0.982	0.98	0.98	0.98
passive-agressive	0.982	0.98	0.98	0.98
KNN	0.987	0.99	0.99	0.99
Random Forest	0.987	0.99	0.99	0.99
Linear-SVC(L1)	0.987	0.99	0.99	0.99
Linear-SVC(L2)	0.987	0.99	0.99	0.99
SGDC(L1)	0.972	0.97	0.97	0.97
SGDC(L2)	0.980	0.98	0.98	0.98
SGDC(Elastic-net)	0.978	0.98	0.98	0.98
nearest centroid	0.944	0.95	0.94	0.94
naive-bayes(M)	0.973	0.97	0.97	0.97
naive-bayes(B)	0.933	0.94	0.93	0.93

TABLE II
RESULTS IN AI AND BIOLOGY DATASET USING URI FEATURES (CONFIDENCE = 0.4)

	Accuracy	Precision	Recall	F1
Ridge	0.986	0.99	0.99	0.99
Perceptron	0.986	0.99	0.99	0.99
passive-agressive	0.980	0.98	0.98	0.98
KNN	0.980	0.98	0.98	0.98
Random Forest	0.986	0.99	0.99	0.99
Linear-SVC(L1)	0.986	0.99	0.99	0.99
Linear-SVC(L2)	0.986	0.99	0.99	0.99
SGDC(L1)	0.968	0.97	0.97	0.97
SGDC(L2)	0.983	0.98	0.98	0.98
SGDC(Elastic-net)	0.976	0.98	0.98	0.98
nearest centroid	0.959	0.96	0.96	0.96
naive-bayes(M)	0.980	0.98	0.98	0.98
naive-bayes(B)	0.973	0.97	0.97	0.97

training and testing is as reference of the maximum time cost of using ngram features. As shown in Fig. 4, all the time cost of training and testing using ngram are almost twice as much as using URI. For example, the Random Forest classifier using ngram features took 25.316s when training, and 0.210s when testing, whereas it only took 10.568s during training and 0.163s during testing when use URI features.

The performances of the same classifiers when using ngram and URI features were similarly in this dataset, but we can still say that the classifiers when using URI features (cofidence=0.4) performed a bit better than using ngrams features, since the overall accuracy was slightly higher than using ngrams as features. This is because the extracted URIs are disambiguous, compared with ngram features which are ambiguous and with no connection. Besides, the bag-of-word model is more sparse when using URI features (confidence=0.4), so the time cost was around half of the classifiers using ngrams. If we selected smaller value for confidence, as shown in Fig.2, the accuracy reached 99.4% (confidence=0.2, 0.3) using Random Forest classifier, and 99.4% (confidence=0.2), 99.2% (confidence=0.3) using KNN classifier. However, this is at a cost of much higher time consumption. The classifier performances of different confidence parameter value selection will be discussed in subsection IV-B.

B. Disambiguation Confidence Selection

Disambiguation confidence was used to take topical pertinence, contextual ambiguity into account. In this project, we chose the different value of confidence parameters (0.2, 0.3, 0.4, 0.5, 0.7). The classification accuracy, training time and testing time cost when select different confidence values under the categories AI and Biology dataset are reported in this subsection.

In Fig. 2, the y-axis represents to the three classifiers (KNN, Random Forest, Bernoulli Naive Bayes) accuracy and x-axis represents to the disambiguation confidence value we selected. In Fig. 3, the y-axis represents the training time cost (dashed line) and testing time cost (solid line). From Fig. 2 the accuracy reached the highest value which is 99.4% for KNN and Random Forest when the confidence equals 0.2, whereas Bernoulli Naive Bayes reached the highest performance 97.3% when confidence equals to 0.4. All the classifiers has the lowest accuracy when confidence equals to 0.7. Besides, the time costs for training and testing of all the classifiers reached the lowest point when confidence equals to 0.4. We can draw a conclusion that in this project, the overall classifiers have better performance when the confidence is selected by 0.4 since the accuracy is near the highest point and time cost is lowest.

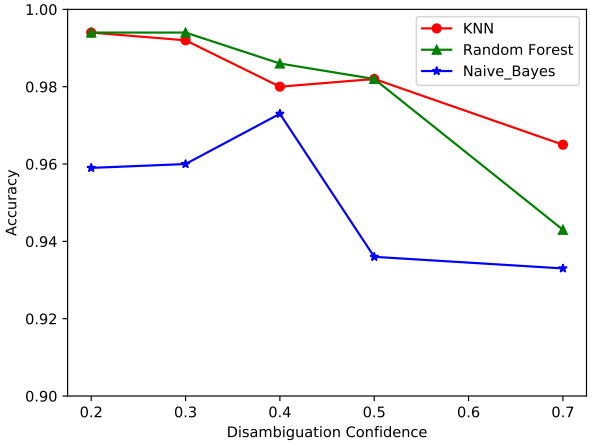


Fig. 2. Accuracy when select different Disambiguation Confidence.

V. CONCLUSION

In this report, we first introduced how to pre-process the arff format data for the text classification and how to split the train and validation dataset. After that, we report the 8 different classifiers performance using different feature extration methods and compare the performance results with different confidence value when using DBpedia Spotlight to select URIs. Different evaluation metrics are applied to the evaluate the classifiers. Through finishing all the experiments, we have a intuitively understanding for semantic annotation. In the future, we may want to try a more robust method for collective disambiguation, which is by harnessing context from knowledge bases and using a new form of coherence graph.

REFERENCES

[1] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

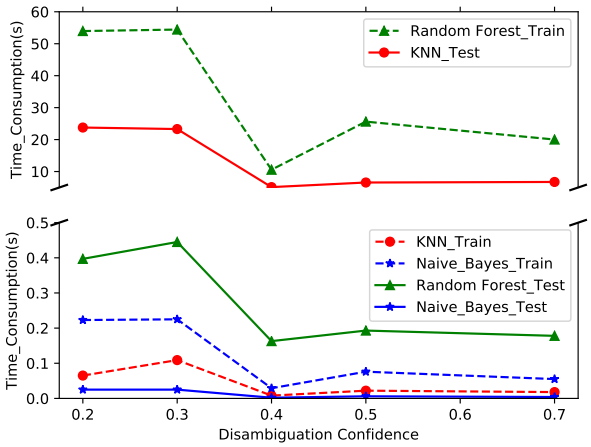
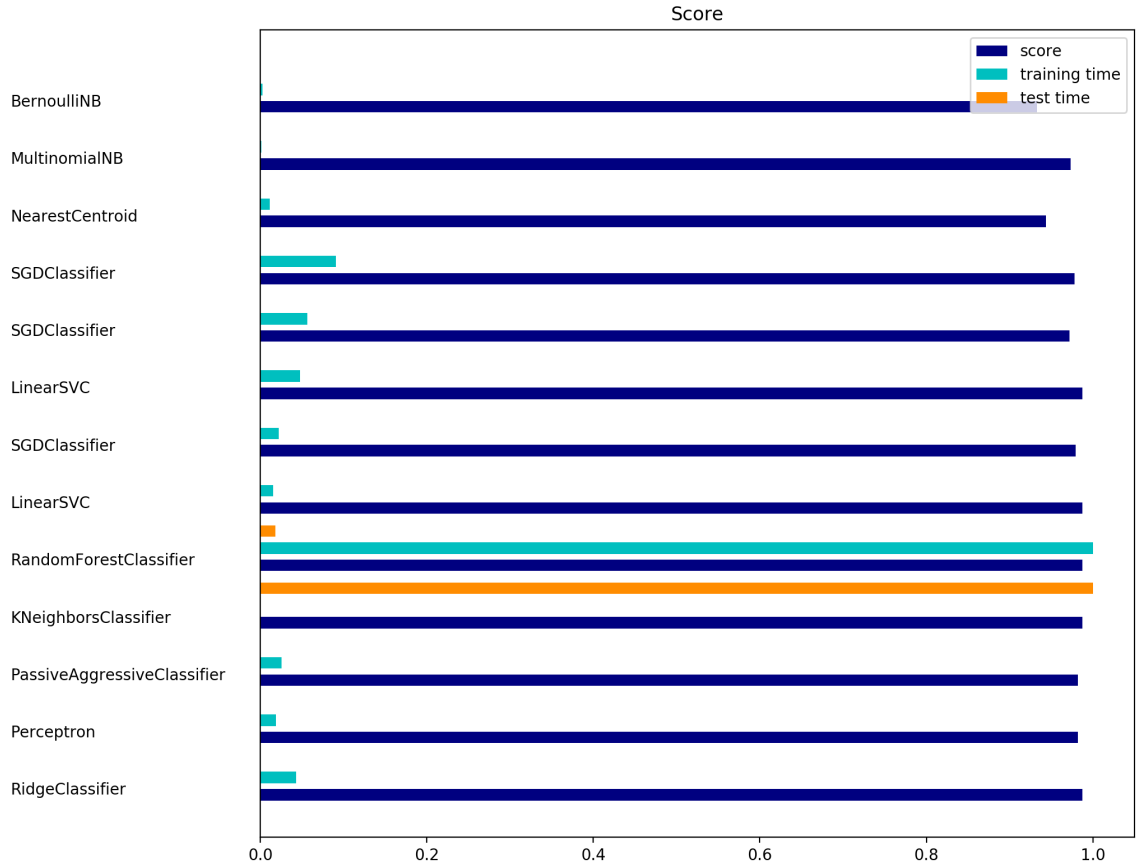
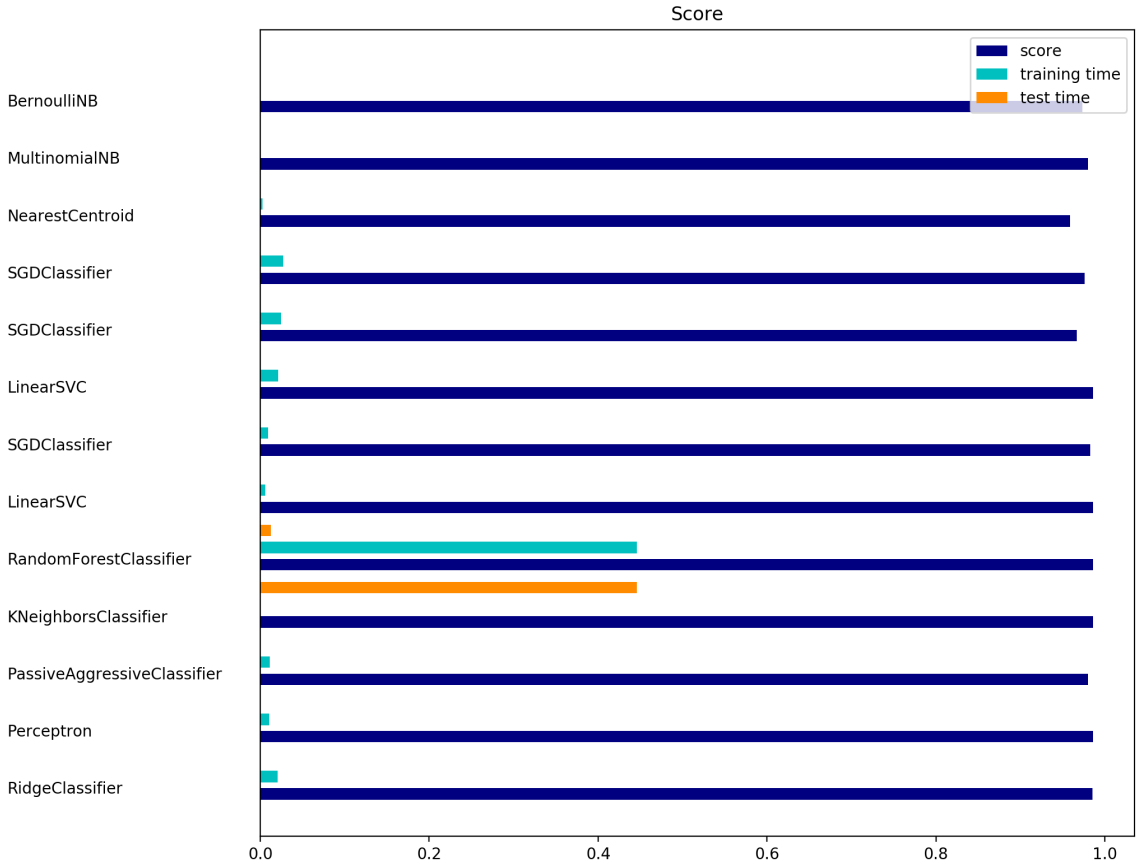


Fig. 3. Training and testing time cost when select different Disambiguation Confidence.

[2] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification." in *AAAI*, vol. 333, 2015, pp. 2267–2273.
[3] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.
[4] L. Masinter, T. Berners-Lee, and R. T. Fielding, "Uniform resource identifier (uri): Generic syntax," 2005.
[5] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "Dbpedia spotlight: shedding light on the web of documents," in *Proceedings of the 7th international conference on semantic systems*. ACM, 2011, pp. 1–8.
[6] T. Narten, "Guidelines for writing an iana considerations section in rfcs," 2008.
[7] Alias-i, "Lingpipe 4.0.0," http://alias-i.com/lingpipe, retrieved on 24.08.2010, 2008.



(a)



(b)

Fig. 4. Classification accuracy and training, testing time cost (a) AI and Biology classification results using Ngram features. (b) AI and Biology classification results using URI features.