CSE 575: SML
Improving Information Retrieval for Knowledge Extraction and Open Book Question Answering

Pratyay Banerjee
pbanerj6@asu.edu

Kuntal Pal
kkpal@asu.edu

Aditya Narayanan
anaray38@asu.edu

Kunal Bagewadi
kbagewad@asu.edu

Prasanth Sukhapalli
psukhapa@asu.edu

Bhavani Balasubramanyam
bbalasu6@asu.edu

Abstract

Information Retrieval (IR) and Knowledge Extraction(KE) is a core component for many NLP Tasks. Especially in Open Domain Question Answering (QA), we need to choose among very similar knowledge sentences. The knowledge sentences need to be relevant and not redundant. In this project, we improve on IR and KE for an application task of OpenBook QA. With improved KE, we also improve on OpenBook QA accuracy. We also introduce a new architecture BERT-CNN which uses sentence encodings for both tasks IR and QA, and show it performs better than existing models.

Mentor: Jun Wu

1 Introduction

Information Retrieval is a very core component for knowledge extraction and downstream tasks like open domain question answering. Current systems rely on Search Engines (Google/WikiSearch) and IR techniques using TF-IDF and BM-25 scores. These introduce noise in the retrieved documents which introduces errors in the downstream tasks. New systems utilizing Deep Learning techniques like CNNs,Bi-LSTMs and language models are upcoming but have room for improvement. Moreover new Language models learned through unsupervised learning techniques to derive contextual word embeddings, such as ELMO [2], Open-AI GPT [3], BERT [4] have recently been developed. These Language models can themselves be used for document ranking and can be used as features for more complex models.

The task of IR is hard for such applications like knowledge extraction for Open Domain Question Answering because the impact of irrelevant distracting knowledge to the QA systems. For example,

Question : Beak shape can influence a bird's ability ?
Relevant Fact : Beak is related to food.
Distracting Fact : Beak and mate has no relation.
Answer Options : "to mate with it's partner" , "**to chew up certain worms**"

Figure 1: Distracting Knowledge

Increasing Language Understanding, can lead to better IR techniques, and which in turn leads to better QA performance.

In this work, we show how improvement over IR techniques increases accuracy of downstream tasks, such as OpenBook QA [5].For deeper Natural Language Understanding, we evaluate BERT CLS Token Embedding and CNN based Sentence Embedding using BERT for IR and KE. We compare it against TF-IDF based features. We do a comparative empirical study of SML algorithms for the IR task. We chose the best IR model and evaluate over knowledge retrieval task of OpenBookQA. We evaluate a new BERT-QA model with CNN based Sentence Embedding. We try to understand the relevance of features present in CLS token using the observed feature importances and provide derived insights.

2 Dataset Description

In our experiments, we work on 2 Datasets, the first is the IR dataset over which we evaluate our features and choose the most appropriate model. The second is the QA dataset which needs external knowledge for systems to be able to perform QA.

- Microsoft (MS) AI Challenge Document Ranking Dataset :
  Size : $\sim$ 5 Million Total, Hidden Test Set
  Description : Each row contains 1 Query and 10 Documents to rank. Documents are of Science facts and Common knowledge.
  Label : Correct Document Number, i.e which document should be @ rank 1
  Metric : Mean Reciprocal Rank, Precision@K
  Baselines : TF-IDF

- Open Book Question Answering Dataset :
  Size : 4957 Train Set, 500 each in Validation and Test Sets. Knowledge 1350
  Description : Each row contains a Question and 4 answer options. Questions are from Science and Common Knowledge domain.
  Label : Correct Answer Option
  Metric : Accuracy, Precision@K
  Baselines : BERT Based MCQ model without any knowledge, with TF-IDF retrieved knowledge facts

3 Knowledge Extraction and Information Retrieval

In this section, we define the process we extracted features from BERT, explore and evaluate our IR models using BERT features. The dataset we choose for this evaluation is the Microsoft AI Challenge dataset.

3.1 Feature Selection and Generation

For feature generation we do the following steps:

1. **Baseline Feature** : TF-IDF scores for each Query-Document Pair.
2. For generating Sentence Embeddings, we finetune BERT Base using our entire corpus for few epochs. Both Language Model finetuning and NextSentencePrediction finetuing were done.
3. **CLS Token Sentence Embedding** : We extract only the CLS token after feeding our Query-Document pair to BERT. Feature Dimension : 768
4. **CNN Based Sentence Embedding** : We concat last 4 layers encodings of BERT Base including CLS token, and feed it through a CNN. Feature Dimension : 25344
5. A **PCA based reduced Sentence Embedding** from the CNN generated embeddings. Feature Dimension : 6000

3.2 Dataset Preparation

The MS AI Challenge Dataset contains 5 million entries, which pose a considerable challenge for being used for learning. We reduced the dataset to a considerable size of 75K Query Document pairs in the training set, and 15K each in the validation and test set. The MS AI Challenge Dataset was chosen for its closeness to the Openbook of the OpenBookQA dataset.

3.3 PCA Analysis

The features generated using BERT-CNN which is termed here as "ALL" data have data distributed across 25344 dimensions. Because of such huge amount of data it is not feasible to evaluate all algorithms to extract knowledge. As a solution, we applied Principal Component Analysis[44]. In our analysis we found that 6000 dimensions. Thus we reduced three-fourth of the total dimensions.

3.4 Model Description

3.4.1 Tree Family of Algorithms

We decided to do analysis on the following tree algorithms. The parameters we choose to find optimal model are Minimum Samples Split and Maximum Depth for each of the classifiers.

Decision Tree Classifier

This classifier creats a set of decision rules inferred from dataset to predict the target class. The range for Minimum Samples Split was from 5 to 20. The range for Maximum Depth was from 2 to 20. We have used Grid Search Cross Validation for hyperparameter tuning. We found that the model in which 5 samples were used to perform a split and tree of depth 6 was optimal.

ExtraTrees Classifier

This classifier is randomized version of Decision Tree Classifier wherein best split of the nodes is chosen amoungst all possible features. The range for Minimum Samples Split was from 5 to 20. The range for Maximum Depth was from 2 to 20. We have used Grid Search Cross Validation for hyperparameter tuning. We found that the model in which 5 samples were used to perform a split and tree of depth 11 was optimal.

Random Forest Classifier

This is an ensemble classifier which fits different decision tree classifiers on various samples of dataset. The range for Minimum Samples Split was from 5 to 30. The range for Maximum Depth was from 2 to 20. We have used Grid Search Cross Validation for hyperparameter tuning. We found that the model in which 10 samples were used to perform a split and tree of depth 7 was optimal. This model gave us the best accuracy in the Tree Family of Algorithms.

ExtraTrees Ensemble Classifier

This is an ensemble classifier which fits different extremely randomized decision tree classifiers on various samples of dataset.The range for Minimum Samples Split was from 5 to 50. The range for Maximum Depth was from 2 to 20.We have used Grid Search Cross Validation for hyperparameter tuning.We found that the model in which 40 samples were used to perform a split and tree of depth 9 was optimal.

3.4.2 Linear Models

Linear models are simply a linear combination of weights and features. The following are some of the linear models chosen for the classification task.

Logistic Regression

We have used Binary Logistic Regression from the package sklearn. Logistic Regression is a discriminative model where the decision function learns from the posterior probability. The

hyperparameters selected for tuning are C(Inverse of Regularization strength) and tol(tolerance value). For our problem, we use an L2 penalized Logistic function. Our classifier uses the Coordinate descent algorithm which is highly suited for binary classification. The range of C is chosen as [0.001, 0.01, 0.1] and tolerance is given as [0.001, 0.0001]. The Hyperparameter tuning is done using GridsearchCV. The best model has the tolerance value as 0.001, C as 0.001. It seems that the minimum C value provides a good balance with the training error and the testing error.

Perceptron

Perceptron is a single layer neutral network which consist of nodes equal to the number of features and each node is associated with a weight creating a linear equation. We have chosen the params alpha ([0.001, 0.01, 0.1, 1,0.0001]), tolerance ([0.001,0.0001]) and validation fraction ([0.05,0.1,0.2]). GridSearchCV performed hyperparameter tuning and gave us the results for the best model as validation fraction to be 0.05, alpha to be 0.001, tolerance to be 0.001. The learning rate is calibrated carefully and gave us better accuracy for 5% of validation dataset from training data.

Passive Aggressive Classifier

Passive Aggressive Classifier is suited to deal with continous large streams of data. Passive Agressive Classifier takes aggressive attempts in approximating the function when it finds misclassified points while training. The hyperparams that are to be tested for this Classifier are C [0.001, 0.01, 0.1, 1,10], tolerance [0.001,0.0001] and validation fraction [0.1,0.2,0.05]. The best model gave us tolerance as 0.0001, C as 0.001, validation fraction as 0.1 after doing GridSearchCV.

SGD Classifier

The SGD Classifier makes use of the Stochastic Gradient Descent Learning Method. It uses log as the loss function which provides a Logistic Regression Classifier with a different learning algorithm. The Hyperparameters chosen to be modified are alpha [0.0001, 0.001, 0.01], l1 ratio [0.15,0.30], epsilon [0.1,0.3,0.5], validation fraction [0.1,0.2], n iter no change [5,10]. And after GridSearchCV tuning, we found the best params as l1 ratio as 0.15, epsilon as 0.5, n iter no change as 10, alpha as 0.001, validation fraction as 0.2.

3.4.3 Suport Vector Machines

Linear

This is linear support vector machine based classifier which uses LibLinear optimizer. We used two main hyperparameters for Grid Search over the parameter space. The value of penalty parameter of error term(C) is varied from 0.001 to 10 and two distance metric used L1 and L2.

Linear Kernel Based

This is linear kernel based support vector machine which uses LibSvm optimizer. We used two main hyperparameters for Grid Search over the parameter space. The value of penalty parameter of error term(C) is varied from 0.001 to 10 and gamma(kernel coefficient) is varied from 0.001 to 1.

Polynomial Kernel Based

This is polynomial kernel based support vector machine which uses LibSvm optimizer. We used three main hyperparameters for Grid Search over the parameter space. The value of penalty parameter of error term(C) is varied from 0.001 to 10, gamma(kernel coefficient) is varied from 0.001 to 1 and the degree of polynomial is varied from 2 to 5.

Radial-Basis Kernel Based

This is Radial-Basis kernel based support vector machine which uses LibSvm optimizer. We used two main hyperparameters for Grid Search over the parameter space. The value of penalty parameter of error term(C) is varied from 0.001 to 10 and gamma(kernel coefficient) is varied from 0.001 to 1.

Sigmoid Kernel Based

This is Sigmoid kernel based support vector machine which uses LibSvm optimizer. We used two main hyperparameters for Grid Search over the parameter space. The value of penalty parameter of error term(C) is varied from 0.001 to 10 and gamma(kernel coefficient) is varied from 0.001 to 1.

3.4.4 Naive Bayes

Naive Bayes algorithms are a set of supervised learning algorithms that are based on Bayes' theorem with the assumption of conditional independence between every pair of features, given the value of the class variable. Naive Bayes classifiers can be extremely fast compared to more sophisticated algorithms.

Gaussian Naive Bayes

Gaussian Naive Bayes (GaussianNB) implements the Gaussian Naive Bayes algorithm for classification. It can perform updates to model parameters via the fit function. The function accepts two parameters, priors and var_smoothing. The values used for priors was none, and the values used for var_smoothing was 1e-09.

Multinomial Naive Bayes

Multinomial Naive Bayes (MultinomialNB) implements the naive Bayes algorithm for multinomially distributed data, and is one of the two classic naive Bayes variants used in text classification. This distribution is parametrized by alpha which is the smoothing parameter, fit_prior which tells whether the classifier needs to learn class prior probabilities or not, and class_prior which is the prior probabilities of the classes. Here, the values we have chosen are 1.0, none and true.

Bernoulli Naive Bayes

Bernoulli Naive Bayes (BernoulliNB) implements the naive Bayes training and classification algorithms for data that is distributed according to multivariate Bernoulli distributions. Here, each of the multiple features is assumed to be a binary-valued variable. Therefore, this class requires samples to be represented as binary-valued feature vectors. The parameters it accepts are alpha which is the smoothing parameter, binarize which is the threshold for mapping of sample features, fit_prior which tells whether the classifier needs to learn class prior probabilities or not, and class_prior which is the prior probabilities of the classes. Here, the values we have chosen are 1.0, 0.0, none and true.

3.4.5 Nearest Neighbors

The nearest neighbors model uses k-nearest neighbors (k-NN) classification. It is an instance based learning classifier which lets the nearest k neighbors of the given sample point vote for the output class of that sample. The common parameters to choose are k and the distance metric. We choose k by cross-validation and distance metric as Euclidean. Its evaluation is detailed in section 3.6.5.

The other classifier in this family is radius neighbors classifier. This classifier locates the neighbors within the the specified radius of a sample point and lets those neighbors vote for the output class of that sample point. Unlike k-NN, this classifier can lead to any number of neighbors of a sample point, inlcding zero neighbors. This classifer gives 0.5 accuracy which is not better than random guessing. So radius neighbor is not pursued for further evaluation.

3.5 Neural Networks

We use simple feed forward networks and Deep Learning models BERT and BERT CNN.

Feed Forward Networks We tried both 1 layer FFNs and 2 layer FFNs and searched for best appropriate, batch size, Hidden Layer size, learning rate and took the model with best Validation accuracy.

Deep Learnng Models We used vanilla BERT modified for IR task with a single layer of Feed Forward network for classification. The entire model was finetuned. The other model BERT-CNN uses the output encodings of vanilla BERT and passes the encodings to a CNN with max-pooling, and 4 features which are n-grams of lengths [2,3,4,5]. The output of the CNN is fed to a Feed Forward network. We finetune both the CNN and BERT layers.

3.6 Model Selection and Evaluation

In this section we show the results of the different algorithms over the IR task.

3.6.1 Tree Family of Algorithms

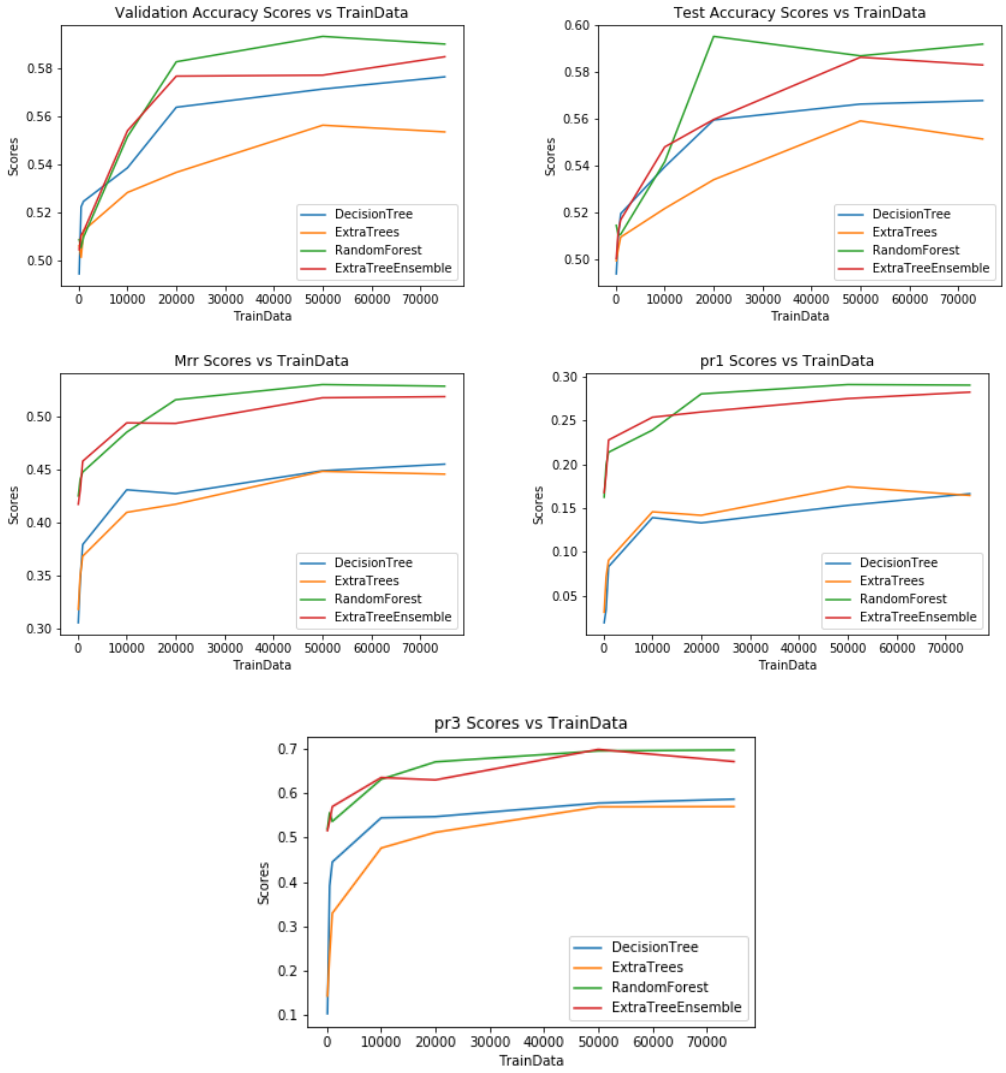


Figure 2: Plots for Tree Family Algorithms

Four models of Tree Family of Algorithms as mentioned in the previous sections are evaluated and compared. Tables 1,2,3

Tokens Data					
	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Decision-Tree	56.18	55.73	0.43	0.15	0.53
Extra-Tree	53.25	52.59	0.39	0.11	0.48
Random-Forest	58.65	57.00	0.47	0.22	0.60
Extra-Tree-Ensemble	57.53	57.06	0.48	0.24	0.61

Table 1: Comparative results for all models of Tree Family on Token data

CLS Data					
	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Decision-Tree	57.66	56.80	0.45	0.16	0.58
Extra-Tree	55.36	55.16	0.44	0.16	0.57
Random-Forest	59.03	59.21	0.52	0.29	0.69
Extra-Tree-Ensemble	58.50	58.32	0.51	0.28	0.67

Table 2: Comparative results for all models of Tree Family on CLS data

All Data					
	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Decision-Tree	55.84	55.68	0.46	0.18	0.58
Extra-Tree	54.39	53.60	0.40	0.12	0.45
Random-Forest	58.76	57.88	0.53	0.29	0.69
Extra-Tree-Ensemble	57.74	56.74	0.50	0.26	0.65

Table 3: Comparative results for all models of Tree Family on All data

3.6.2 Linear Family of Algorithms

The Previously mentioned linear classifiers are trained and evaluated with the best hyperparameters. The Classification Results based on the CLS token data obtained from BERT is given below:

CLS Data					
Models	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Logistic Regression	0.6031	0.5936	0.5208	0.2806	0.6786
Passive Aggressive	0.6063	0.5917	0.516	0.2786	0.6646
SGD Classifier	0.6069	0.5939	0.5208	0.2793	0.6833
Perceptron	0.5871	0.5729	0.4894	0.2513	0.6253

Table 4: Comparative Study of results obtained from Linear Classifiers using CLS token Embeddings

Tokens Data					
Models	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Logistic Regression	0.6472	0.6511	0.6008	0.376	0.777
Passive Aggressive	0.6393	0.6425	0.5971	0.3713	0.7813
SGD Classifier	0.6469	0.6427	0.6009	0.376	0.7733
Perceptron	0.6429	0.6507	0.5967	0.371	0.778

Table 5: Comparative Study of results obtained from Linear Classifiers using TF-IDF token Embeddings

All PCA Data					
Models	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Logistic Regression	0.6449	0.6469	0.5987	0.378	0.7786
Passive Aggressive	0.6302	0.6295	0.5707	0.34933	0.7313
SGD Classifier	0.6429	0.6453	0.5971	0.3766	0.7766
Perceptron	0.61	0.6111	0.5557	0.33	0.712

Table 6: Comparative Study of results obtained from Linear Classifiers using All PCA data

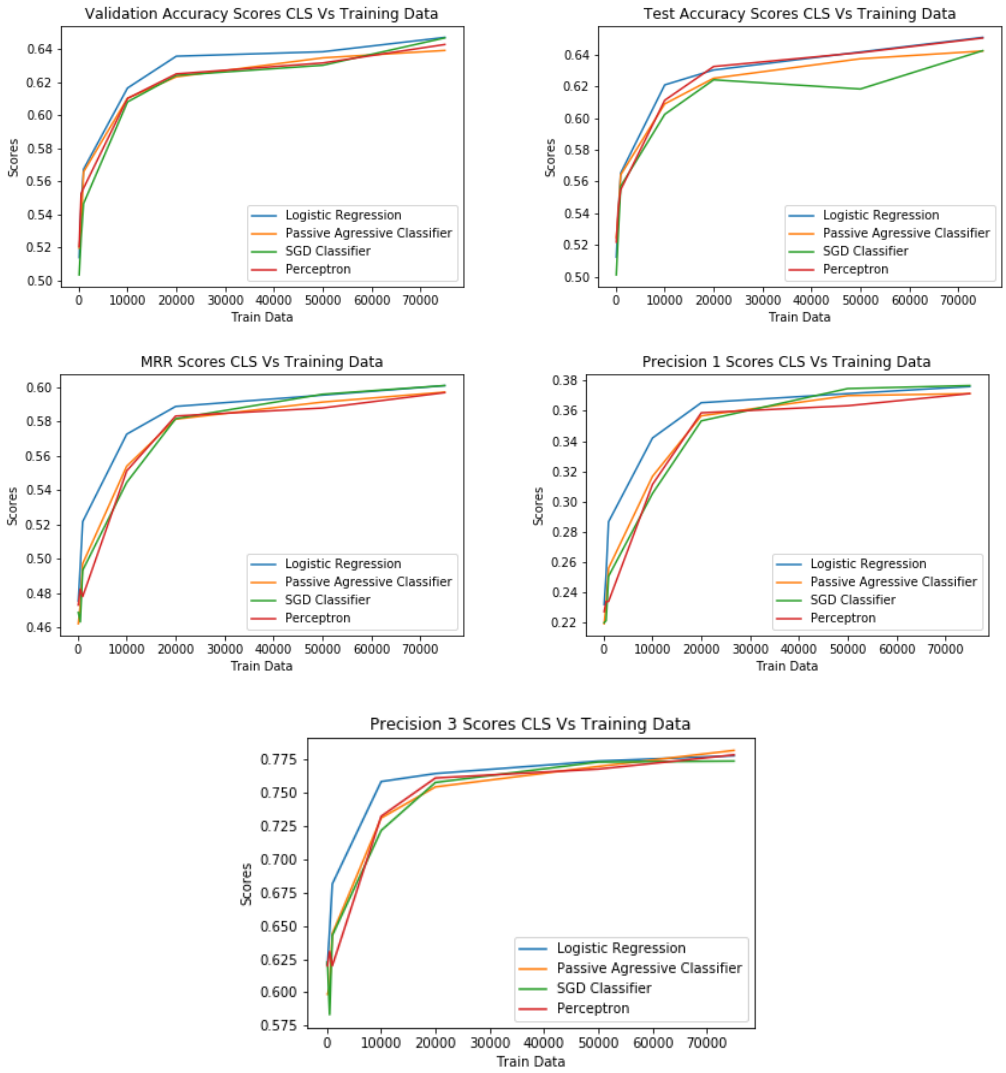


Figure 3: Plots for Linear Family of Classifiers

It is found that the Logistic Regression is the best model obtained from the handpicked classifiers in the linear family to be performing well on various categories of input data. Also, other classifiers were found to give nearly comparable results when CLS token embeddings were given as input. Logistic Regression uses the little complex Sigmoid function as decision function to determine the final output. SGD Classifier is similar to Logistic Regression but uses Stochastic Gradient Descent as Optimising algorithm. Perceptron on the other hand performed well on huge datasize but failed to compete with other classifiers on TF-IDF tokens.

3.6.3 Support Vector Machines

Four kernel based models and linear model of Support Vector Machines[45] as mentioned in the previous sections are evaluated and compared on the three types of data we have. There results can be found from Tables 7,8,9. From the results and the graphical representations is is clear that SVM Linear performs far better than all other kernel-based approaches like polynomial, gaussian and RBF. One reason being, the data is generated using BERT language model's embeddings which retains some structural cues of sentences. So on applying kernel-based methods the data gets transformed in the kernel space which confuses the kernel-based models leading to inferior results. Another reason is all the kernel-based SVC uses LibSvm optimizer which is very slow to converge given

large amount of data. So for all the kernel based aapproaces we kept till 1000 iterations which is not enough to produce good results. With the reduced 6000 features for "ALL" data and 1000 iterations it took a day to produce the results with Libsvm whereas linear SVC with LibLinear optimizer takes 2 hours. This shows the importance of using the optimizer iterations.

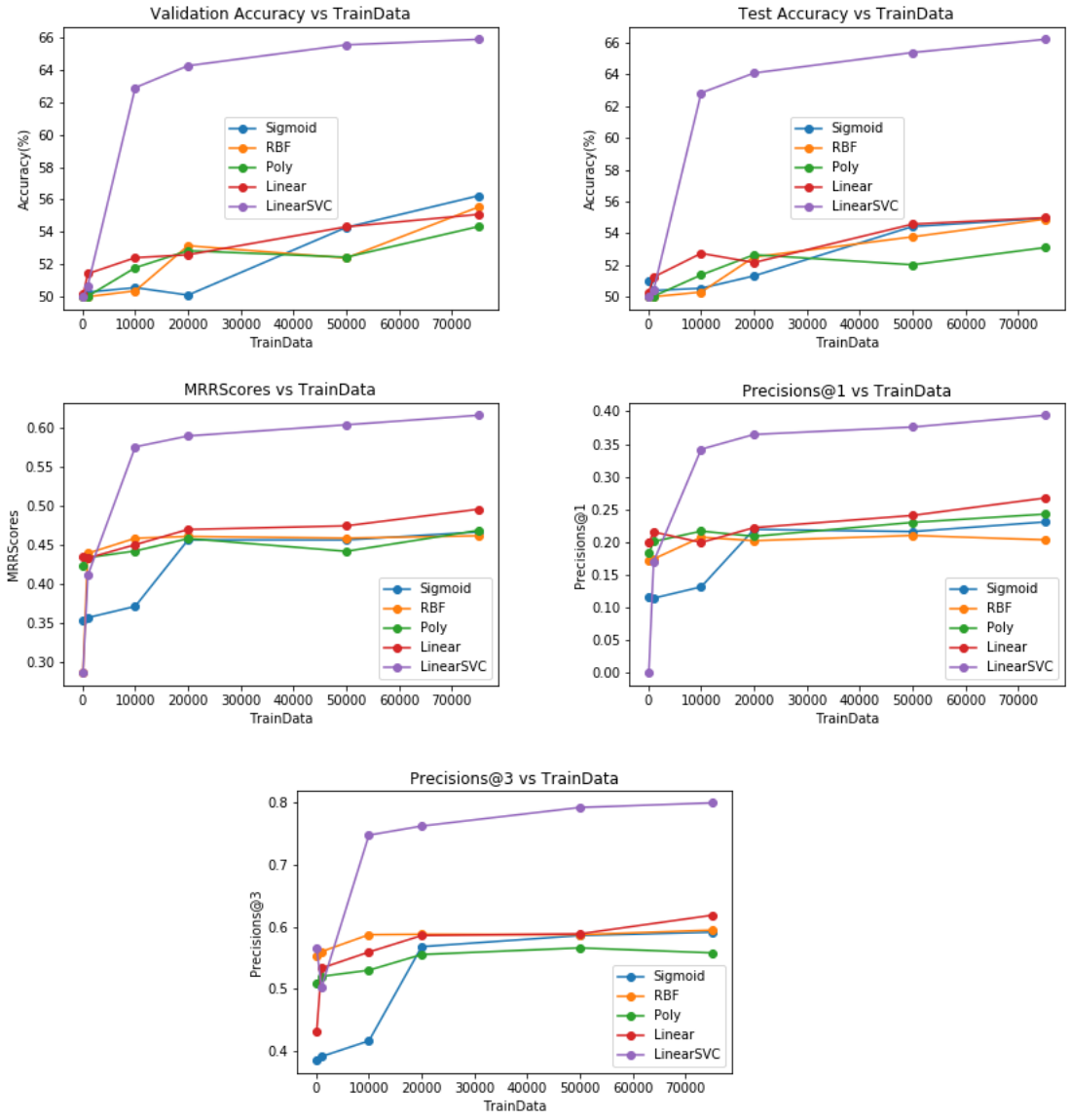


Figure 4: Plots for SVM Family Algorithms

Tokens Data					
Models	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Linear-Kernel	55.74	54.15	0.47	0.23	0.60
RBF-Kernel	55.56	54.08	0.47	0.23	0.61
Sigmoid-Kernel	55.61	54.71	0.47	0.20	0.59
Polynomial-Kernel	51.93	50.89	0.45	0.20	0.57
SVCLinear	60.00	59.05	0.51	0.27	0.66

Table 7: Comparative results for all models of SVM Family on Token data

CLS Data					
Models	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Linear-Kernel	50.11	50.27	0.42	0.18	0.53
RBF-Kernel	55.33	54.87	0.28	0.17	0.40
Sigmoid-Kernel	52.98	52.96	0.34	0.18	0.40
Polynomial-Kernel	53.94	50.00	0.50	0.28	0.64
SVCLinear	64.43	64.71	0.59	0.36	0.78

Table 8: Comparative results for all models of SVM Family on CLS data

All Data					
Models	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
Linear-Kernel	52.59	52.15	0.43	0.19	0.53
RBF-Kernel	52.39	52.50	0.45	0.20	0.58
Sigmoid-Kernel	50.06	50.53	0.45	0.21	0.58
Polynomial-Kernel	51.79	52.01	0.43	0.20	0.50
SVCLinear	65.89	66.20	0.61	0.39	0.80

Table 9: Comparative results for all models of SVM Family on All data

3.6.4 Naive Bayes Family of Algorithms

The three models of Naive Bayes family of algorithms as mentioned in the previous section are evaluated and compared. Below are the tables that summarize the same: 10,11,12

Tokens Data					
	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
GaussianNB	58.15	57.27	0.45	0.12	0.66
MultinomialNB	58.17	58.07	0.51	0.27	0.67
BernoulliNB	58.81	58.79	0.52	0.29	0.69

Table 10: Comparative results for all models of Naive Bayes family on Token data

CLS Data					
	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
GaussianNB	50.00	50.00	0.28	0.0	0.0
MultinomialNB	58.09	57.85	0.53	0.29	0.70
BernoulliNB	50.00	50.00	0.43	0.18	0.55

Table 11: Comparative results for all models of Naive Bayes family on CLS data

All Data					
	Val Accuracy	Test Accuracy	MRR	Precision@1	Precision@3
GaussianNB	50.00	50.00	0.28	0.0	0.0
MultinomialNB	54.79	54.29	0.32	0.0	0.44
BernoulliNB	50.00	50.00	0.28	0.0	0.0

Table 12: Comparative results for all models of Naive Bayes family on All data

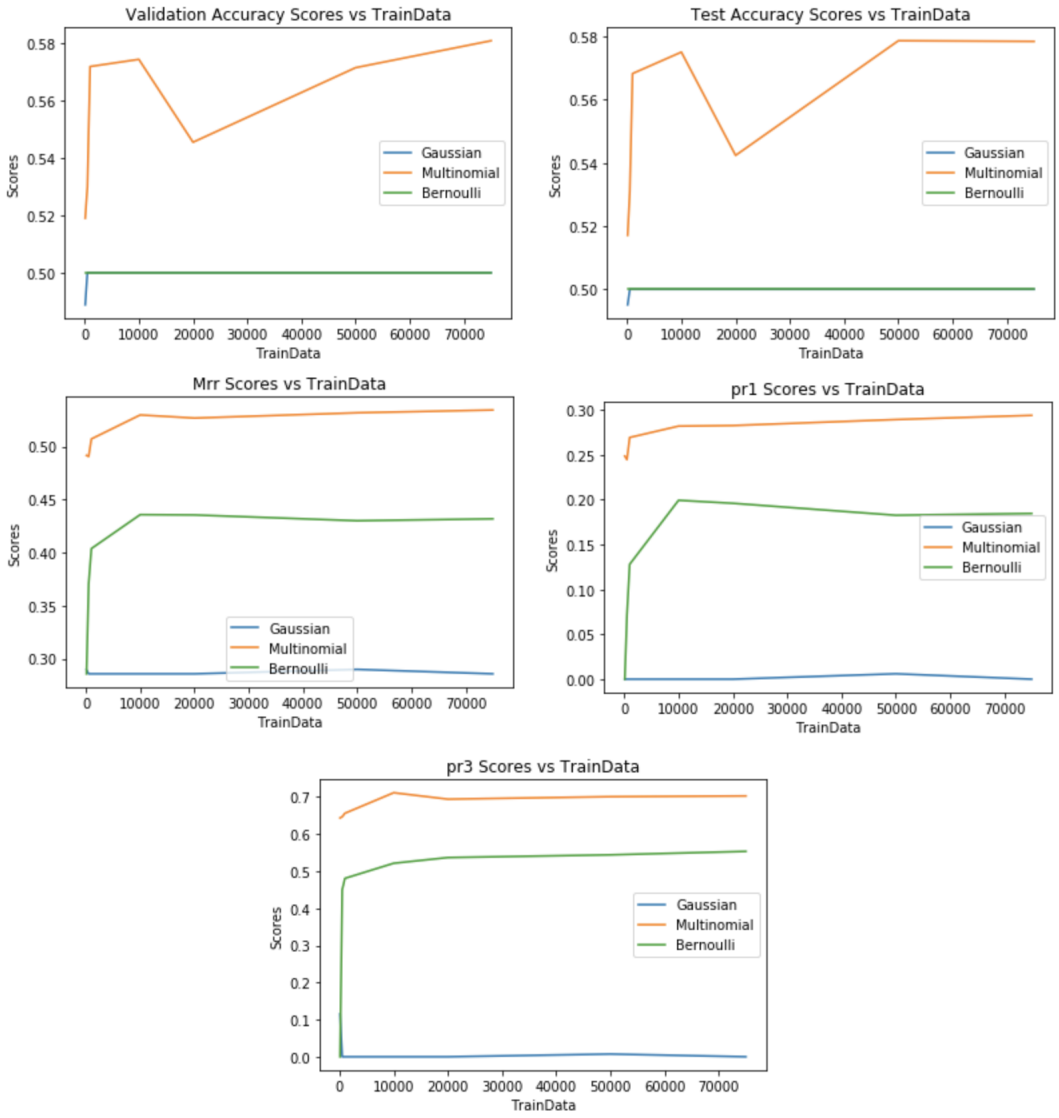


Figure 5: Plots for Naive Bayes Family Algorithms

3.6.5 Nearest Neighbors Algorithm

The k-NN algorithm was implemented using sklearn's KNeighborsClassifier. We used cross-validation on training data to select an opitmum value of k. We had 3 training datasets based on features extracted: TF-IDF tokens, CLS tokens using BERT and all tokens. With TF-IDF tokens, 3-fold cross-validation yielded k=949 as optimum value as seen in figure 6. The validation accuracy with using this dataset is 60.29%. With dataset using CLS tokens, 3-fold cross-validation yielded k=99 as optimum value as seen in figure 6. The validation accuracy with using this dataset is 65.19%, which is better than TF-IDF token dataset. With dataset using all tokens, 3-fold cross-validation yielded k=99 as optimum value. The validation accuracy with using this dataset is 54.56%.

So it is inferred that CLS tokens are best feature to be used with this algorithm, with k=99. The result metric with this configuration is given in table 13. The evaluation of this model with different training data size is shown in figure 7. It shows that more training data leads to better results.

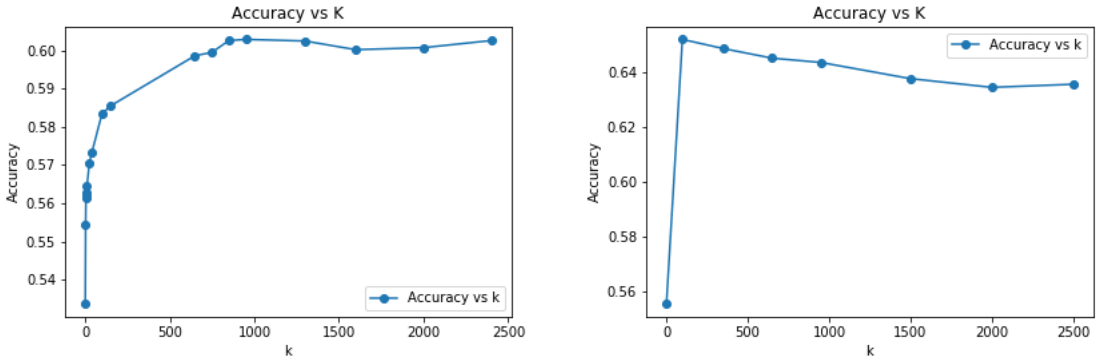


Figure 6: Cross-validation of k-NN algorithm for selecting k. Left: TF-IDF token dataset(best k=949). Right: CLS token dataset(best k=99)

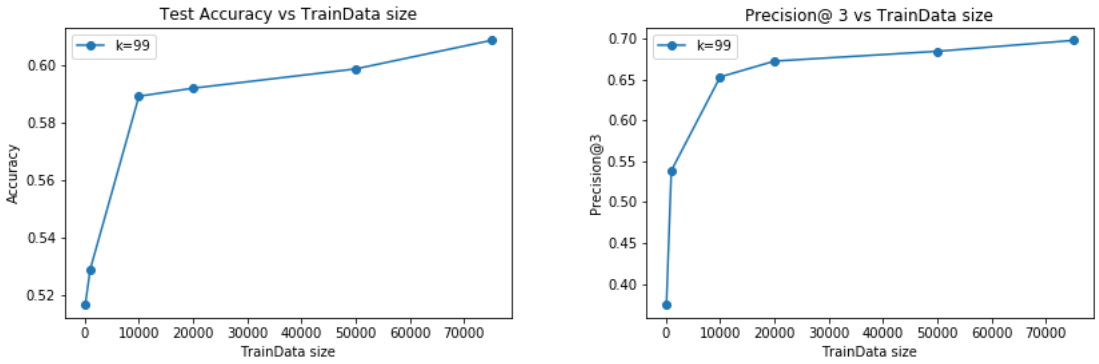


Figure 7: k-NN evaluation result with CLS dataset with k=99. Left: Test accuracy over various train data size. Right: Precision@3 score over various train data size)

Dataset	Val Accuracy(%)	Test Acccuracy(%)	MRR	Precision@1	Precision@3
CLS	61.97	60.87	0.547	0.3	0.698
TF-IDF	57.39	56.71	0.511	0.273	0.646
All	54.95	53.94	0.505	0.252	0.634

Table 13: k-NN algorithm test scores with CLS, TF-IDF(tokens), All dataset. Precision@3 score is best with CLS dataset using k=99.

3.7 Neural Networks

In this section we compare the performance of Neural Network models on the IR task.

3.8 Comparison of the families

The following section shows the comparison of the algorithm families and features.

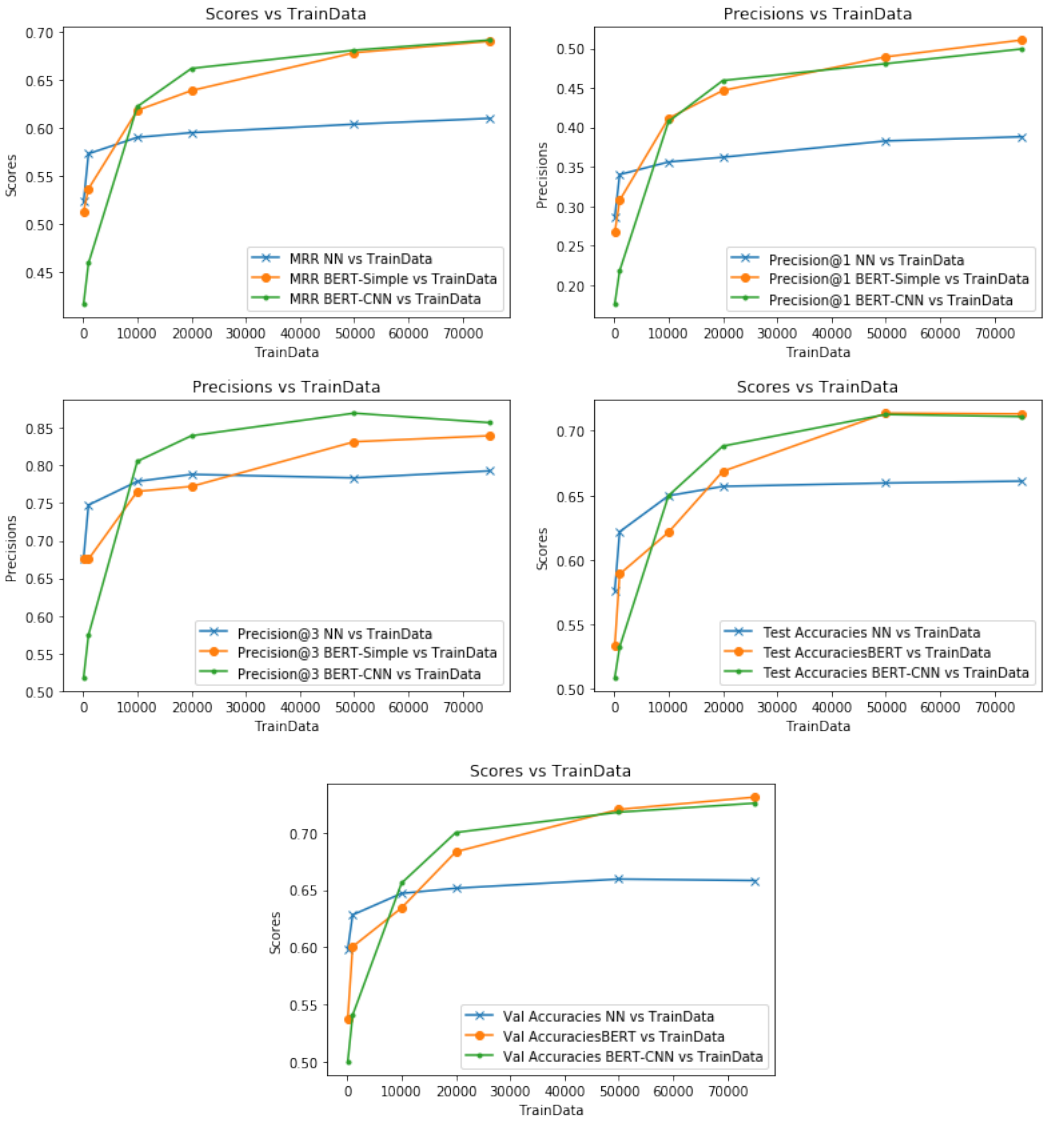
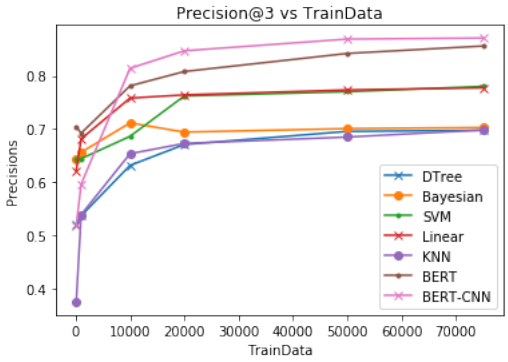


Figure 8: Plots for different Neural Network Models



P3	TFIDF	CLS	ALL	PCA
SVM	66.86	78	NA	80
NN	**67.7**	80.1	80.4	**80.2**
BERT	NA	**83.8**	**84.8**	NA
DT	61	69	69	65
KNN	64.67	69.8	NA	63.4
Bayes	67.53	70.26	44.533	64.53
Linear	67.86	77.73	68.6	77.86

Figure 9: Learning Curve and Comparison of features

		CLS-Tokens	Validation 2		
	Best Model	*Accuracy*	*Mrr*	*P1*	*P3*
DT-Family	Random Forest	59.03	52.9	29.1	69.7
KNN	k=99	60.87	54.7	30	69.8
Naive Bayes	MultinomialNB	58.09	53.4	29.4	70.3
Linear Family	Logistic Regression	65.11	60.1	37.6	77.7
SVM	LinearSVC	65.89	61.6	39.4	80
NN	2 Layer NN	66.04	60.5	39.6	80.1
BERT	Base ,Batch:64, lr :1e-5	70.4	68.9	49.8	85.6
BERT-CNN	Base ,Batch:64, lr :5e-6	**71.2**	**69.6**	**50.3**	**86.1**

Figure 10: Comparison across families

Model - Data	Precision@3
BERT- Large 75K	0.873
BERT - CNN - Large 75K	0.881
BERT - Large 2M	0.904
BERT - CNN - Large 2M	***0.907***

Figure 11: Throwing more data to BERT Base and BERT Large

4 OpenBook QA

For the QA task, we evaluate 2 Models, each with retrieved knowledge. The Baseline model is the No Knowledge model.

1. BERT - Large
2. BERT - Large with Sentence Embeddings using CNN

We also evaluate the new IR model over OpenBook of the dataset. As Baseline we compare with 2 models, a TF-IDF model and a vanilla BERT Pretrained over STS-B dataset (A Textual Similarity Dataset).

4.1 Model Description

In this section we describe the QA model. We use a vanilla BERT Large model, where each question and answer option are combined to create a Hypothesis. Using this hypothesis we score the probability of this hypothesis being true given a knowledge passage. The model is similar to the models described in [1].

BERT-CNN QA Model : The new model which we introduce includes a CNN over the BERT model, which contains a MaxPool layer, and 4 features which are n-grams of lengths [2,3,4,5]. We evaluate both models with our augmented knowledge.

4.2 Evaluation

We evaluate the new IR models in knowledge extraction of the OpenBook task.

Model - Trained On Dataset	Precision@1	Precision@3
TF-IDF	228/500	324/500
BERT - STS-B	288/500	368/500
BERT - Large - MS AI	240/500	304/500
BERT - Large - OBQA	258/500	358/500
BERT CNN - MS AI	**314/500**	**389/500**
BERT CNN - OBQA	264/500	353/500

Figure 12: IR Model Comparisons over OpenBook

We also evaluate the knowledge extracted from OpenBook using IR models on the QA task.

Model (BERT Large)	Val Accuracy	Test Accuracy
No Knowledge - Leaderboard	60.2	60.4
Knowledge STS	66.8	66.2
BERT-CNN + Knowledge + MS AI	**67.4**	**67.2**
BERT-CNN + *Oracle* Knowledge	*74.2*	*74.4*

Figure 13: QA Model Comparisons

5 Analysis and Insights

The experiments helped us to gain various essential insights.

- BERT, a pretrained language model captures a lot of semantics and language structure, which is shown by the performance of other models using BERT features.
- Analyzing feature importances of CLS token encodings, lead to identification of only few dimensions being used by most models to determine effectively the Relevance indicating BERT captures semantics across only a few dimensions.
- Using BERT encodings and a CNN as a feature increases semantic information and leads to better task performance, both IR and QA.
- Sentence embeddings and PCA of embeddings did not reduce classifier performance by much and enabled further explorations and experiments .
- Most SML algorithms are not scalable for a huge dataset, whereas BERT was easily able to scale to 2M Query-Document pairs. SVM took 2 days to train with 75K pairs, BERT took 20 mins and with better performance.
- IR of OpenBook QA showed the Transfer Learning capability of BERT-CNN model trained over MS AI data.
- The improvement of QA task is only 7%, shows how hard the overall task is and more precise external knowledge is needed.
- KNN with 99 neighbours approached an acceptable performance, such a neighbourhood indicates a minimum neighbourhood for distinguishing Relevant and Irrelevant documents
- On the other hand, the SML models fail in comparison to fine-tuned BERT as expected, as features for the models come from an unsupervised BERT model, and a supervised BERT model will update its parameters to the required task and Domain much better.

6 Conclusion and Future Work

We showed through BERT features that semantic information is needed to improve information retrieval and knowledge extraction. More work can be done to improve it as the human performance is far from our achieved results. Analysis of question answering task, robustness of our models and testing with adversarial data has not yet been done. That can be another area of research. We explored the explainability of BERT features using feature importances and got some insights. But in future we would like to explain better by establishing a relationship between CLS token features and concrete natural language semantics and sense.

References

[1] Pan, X., Sun, K., Yu, D., Ji, H., Yu, D. (2019). Improving Question Answering with External Knowledge. arXiv preprint arXiv:1902.00993.

[2] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

[3] Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training. URL : `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper.pdf`

[4] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[5] Mihaylov, T., Clark, P., Khot, T., & Sabharwal, A. (2018). Can a suit of armor conduct electricity? a new dataset for open book question answering. arXiv preprint arXiv:1809.02789.

[6] Roberts, I. (2002). Information Retrieval for question answering (Doctoral dissertation, MSc Dissertation, Department of Computer Science, The University of Sheffield, UK. Available, Februray 2003, from http://www. dcs. shef. ac. uk/teaching/eproj/msc2002/abs/m1ir. htm).

[7] Gaizauskas, R., Hepple, M.,& Greenwood, M. (2004, December). Information retrieval for question answering a SIGIR 2004 workshop. In ACM SIGIR Forum (Vol. 38, No. 2, pp. 41-44). ACM.

[8] Liu, T. Y. (2009). Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3), 225-331.

[9] Page, L., Brin, S., Motwani, R.,& Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Stanford InfoLab.

[10] Singhal, A. (2001). Modern information retrieval: A brief overview. IEEE Data Eng. Bull., 24(4), 35-43. Xu, J.,& Li, H. (2007, July). Adarank: a boosting algorithm for information retrieval. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 391-398). ACM.

[11] Hirschman, L.,& Gaizauskas, R. (2001). Natural language question answering: the view from here. natural language engineering, 7(4), 275-300.

[12] Abney, S., Collins, M.,& Singhal, A. (2000, April). Answer extraction. In Proceedings of the sixth conference on Applied natural language processing (pp. 296-301). Association for Computational Linguistics.

[13] Wang, J. (2009, April). Mean-variance analysis: A new document ranking theory in information retrieval. In European Conference on Information Retrieval (pp. 4-16). Springer, Berlin, Heidelberg

[14] Dimitriadis, D.,& Tsoumakas, G. (2019). Word Embeddings and External Resources for Answer Processing in Biomedical Factoid Question Answering. Journal of Biomedical Informatics, 103118.

[15] Carbonell, J. G.,& Goldstein, J. (1998). The Use of MMR and Diversity-Based Reranking for Reodering Documents and Producing Summaries.

[16] Lee, K. S., Park, Y. C.,& Choi, K. S. (2001). Re-ranking model based on document clusters. Information processing& management, 37(1), 1-14.

[17] Yang, L., Ji, D., Zhou, G., Nie, Y.,& Xiao, G. (2006, November). Document re-ranking using cluster validation and label propagation. In Proceedings of the 15th ACM international conference on Information and knowledge management (pp. 690-697). ACM.

[18] Zhou, L. (2009, August). Using term relation in context sensitive information retrieval. In 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery (Vol. 1, pp. 354-358). IEEE.

[19] Li, Z., Xu, G., Liang, X., Li, F., Wang, L.,& Zhang, D. (2019). Exploring the Importance of Entities in Semantic Ranking. Information, 10(2), 39.

[20] Benedetti, F., Beneventano, D., Bergamaschi, S.,& Simonini, G. (2019). Computing inter-document similarity with Context Semantic Analysis. Information Systems, 80, 136-147.

[21] Van Lierde, H.,& Chow, T. W. (2019). Query-oriented text summarization based on hypergraph transversals. arXiv preprint arXiv:1902.00672.

[22] Verma, V. K., Yadav, A.,& Jain, T. (2019). Key Feature Extraction and Machine Learning-Based Automatic Text Summarization. In Emerging Technologies in Data Mining and Information Security (pp. 871-877). Springer, Singapore.

[23] Dammak, F., Kammoun, H.,& Hamadou, A. B. (2017, November). Improving pairwise learning to rank algorithms for document retrieval. In 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1-8). IEEE.

[24] Weixin, T.,& Fuxi, Z. (2009, May). Learning to rank using semantic features in document retrieval. In 2009 WRI Global Congress on Intelligent Systems (Vol. 3, pp. 500-504). IEEE.

[25] Patel, S., Khanna, K.,& Sharma, V. (2016, April). Documents ranking using new learning approach. In 2016 International Conference on Computing, Communication and Automation (ICCCA) (pp. 65-70). IEEE.

[26] Jung, C., Jiao, L. C.,& Shen, Y. (2011, September). Ensemble ranking SVM for learning to rank. In 2011 IEEE International Workshop on Machine Learning for Signal Processing (pp. 1-6). IEEE.

[27] Gopal, S.,& Raghav, S. (2017, August). Automatic document retrieval using SVM machine learning. In 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon) (pp. 896-901). IEEE.

[28] Taneja, H.,& Gupta, R. (2010, June). Web information retrieval using query independent page rank algorithm. In 2010 International Conference on Advances in Computer Engineering (pp. 178-182). IEEE.

[29] Yang, X., Lo, D., Xia, X., Bao, L.,& Sun, J. (2016, October). Combining word embedding with information retrieval to recommend similar bug reports. In 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE) (pp. 127-137). IEEE.

[30] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K.,& Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

[31] Perone, C. S., Silveira, R.,& Paula, T. S. (2018). Evaluation of sentence embeddings in downstream and linguistic probing tasks. arXiv preprint arXiv:1806.06259.

[32] Peters, M. E., Neumann, M., Zettlemoyer, L.,& Yih, W. T. (2018). Dissecting contextual word embeddings: Architecture and representation. arXiv preprint arXiv:1808.08949.

[33] Salant, S.,& Berant, J. (2017). Contextualized word representations for reading comprehension. arXiv preprint arXiv:1712.03609.

[34] Levy, O.,& Goldberg, Y. (2014). Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (Vol. 2, pp. 302-308).

[35] Yu, L., Hermann, K. M., Blunsom, P.,& Pulman, S. (2014). Deep learning for answer sentence selection. arXiv preprint arXiv:1412.1632.

[36] Habibi, M., Weber, L., Neves, M., Wiegandt, D. L.,& Leser, U. (2017). Deep learning with word embeddings improves biomedical named entity recognition. Bioinformatics, 33(14), i37-i48.

[37] Mike Mintz et al.: Distant supervision for relation extraction without labeled data, ACL 2009

[38] Mintz, M., Bills, S., Snow, R.,& Jurafsky, D. (2009, August). Distant supervision for relation extraction without labeled data. In Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2 (pp. 1003-1011). Association for Computational Linguistics.

[39] Improving Web Search Ranking by Incorporating User Behavior Information Agichtein, E., Brill, E.,& Dumais, S. (2006, August). Improving web search ranking by incorporating user behavior information. In Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 19-26). ACM.

[40] Learning to rank for information retrieval Liu, T. Y. (2009). Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3), 225-331.

[41] Feature ExtractionA Pattern for Information Retrieval Manolescu, D. A. (1998). Feature extractionA pattern for information retrieval. Proceedings of the 5th Pattern Languages of Programming, Monticello, Illinois.

[42] Contextual Word Representations: A Contextual Introduction, 2019 https://arxiv.org/abs/1902.06006

[43] Learning to rank: from pairwise approach to listwise approach Liu, T. Y. (2009). Learning to rank for information retrieval. Foundations and Trends in Information Retrieval, 3(3), 225-331.

[44] Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." science 313.5786 (2006): 504-507.

[45] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.