3-cfu Project Work: Question classification on the Text REtrieval Conference (TREC) Question Classification dataset.

Nicola Carrassi
Master's Degree in Artificial Intelligence, University of Bologna
nicola.carrassi@studio.unibo.it

Abstract

In this report we will assess the use of various techniques to tackle the problem of Question Classification on the TREC dataset. The task consists of classifying the correct category of a given question. The correct question belongs to 1 of 6 different categories. Given the fact that each question belongs to one and only one of the different categories, this is a Multi-Class classification task. Three approaches have been tested: Machine Learning techniques applied on the latent space, Transformer-based techniques with multiple architectures and a Graph Neural Network (GNN). The results suggest that the Transfomer-based approach outperforms both the ML techniques and the GNN on this task, achieving a better performance in terms of accuracy, obtaining in the best configuration (BERT-base) an accuracy score of 0.978

1 Introduction

The task of question classification involves assigning a predefined category to a given question. This problem has been studied extensively in natural language processing (NLP). This study will takcle this problem on the TREC dataset. The TREC dataset consists of over 5,000 labeled questions from various domains, such as entertainment, science, and geography, among others.
Accurately classifying questions into their appropriate categories is crucial for many applications, such as information retrieval, question answering systems, and dialogue systems. Therefore, the development of effective question classification methods is of great interest to the NLP community.
In recent years, there has been significant progress in developing deep learning models for question classification, such attention-based models. However, there is still room for improvement in terms of accuracy and generalization.
In this paper, we present a comprehensive study of question classification on the TREC dataset, comparing the performance of various deep learning models. We also propose a novel approach that leverages graph neural networks (GNNs) to model the relationships between words in a question. Our experimental results demonstrate that our BERT-base approach outperforms existing state-of-the-art, achieving a slight improvement in terms of accuracy of the test set.
Multi-class classification is a type of classification problem where the goal is to classify instances into one of three or more classes. In other words, it involves predicting a target variable that can take one of several possible discrete values. Each instance is assigned to a single class and the model is trained on labeled data that has already been assigned to different classes. In multi-class classification, the model typically predicts a probability distribution over the classes for each instance, indicating the likelihood of the instance belonging to each class. The class with the highest probability is then selected as the predicted class for that instance. Traditionally this kind of problems has been dealt with using approaches based on Machine Learning such as SVM (Zhang and Lee, 2003). These kind of techniques represent a good baseline with respect to more complex architectures such as Transformer-based approaches (Vaswani et al., 2017).
In this work we decided to test and compare the results obtained with machine learning methods applied on the latent space (obtained performing the Latent Semantic Analysis), 2 different Transformer-based architectures and finally a Convolutional Graph Neural Network.
The ML-based architectures are a simpler alternative to Transformer-based approaches and can be a solution in settings with less computational power. This result can be considered a baseline result which could be improved with more complex techniques, such as GNN (Morris et al., 2018) and Transformers. As regards transformers, two differ-

ent architectures have been tested: BERT (Devlin et al., 2018) and DistilBERT (Sanh et al., 2019). We evaluated all the models on accuracy, since this is the metric which is used to benchmark this particular dataset.

2 Background

Latent Semantic Analysis (LSA) is a natural language processing technique that represents text data as a lower-dimensional representation in a vector space, known as the Latent Semantic Space. This representation captures the meaning of the text by preserving the relationships between words and documents. LSA uses singular value decomposition to reduce the dimensionality of the document-term matrix and to identify latent semantic relationship between documents and terms.

Performing LSA for question classification as done by (Datla et al., 2016) may be useful in order to caputre the semantic relationship between words and documents, overcoming the limitations of traditional rule-based approaches to question classification, which may not be able to handle the complexity of natural language questions.

Simply performing Latent Semantic Analysis (LSA) on a set of documents does not provide a way to classify those documents on its own. Instead, we need to use additional techniques to extract meaningful information from the latent semantic space created by LSA and use that information to classify the documents. Combining LSA with other classification techniques, we can create a more powerful and flexible system for organizing and analyzing large collections of documents.

A simple approach can be classifying documents based on the similarity to the training data. A simple solution can be to use K-Nearest Neighbors (Cunningham and Delany, 2021) as algorithm which allows to classify new documents. K-Nearest Neighbors (KNN) is a simple and popular classification algorithm in machine learning. The basic idea behind KNN is to classify a new data point by finding the K closest data points in the training set and using their labels to determine the label of the new point. The K in KNN refers to the number of nearest neighbors to consider when making a classification decision. This is a non-parametric algorithm, meaning that it does not make any assumptions about the underlying distribution of the data.

Another simple classifier can be build using the Rocchio Classifier (Zeng and Huang, 2011). The Rocchio classifier is a simple and effective algorithm for document classification. It is a supervised learning algorithm that works by learning a set of weight vectors representing each class based on a training set of labeled documents. When a new, unlabeled document needs to be classified, the algorithm compares the document's feature vector to each of the class weight vectors and assigns the document to the class with the closest weight vector. The key advantage of the Rocchio classifier is that it can be very effective for text classification tasks with relatively small datasets, where the training data is limited. It is also relatively simple to implement and has been shown to perform well in practice.

A more complex solution, but still widely applied in the literature is the use of Support Vector Machines (SVM). This is a popular supervised learning algorithm used for classification, regression, and outlier detection tasks. The key idea behind SVM is to find a hyperplane that separates the data points into different classes, with the largest possible margin between the hyperplane and the closest data points of each class. SVMs work by transforming the input data into a higher-dimensional space where it is easier to find a hyperplane that separates the data points. The hyperplane is defined by a set of weights and biases that are learned from the training data using an optimization algorithm.

Graph Neural Networks (GNNs) have emerged as a recent trend in machine learning, particularly in the domain of graph and network analysis. GNNs are a class of neural networks that operate directly on graph structures, allowing them to capture the complex relationships and interactions among entities in a graph. The key idea behind GNNs is to use graph convolutional operations to learn representations of nodes and edges in a graph. These representations can then be used for a variety of tasks, such as node classification, link prediction, and graph classification. GNNs can also incorporate information from the graph structure, such as the degree of nodes and the connectivity of edges, to improve the quality of the learned representations. GNNs have shown impressive performance in a variety of tasks, including recommender systems, drug discovery, and social network analysis. They are particularly useful for tasks that involve graph-structured data, such as citation networks, social networks, and biological networks.

3 System description

The first approach is based on the application of different ML techniques on the latent semantic representation of documents, specifically k-NN, Rocchio Classifier and SVM. In particular, to implement the classifiers we relied on the Scikit-Learn library (Buitinck et al., 2013). The library has also been used to obtain the set of features that will be passed to the models for training and converted using the *TfidfVectorizer* to obtain the term-document matrix, which was fed to the *TruncatedSVD* of the same library which performed the SVD.
In the second architecture, two different transformer models were testd. For the implementation and training of the models, the *transformers* library provided by HuggingFace was used, with Pytorch as backend. In detail, the architecture tested were:

- **DistilBERT base (cased)** (Sanh et al., 2019): DistilBERT is a smaller and faster version of the popular BERT model for natural language processing. It is trained using a technique called knowledge distillation, which involves training a smaller model to mimic the behavior of a larger, more complex model. Despite its smaller size, DistilBERT achieves competitive performance on a variety of NLP tasks, including question answering, sentiment analysis, and text classification. It also has significantly faster inference times than the original BERT model, making it more practical for real-world applications. The key difference between DistilBERT and BERT is the size of the model and the number of parameters. DistilBERT has about half the number of parameters as the original BERT model, which makes it more efficient to train and use. However, this comes at the cost of some reduction in accuracy and the ability to handle longer sequences of text.

- **BERT base model (cased)** (Devlin et al., 2018): BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based neural network model for natural language processing tasks such as language understanding and sentiment analysis. It's trained to understand the context of a word based on the words that come before and after it in a sentence, allowing it to achieve state-ofthe- art performance on a wide range of NLP tasks. BERT models can be fine-tuned for specific tasks such as question answering and named entity recognition, it was trained on a large corpus of text data such as Wikipedia and BooksCorpus, and it has been considered a breakthrough on many NLP tasks, as well as being able to handle multiple languages. The main advantage of BERT is its ability to understand the context of a word in a sentence, which enables it to be highly accurate in various NLP tasks.

The last approach consists in the training of a small Graph Neural Network. This approach requires converting the documents into graph data and feeding them to the a network. GNNs use a message passing mechanism that allows information to flow through the graph structure, aggregating node-level information to form a global representation of the graph. To convert the document into graphs, defining the representation and train the models, PytorchGeometric (Fey and Lenssen, 2019) has been used.

4 Data

The Text REtrieval Conference (TREC) Question Classification dataset consists of 5,500 labeled questions in the training set and 500 additional questions in the test set. The dataset contains 6 broad categories and 50 more specific categories.
To evaluate the performance of our approaches against state-of-the-art methods, we will evaluate based on the broad categories instead of the fine categories. The data is collected from four sources:

- 4500 English questions published by USC
- Approximately 500 manually constructed questions for a few rare classes
- 894 questions from TREC 8 and TREC 9
- 500 questions from TREC 10, which serve as the test set

The 6 broad classes we will use are represented as integer values and they are the following:

0. Abbreviation (ABBR)
1. Entity (ENTY)
2. Description and abstract concept (DESC)
3. Human being (HUM)
4. Location (LOC)

5. Numeric value (NUM)

Since the validation set was not provided, it was created from the original training set. In fact, the training set was split considering 20% of data as validation set.
After dividing the sets, the distribution of classes in the training set has been inspected, to determine if there are unbalances in the data. The analysis revealed that there are imbalanced classes, with the class labeled 0 having fewer instances compared to the other classes. Despite these imbalances, we chose not to perform any oversampling or undersampling techniques in order to balance the data.
To enhance the performances of ML approaches and GNN, a preprocessing step has been applied to the data, whereas the use of raw input is prefereable when working with transformers. Here is the list of the preprocessing pipieline that has been applied to the data:

- Conversion to lowercase;
- Removal of contracted forms using the Contractions python library;
- Lemmatization;
- Stopwords removal.

5 Experimental setup and results

As described previously, we considered three main kind of approaches to tackle the problem: we exploited two different Transformer models, we developed a GNN and we used some machine learning classification techniques on the latent space. Starting with the latter class of approaches, we used three different classification algorithms on the latent space: Rocchio Classifier, k-NN and SVM. The first approach we tested was k-NN, to find the best set of parameters to compute the configuration of dimensions for the LSA, a grid search approach was used. This involved testing different combinations of dimensions and selecting the configuration that produces the best results on the validation set. Additionally, also the optimal number of neighbors for the k-Nearest Neighbors was tuned. This produced the configuration we used with the k-NN classifier and the number of dimensions for the latent semantic analysis to be 7 neighbors and 500 dimensions. With the following configuraion we achieved an accuracy of 0.672. The second tested algorithm is Rocchio Classifier, this was tested to overcome the main limitation of the k-NN algorithm which is the big inference time needed to compute the prediction. The classifier was tested without performing just the tuninig on the metric used to compute the distance with respect to the prototypes. Also this simple method produced good results, achieving an accuracy of 0.638. The last algorithm of this category that was tested is SVM. Given the huge number of parameters that can be tuned for this classifier, a randomized search was used to tune the model instead of a grid search. This method allows to search in the space of the possible combination of parameters in less time than the grid search. After tuning the model, it produced an accuracy of 0.698. This table can summarize the results obtained with the ML techniques for classification applied on the latent space. As concerns

	Accuracy	**Correct examples (out of 500)**
k-NN	0.638	319
Rocchio Classifier	0.672	336
SVM	0.698	349

the Transformers approach we fine-tuned the two aforementioned methods models for a maximum of five epochs stopping the training earlier in case of overfitting, on average this happened after the first 2/3 epochs as also described in the following article (Devlin et al., 2018). Some tests to choose the optimal hyperparameters were conducted, in detail, different batch sizes, learning rates and optimizers have been tried, obtaining in both case the following configuration:

- **Optimizer**: AdamW;
- **Batch Size**: 16;
- **Number of epochs**: 3;
- **Learning rate**: 2e-5.

The best results for each model are presented in the table below, showing both accuracy and F1-Score produced by the models:

	Accuracy	**F1-Score**
BERT base	0.978	0.98
DistilBERT base	0.956	0.95

The last approach tested is Graph Neural Network. For this approach the first step is defining the graphs that will be then classified. The graphs were defined for each question and they were formed in the following way:

- Each node corresponds to a term in the question, and it is represented with its representation in the latent space;
- An edge between two terms is created based on a sliding window as suggested in (Rousseau et al., 2015)

Several tests have been made to define the number of hidden dimensions of the network, the dimension of the sliding window, the optimizer, the learning rate and the batch size. After several tests for the possible values, the best configuration obtained was the following:

- **Number of hidden layers**: 512;
- **Optimizer**: Adam;
- **Learning rate**: 2.5e-4;
- **Window size**: 5;
- **Batch size**: 32;
- **Number of epochs**: 10.

The model with the following configuration obtained an accuracy score of 0.716.

6 Discussion

As shown in the previous section, the Transformer approach outperforms both ML methods and the GNN, reaching an accuracy of 0.978 compared to the 0.716 achieved with the GNN solution which performed better than the other approaches.
The ML methods showed to be a good baseline solution for this problem, with the best results provided by the SVM model, even without an in-deep search of the optimal configuration through a grid search in the space of the parameters, which may allow to obtain an even higher accuracy score.
The score obtained with the GNN solution does not represent a fully satisfactionary result, since despite the more complex architecture with respect to a standard machine learning algorithm it did not provide the expected improvement in performance.
Concerning the Transformer models, BERT base provided the best results in terms of accuracy. The previous state-of-the-art solution was also using BERT base as model but used a different training recipe, fine-tuning the network just for one epoch and this may be the reason for the improved accuracy score.

To have a better understanding of the results obtained with the best model, some error analysis has been made on the test set; of the 500 test data, the model correctly predicted 489 examples, making only 11 mistakes. Out of this 11 mistakes, the majority of them are relative to the class Entity (with label 1), which the model has more problems to classify correctly. This is due to the fact that in this class we have questions that this class has question really different among them, including both questions about religion, plants, sport, languages and so on.

7 Conclusion

In conclusion, the results from this experiment show that the transformer approach, specifically BERT, significantly outperforms both the approaches based on GNN and the use of machine learning techniques in the question classification task. The BERT-based approach performed also slightly better than the current state-of-the-art solution, which was also based on the same model. This is due to the different training recipe used, in particular the previous state of the art fine-tuned BERT just for one single epoch instead of the 3 epochs used in this case. The performances of the other methods are still valuable since they represent some cheaper alternative in terms of computational cost which still provide reliable results in the majority of the cases.
The last solution, based on GNN, was developed to show the capabilities of this emerging trend also for the task of question classification and it just represents a baseline network. Future work may include using a more complex architectures and using a different dataset, because the performances are already saturated, since BERT achieved an accuracy of the 97,8%

References

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.

Pdraig Cunningham and Sarah Jane Delany. 2021. k-nearest neighbour classifiers - a tutorial. *ACM Computing Surveys*, 54(6):1–25.

Vivek Datla, Sadid A. Hasan, Yassine Benajib Joey Liu, Ashequl Qadir Kathy Lee†, and Oladimeji Farri Aaditya Prakash. 2016. Open domain real-time question answering based on semantic and syntactic question similarity.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with pytorch geometric. Cite arxiv:1903.02428.

Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. 2018. Weisfeiler and leman go neural: Higher-order graph neural networks.

François Rousseau, Emmanouil Kiagias, and Michalis Vazirgiannis. 2015. Text categorization as a graph classification problem. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1702–1712. The Association for Computer Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Anping Zeng and Yongping Huang. 2011. A text classification algorithm based on rocchio and hierarchical clustering. In *Advanced Intelligent Computing - 7th International Conference, ICIC 2011, Zhengzhou, China, August 11-14, 2011. Revised Selected Papers*, volume 6838 of *Lecture Notes in Computer Science*, pages 432–439. Springer.

Dell Zhang and Wee Sun Lee. 2003. Question classification using support vector machines. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*, pages 26–32. ACM.