Sequence vs. Data augmentation vs. Graph in Text Classification of SemMedDB

Mingchen Li
Department of Computer Science
Georgia State University, Atlanta, USA

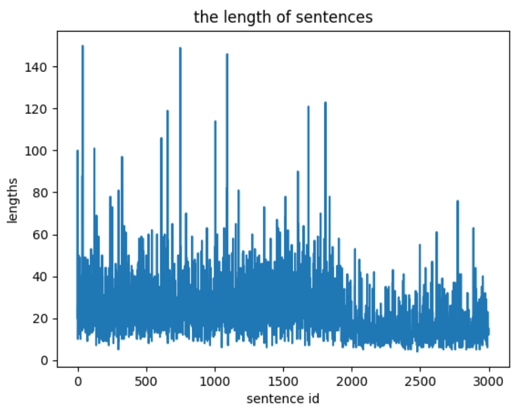

Figure 1: The Statistics of Sentence Length

Abstract

This task aims to filter the predication belonging to the three relations (Interacts_with, stimulates, inhibits) from SemMEdDB [1]. In other words, given a sentence and a triple, the model needs to check whether this triplet matches the sentence. To solve this problem, we regard it as a text classification task. So in this report, we will give a simple analysis of this data set in first, and introduce three methods to solve this problem, at the last, a discussion of our results and limitations will be provided.

1 Data Analysis

As shown in Figure.1, we find the maximum length of sentences is around 140, so this task can be regarded as a short text classification. We used the Sklearn train_test_split function to split data into a training set (2400) and a test set (600). After that, from the training set, we found the number of positive instances is 1284, and the number of negative instances is 1116. So this data is balanced.

[1]https://conservancy.umn.edu/handle/11299/194965

2 Method

In this section, we will provide three methods to solve this task.

2.1 Sequence Model

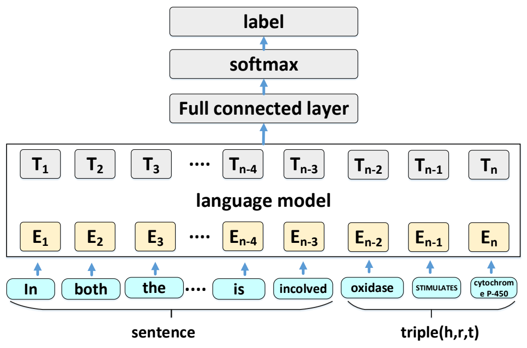

Figure 2: Sequence Model: given a question and its triple, the model predicts the label of this combination.

Given a question q and a triple t, our aim is to predict its label, which is a two-class classification problem that classifies question q and triple t to one of the positive label and negative label.

Figure 2 depicts the architecture of the Sequence Model. The input to Sequence Model is question q and its triple t. The output of the Sequence Model is a probability distribution over two labels, i.e., $p(y|(q:t),\theta)$, where θ denotes the model parameters of the Sequence Model. This model encodes the question and triple $q:t$ (: is the concatenation operation) by the language model, such as GPT2 (Radford et al., 2019), and the final hidden state is used as the question triple embedding s. Finally, the latent representation s is fed to a fully connected layer, followed by a softmax for classification. The whole network is fully differentiable and can be optimized by minimizing the traditional cross-entropy loss.

2.2 Data Argumentation

In this section, we will provide three methods to do the data argumentation, to rich the train data.

- Strategy 1. **Random Selection**. There are four operations:1) Randomly swapping the positions of the words in the sentence. 2) Randomly removing words from the sentence.3) Randomly inserting a random synonym of a random word at a random location. 4) Randomly replacing words with their synonyms.
- Strategy 2. **By Wordnet**. To augment the sentence by replacing entities with synonyms from the WordNet.
- Strategy 3. **CharDelection**. By removing some parts of the sentence to make an argument sentence.

2.2.1 Graph Model

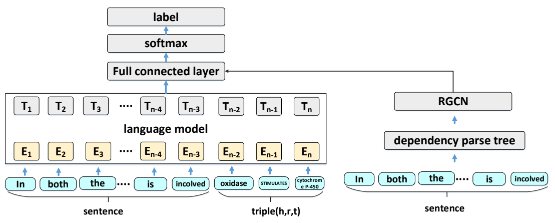
label
softmax
Full connected layer
T_1 T_2 T_3 ···· T_{n-4} T_{n-3} T_{n-2} T_{n-1} T_n
language model
E_1 E_2 E_3 ···· E_{n-4} E_{n-3} E_{n-2} E_{n-1} E_n
In both the ···· is incolved oxidase STIMULATES cytochrome P-450
sentence
triple(h,r,t)
RGCN
dependency parse tree
In both the ···· is incolved
sentence

Figure 3: Graph Model: given a question and its triple, the model predicts the label of this combination.

Graph neural networks have triggered great progress in graph-based text classification methods, such as TextGCN(Yao et al., 2019), TensorGCN (Liu et al., 2020). It makes sense to explore the application of graph neural networks in this task. So how to build the graph from a sentence? In this work, we construct the dependency parse graph for the question and use the relational Graph Convolutional Network (RGCN) (Schlichtkrull et al., 2018) to learn the semantic representation of the graph, and then we combine the information calculated by the language model to predict the label.

To encode the information of the dependency parse graph, and learn the interaction information of each node, we utilize the relational Graph Convolutional Network (RGCN) (Schlichtkrull et al., 2018) as graph convolution. RGCN allows capturing the edge information so that we can consider the differences in the incoming and outgoing relations. In particular, the node representation g_n^0 is initialized by calculating the embedding of nodes by the pre-trained model BERT [2], the embedding will be updated during the training progress, and then the node representation g_n^l is computed as:

$$g_n^l = \sum_{r \in \mathcal{R}} \sum_{u \in N_r(n)} \frac{1}{|N_r(n)|} W_r^l h_n^l$$

where $N_r(n)$ denotes the set of neighbors of node n under relation $r \in \mathcal{R}$, $\mathcal{R}$ denotes the relation set. $W_r^l \in \mathbb{R}^{t \times d}$ calculate the edge information between the node u and node n. By the above operation, the information of dependency parse graph gp will be encoded as $gp^v \in \mathbb{R}^{z \times d}$, where z is the number of nodes in the dependency parse graph, $d = 1024$ is the embedding dimension. So, the output of the Graph Model is a probability distribution over two labels, i.e., $p(y|(s + gp^v), \theta)$.

3 Experiments

3.1 Datasets

Annotated SemMedDB, we split this data set into train set (2400) and test set (600).

3.2 Hyperparameter Settings

Sequence Model we gave an example for the classification by using Roberta-large to learn the semantic representation for question and triple. For other models please check the code file, and use the default hyperparameters.

We set the dropout rate to 0.1, batch size to 32, and use AdamW optimizer (Loshchilov and Hutter, 2017) with the learning rate of 5e-8. We also apply gradient clipping to constrain the maximum value of L_2-norm of the gradients to be 1.

These experiments are run in Quadro RTX 8000.

Graph Model In the output of the RGCN layer, we add a Normalization Layer, and q Dropout Out Layer, and we set the dropout rate to 0.5. The batch size is set to 32, and the AdamW optimizer (Loshchilov and Hutter, 2017) also be used in this model.

3.3 The performance of Sequence model

In this work, the language models (BERT, DistilBERT, CamemBERT, RoBERTa, GPT2) were used

[2]https://huggingface.co/docs/transformers/model_doc/bert

			TestSet			TrainSet		
baseline	min	memory	Prec	Recall	F1	Prec	Recall	F1
Bert-base	11.95	6699M	0.74	0.76	0.69	0.98	0.97	0.97
Bert-large	30.80	17009M	0.77	0.75	0.76	1.00	1.00	1.00
DistilBERT-base	2.58	4085M	0.72	0.68	0.69	0.82	0.78	0.81
CamemBERT-base	5.54	6745M	0.73	0.72	0.69	0.83	0.84	0.81
RoBERTa-base	8.46	6957M	0.71	0.64	0.74	0.95	0.90	0.94
RoBERTa-large	45.27	17293M	**0.81**	**0.78**	**0.79**	1.00	1.00	1.00
Gpt2-base	82.39	10005M	0.77	0.71	0.76	1.00	0.99	1.00
Gpt2-large	882.39	21833M	0.77	0.74	0.76	1.00	1.00	1.00

Table 1: The performance of sequence models

			TestSet			TrainSet		
baseline	min	memory	Prec	Recall	F1	Prec	Recall	F1
RoBERTa-large	45.27	17293M	0.81	0.78	0.79	1.00	1.00	1.00
Strategy 1.	142.04	17293M	0.80	0.75	0.80	0.94	0.90	0.94
Strategy 2.	54.10	17293M	**0.82**	**0.80**	**0.80**	0.98	0.98	0.98
Strategy 3.	102.73	17293M	0.79	0.78	0.77	0.98	0.98	0.98
Strategy(1.+2.+3.)	104.16	17293M	0.80	0.77	0.78	0.96	0.95	0.96

Table 2: The performance of RoBERTa-large with different data argumentation strategies.

to learn the semantics of sentence and triple. As shown in Table 1, we summarized the running time and the occupied memory of each model. For the GPT-2 large, we found if we set the batch size is 32, the occupied memory will be beyond 48G, so we set the batch size to 8 for this model. It consumes a lot of time and memory, however, the performance is not the best. We found that just using Roberta-large, the model can get the highest F1 score in the testing set.

3.4 The performance of different data augmentation strategies.

label	source	St1.	St2.	St3.	St(1+2+3)
y	1284	6420	2568	2568	8988
n	1116	5580	2232	2232	7812

Table 3: Statistics of training set after doing the data argumentation, St1, St2, St3 refers to the Strategies in section 2.2 Strategy(1+2+3) refers to using Strategy1, Strategy2, and Strategy3 together.

In this section, we compare the performance of three data argumentation methods and use the sequence model RoBERTa-large to learn the representation of sentence and triple. As shown in Table 3, we summarised the number of training datasets about different labels by these argumentation strategies. From Tabel 2, we found that using the knowledge graph such as wordnet to enhance the sentence by replacing entity with synonyms can get the sota performance. For Strategy 1, we found the operation of randomly swapping the position of the word can change the semantics of the sentence, so the performance is lower than Strategy 2.

3.5 The performance of Graph model.

Some work such as TextGCN (Yao et al., 2019) shows that the graph neural network can improve the performance of text classification, however, the latest work (Galke and Scherp, 2022) shows that the graph neural network is invalid. For this situation, we should analyze the data, and why the graph neural network is valid or invalid, if we regard the sentence as a graph, this graph can be a dependency parse graph, a knowledge graph, or a full-connection graph, in our work we just explored the dependency parse graph and full-connection graph, the knowledge graph is not easy to extract. By these graphs, it can learn the different semantic relations for different word nodes or entity nodes,

			TestSet			TrainSet		
baseline	min	memory	Prec	Recall	F1	Prec	Recall	F1
RoBERTa-large	45.27	17293M	0.81	0.78	0.79	1.00	1.00	1.00
RoBERTa-large-argument-GRN	59	19489M	0.78	0.72	0.79	1.00	1.00	1.00

Table 4: The performance of graph model

on the one hand, these semantic relations may help classification, but this method is not suitable for the long document. On the other hand, we think the graph is not easy to learn the semantic representation because the relations for different nodes in this graph are not easy to build. Otherwise, we know that the output of the language model is the semantic matrix of the token, but the graph is built at the word or entity level, combining the token to the word entity level is another problem.

4 Conclusion

This is a normal task about classification task, I just used three methods to solve this problem. There are some limitations to these methods.

- **Data**. The data is not enough, we should consider a more effective way because this data is about medicine. The medicine knowledge graph or the medicine language model can be considered. Boosting models and ensemble learning can be used to improve model performance.
- **Is it right to regard this task as a classification task**. In this task, I combined the triple with the sentence and used the transformer model to learn the fully-connected graph, another solution, we can do the classification of relations, and extract the entities of this sentence. We should note that the head entity and tail entity have the orders.
- **To improve the robustness of the classification model**. We can use contrastive learning to increase the differentiation of labels.

For further work, I have some suggestions. 1) If a sentence contains many triples, how to extract them, to transfer the sentence to triple? 2) How to use the pre-trained transformer models to do the Data Augmentation, we can refer (Kumar et al., 2020). 3) the classification under the larger document. 4) zero shot or few shot medical classifications. 5) multi-label classifications.

References

Lukas Galke and Ansgar Scherp. 2022. Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide mlp. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4038–4051.

Varun Kumar, Ashutosh Choudhary, and Eunah Cho. 2020. Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*.

Xien Liu, Xinxin You, Xiao Zhang, Ji Wu, and Ping Lv. 2020. Tensor graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8409–8416.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pages 593–607. Springer.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377.