[CL-Aff Shared Task] Detecting Disclosure and Support via Deep Multi-Task Learning

Weizhao Xin[0000−0003−1712−2218] and Diana Inkpen[0000−0002−0202−2444]

University of Ottawa
{wxin074,diana.inkpen}@uottawa.ca

Abstract. We propose a novel way of deploying deep multi-task learning models for the task of detecting disclosure and support. We calculate all possible logical relations among six labels, represented in a Venn diagram. Based on it, the six labels are distributed to multiple fragment clusters. Then, a multi-task deep neural network is built on the groups.

Keywords: Deep Multi-Task Learning · Natural Language Processing · Word Embeddings.

1 Introduction

Deep Learning (DL) has achieved great success in many fields, including, but not limited to, natural language processing, computer vision, and speech recognition. But there are still many limitations and challenges related to training DL models, such as overfitting, hyperparameter optimization, long training times, high memory usage, etc.

Even if we do not consider the high demands for computing power, there is still some interesting techniques in the classical neural network models to improve the performance, like, deep multi-task learning structures. Multi-task learning (MTL), particularly with deep neural networks, can not only reduce the risk of overfitting, but also improve the results for each task, compared with single-task learning. [7]

Switching the topic to the 2020 CL-Aff Shared Task [3], the inspiration of this shared task is the growing interest in understanding how humans initiate and hold conversations. We want to know people's reactions, both in terms of emotion and information. As task 2 is an open-ended problem, we will only focus on task 1 in this paper.

For task 1, the OffMyChest conversation dataset is provided. Twelve thousand samples are included in this dataset, and each entry contains a sentence and six binary labels, including *Information_disclosure, Emotion_disclosure, Support, General_support, Info_support*, and *Emo_support.*

2 Data Preprocessing

The distribution of the six labels in training data is shown in table 1. We can see that in all the labels, the negative data accounts for a high proportion, especially for the label *General_support*, in which the negative data reaches a proportion of 94.7%. This tells us to pay attention to the class weights during training. Nonetheless, it is likely that the result on the label General_support will be among the lowest since it has the highest class imbalance.

Table 1. Data Distribution in the Training Set

Label	True	False
Emotional_disclosure	3948	8912
Information_disclosure	4891	7969
Support	3226	9634
General_support	680	12180
Info_support	1250	11610
Emo_support	1006	11854

Table 2 shows the token analysis of the training dataset. As shown in the table, although the maximum token length reaches 171, 95% percent of sentences have a length of no more than 34. So the objective length in sentences preprocessing is 34. After retrieving the word embedding vectors, all sentences will be transformed into vectors with shape ($34 \times embedding\_dimension$).

Table 2. Word Preprocessing Results

	Value
Max token length	171
Min token length	1
Mean token length	15.07
Median token length	13
Number of unique tokens	11460
Total number of tokens	193837
Length that covers 95% of sentences	34

3 Word Embedding

Preprocessing steps included standard operations like splitting the sentences into words, omitting all punctuation marks, transforming the sentences into sequences and padding them to the same length, 34. The word embedding method we chose is BERT [2]. Unfortunately, as the GPU we had available did not have sufficient memory, we could not use the BERT embedding as a layer in the model. Instead, we use bert-as-service [9] to do word embeddings beforehand. By losing some performance in value updating, this allowed us to use less memory and reduced the computation time. We used the default configuration and the bert-as-service configuration called "ELMo-like contextual word embedding". The former one aims to generate sentence embeddings and the latter one will create embeddings which have similar shape as the ELMO embeddings [6], in other words, it will generate separated embedding for all words in the padded sentences. We obtained two word embedding files, one with shape $12,860 \times 1,024$, and the second one with shape $12,860 \times 34 \times 1,024$. $12,860$ is the number of instances in the training set, while 34 is the objective length we defined for each sentence and $1,024$ is the dimension of each word (or of the whole sentence in the default configuration).

4 Models

In the model, we want to fully utilize the power of the neural networks on multitasking with hard parameter sharing [7].

In general, when training a model on a task using noisy datasets, we need to ignore the data-dependent noise and to learn good patterns based on other features. Because different tasks have different noise patterns, a model trained for multiple tasks can generate a more general representation and can average the noise patterns on the different tasks [7].

Furthermore, as similar tasks have similar patterns, we want their task-specific layers to be at a closer position compared with other tasks in the model. For example, among the six labels of our shared task, it is easy to image that the label *Support* has a strong relationship with the label *General_support*, *Info_support* and *Emo_support*, as they all refer to something about *support*. Considering each label as a set, which contains entries in the training data where the corresponding label is *1*, the relationship among four labels can be described by the Venn diagram in Fig. 1. The numbers on the graph show the size of intersections between or among the sets. Venn diagram [1] is a simple diagram used to represent unions and intersections of sets. However, based on the number of sets, it could be extremely complicated and we will see it below.

From the Fig. 1, we can see that the label *Support* almost covers all the cases in the other three sets, except for some trivial cases. Then how to reflect this relationship in the neural network? Because the label *Support* covers a more general concept, it should be treated at a lower layer. The other three labels refine information from the *Support* layer, using part of its information, while sharing some neurons between themselves, as shown in Fig. 2. The bottom of the Fig. 2 is a large dense layer, which is split into several parts; we call it a fragment layer. Above the fragment layer in Fig. 2 are territories of four labels; this means that the label-corresponding task-specific layers will only connect to their specific territories (neuron s).

Each label's territory has some overlap with other labels' territories, and the label *Support* occupies the whole layer, from the leftmost node to the rightmost node.

The example above is only for four labels. Nevertheless, *Support* can contain all the other three, which means that the intersections only appears among three layers. What about the six labels in our task? The Venn diagram is far less clear, as shown in Fig. 3. Discarding the pieces in the figure whose size is too small (intersections comprised of less than 10 instances), there are in total 31 major intersections in the Venn diagram. And for six-labels, each label is comprised of 15, 16, 28, 12, 12, and 15 intersections separately.

Roughly, from the bottom to the top, the network contains the input layer, shared hidden layers, task-specific layers and task-specific outputs. The connection between shared hidden layers and task-specific layers is based on the fragments and the Venn graphs, as shown above.

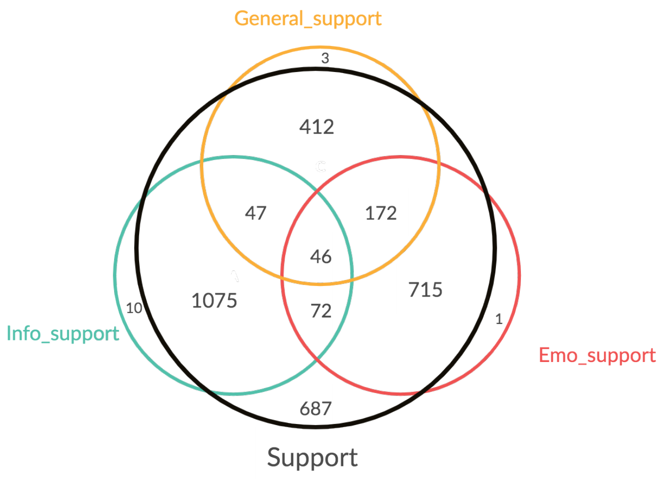

Fig. 1. Venn graph for label *Support, General_support, Info_support* and *Emo_support*

5 Experiment

5.1 Structure

As we have two types of embeddings, of shape (12860×1024) and shape $(12860 \times 34 \times 1024)$, we tried two types of models in the experiment.

For the data with shape (12860×1024), from the bottom to the top, we have an input layer, fragment dense layers, concatenate layers and output layers. As we

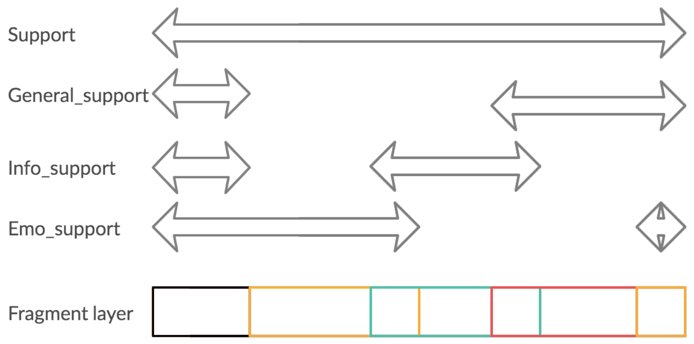

Fig. 2. An example showing how fragment layer works for *Support*, *General_support*, *Info_support* and *Emo_support*

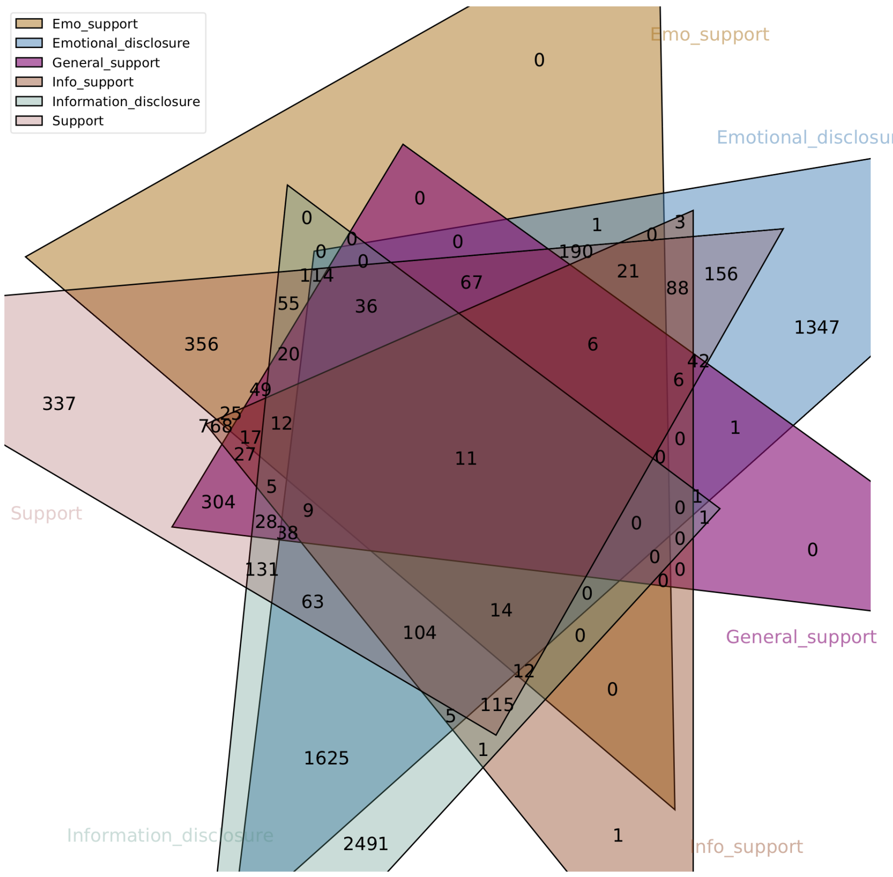

Fig. 3. Venn graph for all six labels

mentioned above, the embedding layer is not included in the model because the word embeddings are already generated in preprocessing, using bert-as-service.

For the data with shape $(12860 \times 34 \times 1024)$, from the bottom to the top, the model is composed of an input layer, bidirectional LSTM layer, fragment dense layers, concatenate layer, attention layer [8], flatten layer and output layer.

5.2 Training

During training, we use mini-batch gradient descent with size at least 512 and the Adam optimizer [4] is used with a learning rate of 0.0001. The loss function is *binary crossentropy* and activation function used in the model are mostly *leaky relu* [10], except the output layers, which use *sigmoid* function.

5.3 Results and Parameter Description

We evaluate the performance of models on the training dataset. The split ratio we used is 0.6:0.2:0.2, which means 60% of data is used for training, 20% for validation and 20% for testing.

Table 3 shows the result of two models. The parameters used in Model 1 are: learning rate is $2e^{-5}$, epochs is 20 and batch size in mini-batch SGD is 1024. The corresponding file name in submission is *system_runs_uottawa1*. The parameters used in Model 2 are: learning rate is $2e^{-5}$, epochs is 20 and batch size in mini-batch SGD is 512. The corresponding file name in submission is *system_runs_uottawa2*.

Table 3. Experiment result of two models

	Label	accuracy	precision	recall	F1
Model 1 for data with shape (12860×1024)	Emotional_disclosure	0.6640	0.4697	0.7440	0.5758
	Information_disclosure	0.7000	0.6048	0.6429	0.6233
	Support	0.8205	0.6153	0.7446	0.6738
	General_support	0.8975	0.2260	0.3884	0.2857
	Info_support	0.8982	0.4857	0.5251	0.5046
	Emo_support	0.9305	0.5081	0.4181	0.4587
	Macro scores	0.8184	0.4849	0.5771	0.5203
Model 2 for data with shape $(12860 \times 34 \times 1024)$	Emotional_disclosure	0.6847	0.4902	0.7133	0.5811
	Information_disclosure	0.7012	0.6110	0.6215	0.6162
	Support	0.8233	0.6213	0.7436	0.6770
	General_support	0.9272	0.3023	0.2902	0.2961
	Info_support	0.8772	0.4177	0.6181	0.4986
	Emo_support	0.9284	0.4920	0.5151	0.5033
	Macro scores	0.8236	0.4890	0.5836	0.5287

From the table, we can see that *General_support* always has the worst result. This is resasonable, considering the imbalance of this label, in which the positive cases:negative cases is 680:12,180.

6 Conclusion and Future Work

In this paper, we presented a multi-task deep learning model. Our model has a reasonable result on some of the labels, but not all, especially not for *General_support*. The reason is that *General_support* classifies quotes and catchphrases, which have less distinctive features than *Emotional_XXX* or *Information_XXX*, while having less positive cases appearing in the dataset.

During the experiments, we tried several other methods for training models, for instance, using LIWC [5] as auxiliary input/output to assist the main tasks. Elmo embeddings and GloVe embeddings were also tried, and a combination using Elmo. A transformer as the classification model was also tested. But they could not improve the performance, unfortunately.

One possible direction of further work for this task is to make use of large unlabeled data provided. An idea here is to use it to find the different patterns in the texts in order to split them into several groups, and then to train models on each group. Sentences can have different patterns and structures, which requires different mapping functions in the network. If we can separate them into clusters in which sentences have similar patterns, there might be an improvement in the classification results.

References

1. Xxv. on the diagrammatic and mechanical representation of propositions and reasonings. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science **10**(61), 168–171 (1880). https://doi.org/10.1080/14786448008626913, https://doi.org/10.1080/14786448008626913
2. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. CoRR **abs/1810.04805** (2018), http://arxiv.org/abs/1810.04805
3. Jaidka, K., Singh, I., Jiahui, L., Chhaya, N., Ungar, L.: A report of the CL-Aff OffMyChest Shared Task at Affective Content Workshop @ AAAI. In: Proceedings of the 3rd Workshop on Affective Content Analysis @ AAAI (AffCon2020). New York, New York (February 2020)
4. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014), http://arxiv.org/abs/1412.6980
5. Pennebaker, J., Boyd, R., Jordan, K., Blackburn, K.: The development and psychometric properties of liwc2015 (09 2015). https://doi.org/10.15781/T29G6Z
6. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations. In: Proc. of NAACL (2018)
7. Ruder, S.: An Overview of Multi-Task Learning in Deep Neural Networks. ArXiv e-prints arXiv:1706.05098 (Jun 2017)
8. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR **abs/1706.03762** (2017), http://arxiv.org/abs/1706.03762
9. Xiao, H.: bert-as-service. https://github.com/hanxiao/bert-as-service (2018)
10. Xu, B., Wang, N., Chen, T., Li, M.: Empirical evaluation of rectified activations in convolutional network. CoRR **abs/1505.00853** (2015), http://arxiv.org/abs/1505.00853