[CL-Aff Shared Task] Happiness Ingredients Detection using Multi-Task Deep 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 deep multi-task learning model for the task of detecting happiness ingredients. The two classes/labels "agency" and "social" are treated as two separate tasks for training Deep Learning classifiers. Then, we train a multi-task deep learning classifier to see if the shared knowledge between the two tasks can improve the overall results. In addition, we compare several models that use different kinds of word embeddings: different dimensions of the vectors, fixed versus trainable embeddings, initialized randomly or with existing embeddings.

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

1 Introduction

Deep learning has achieved great success in many fields, such as natural language processing, computer vision and speech recognition. But there are still many limits and challenges in deep learning, including overfitting, hyperparameter optimization, and sometimes, long training time. Multi-task learning (MTL), particularly with deep neural networks, greatly reduces the risk of overfitting. [6] With multiple tasks being learned simultaneously, our model will try to capture the representation of all the tasks, which significantly lowers the chance of overfitting on each task.

Happiness is one of the important facets of human emotion. In psychology, it is a certain state of mind. The descriptions of happy moments include events that give satisfaction, pleasure, or a positive emotional condition. For the purpose of natural language processing (NLP) tasks, it is difficult to formalize happiness. However, as human affect is context-driven, what we are concerned with here is the contextual and agentic attributes of the descriptions of the happy moments.

The CL-Aff Shared Task aims to challenge the current understanding of emotion and affect in text through a task that models the experiential, contextual, and agentic attributes of the crowd-sourced single-sentences that describe happy moments. The CL-Aff shared task is based on HappyDB [1], which is a corpus of more than 100,000 happy moments crowd-sourced via Amazon's Mechanical Turk. There are two sub-tasks in the shared task for analyzing happiness and well-being in written language on the modified HappyDB corpus. Task 1

is focused on predicting agency and social labels (classes), while task 2 is open ended, encouraging participants to propose new characterizations and insights for the descriptions of the happy moments in the test set. In task 1, beside of two existing tasks, we introduced determining the *Concepts* as the third task, to take full advantage of the power of multi-task learning.

2 Data Preprocessing

The distribution of the social and agent labels in training data is shown in table 1. We can see that whereas the data are almost evenly distributed for the attribute *Social*, for *Agency*, the positive data accounts for a high proportion. This imbalance could causes the performance of our models on *Agency* to not be as good as the performance on the *Social* label.

Table 1. Distribution in Training Data

Label		*Agency*		Sum
		yes	no	
Social	yes	3554	2071	5625
	no	4242	693	4935
Sum		7796	2764	10560

We performed two main steps for the happy moments processing:

- **Split the sentences into word lists and omit all punctuation marks.** Sentences are processed one by one into arrays of words. All punctuations, including comma, period, exclamation mark, question mark and so on, are discarded. A special case is that all abbreviations, like *I'm*, and *we're*, remain unchanged.
- **Transform the sentences into sequences and pad them to become of the same length.** Because a mathematical model can only deal with numbers, the second step is to turn sentences into numbers. First, we number all the words in sequence, starting from 1. (Index `0` is reserved for padding and unknown characters.) Then, we replace all words with their indexes and we pad each sentence at the beginning, as needed, to make them have the same lengths. In this shared task, combining the training and the test data, there are 12,705 unique words in total. The length of the sentences ranges from 1 to 140, with an average length of 14 and a median length of 12. Both of longest and the shortest sentence appear in test data, with hmid 1539 and 4861, respectively. After padding (or cutting), all sentences have the same length of 29, which is no shorter than the actual length for 95% of the original sentences.

We added one step for label processing:

– **Categorise *Agency*, *Social* and *Concepts*.** The class *Agency* and *Social* both contain only binary values: **yes** and **no**, which can be easily transformed into `1` and `0`, whereas the class *Concepts* has `15` different values and many more combinations of them. We use the one-hot encoding method to represent all 15 concepts. Each value will be transformed into a 15-dimension array, in which the locations of the concepts that are present are marked as `1`, and the others are set to `0`.

3 Models

From bottom to top, our model comprises: the embedding layer, the convolutional layer, the dropout and pooling layer, and two detached dense layer heaps. We have compared the results of several deep learning models, like Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM). CNN proved to achieve better results in our experiments, this is why we will present only the results for the CNN-based models in section 4.

3.1 Embedding Layer

The embedding layer is fed with 1-D moment description, which are then embedded into 2-D matrices. The size of the second dimension is 100, which means every word will be transformed into a 100-dimensional vector. For example, for a sentence with a length of 20, we first add 9 zeros in the front to reach the length 29. After passing the embedding layer, the size of the output matrix will be *(29, 100)*.

There are two ways to initialize the values inside the embeddings: randomly, or with pre-trained embeddings from an outside corpus. Then, there are also two ways to handle the values: keep them fixed, or allow them to be updated during training (they are trained for our tasks). From our experiments, using pre-trained embeddings that can be updated during training, lead to the the best results.

GloVe: Global Vectors for Word Representation GloVe is an unsupervised learning algorithm for obtaining vector representations for words. The pre-trained embeddings we used are trained on 2 billion tweets corpus, with 27 billion tokens and 1.2 million vocabulary [5].

3.2 Convolutional Layer

Whereas a 2D convolution layer suits image processing most, here we used a 1D convolution layer, which is usually use for natural language processing [3]. The kernel, also called a filter, has the same length as the input words. After sliding along the input sentence, each filter will generate a 1-D vector. So the size of the final output will only be related with the number of filters and length of a sentence, but not with the dimension of the word embeddings.

After the convolution, the output is fed into a dropout layer and then the max pooling operation is performed.

3.3 Hard Parameter Sharing for Multi-task Learning

Hard parameter sharing [2] is the most commonly used approach to multi-task learning in neural networks. It is applied by sharing the hidden layers between all the tasks, while isolating several task-specific output layers.

Normally, after the convolutional layers, the model will be followed by a fully connected layer and has one output (maybe with multiple dimensions) at the end. But for hard parameter sharing, the first part of model is shared between the multiple tasks, while the layers after convolution are task-specific. Here, the class *Concepts* is treated as the third classification task, besides the two classes *Agency* and *Social*. Fig. 1 is the high level description of our MTL model, for three tasks. The idea is that sharing layers between the tasks could reduce the risk of overfitting for each task. Intuitively, the more tasks we are training simultaneously, the more our model will try to represent all of the tasks, leading to a lower chance of overfitting on a single task. Our proposed MTL model still achieves good results, as shown in the next section.

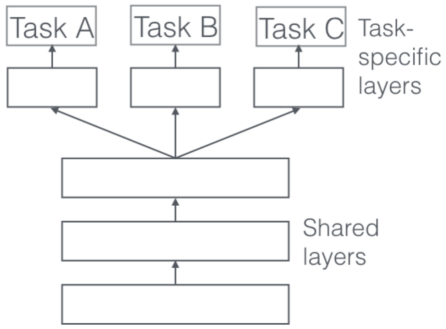

Fig. 1. An example of hard parameter sharing for the three tasks.

4 Experiment

4.1 Training

During training, we use mini-batch gradient descent with size 32 and the Adam optimizer [4] is used with a learning rate of 0.1. The size of the embedding layer was set to 100, and the dropout ratio is 0.2.

4.2 Evaluation and Results

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

Table 2. Experimental Results

Model	CNN		CNN+MTL		CNN+MTL+GloVe, fixed		**CNN+MTL+GloVe,trainable**	
Label	Agency	Social	Agency	Social	Agency	Social	**Agency**	**Social**
Accuracy	0.712	0.770	0.804	0.858	0.831	0.876	**0.835**	**0.892**
AUC	0.770	0.897	0.744	0.856	0.744	0.876	**0.756**	**0.893**
Precision	0.664	0.799	0.759	0.860	0.815	0.876	**0.813**	**0.892**
Recall	0.689	0.778	0.744	0.856	0.744	0.876	**0.756**	**0.893**
F1	0.670	0.767	0.751	0.857	0.767	0.876	**0.776**	**0.892**

Table 2 shows the result of several models. From left to right, the abbreviations of models mean (all word embeddings are of dimension 100 for this set of experiments):

CNN: Convolutional Neural Network model, with randomly initialized embedding layer;

CNN+MTL: Convolutional Neural Network model with randomly initialized embedding layer; followed by multi-task learning layer;

CNN+MTL+GloVe, fixed: Convolutional Neural Network model with embedding layer which is initialized from pre-trained embeddings from GloVe, and values are not allowed to update during training. Followed by multi-task learning layer;

CNN+MTL+GloVe, trainable: Convolutional Neural Network model with embedding layer which is initialized from pre-trained GloVe embeddings, and the values are updated during training. Followed by multi-task learning layer;

From the result, we can see that compared with other models, the CNN model with multi-task learning and pre-trained GloVe embeddings achieves the best result, when the embeddings are allowed to update (they are trained for the two tasks at hand).

Comparing between classes, our model obtains a relatively better result for the class *Social* than for the class *Agency*, probably because of the imbalance in the training data for *Agency*.

All the results in the table are from models with 100-dimension embeddings. We have tested our models on other dimensions, like 50 or 200, and the results did not change much. In other words, the dimension of the embeddings did not affect the model performance significantly. We also tested a two-task multitask learning: the task-specific layers contain only the targets *Agency* and *Social*, without the target *Concepts*. The result of two-task model was very similar with the previous three-task model, which means that target adding the concepts did not contribute much, at least not with the classifying into *Agency* and *Social*.

5 Conclusion and Future Work

In this paper, we presented a multi-task deep learning model. Our experiments show that the model works well on the provided happiness data. We obtained acceptable accuracy, AUC, and F1 scores, especially on the label *Social*.

Although our best model achieved good results, there are still some methods we can use to improve its performance. One direction of future work is to also learn from the unlabelled data (70,000 instances). We did not use the unlabelled in our current model due to time constraints. We propose to use a bootstrap algorithm: to run our current best model on the unlabelled data, then add to the labelled training data the best of the automatically-labelled instances, namely the ones for which the confidence in the prediction is high for both classes (Social and Agency); then to retrain our model on the enhanced training data. The model trained by this bootstrapping method might work better, but only if we do not add too much noise to the training data.

Another direction of future work is to make use of other information provided in the training data, such as age, gender, location, marital status and parental status. Another information from the training data that we plan to use is the provided *concepts*. They are available for the training data but not for the test data. We experimented with detecting concepts as a separate task while training the MTL model, but we could further apply the model to predict concepts on the test data. Then we can use these automatically-detected concepts when we run the model on the test data in order to obtain results for the multi-task model with three tasks.

We mentioned that the sub-task 2 is an open ended task where the participants can propose their own task that could bring insights into the concept of happiness as reflected in texts. As an idea that might be interesting as sub-task 2, that we propose for our future work, is to apply event detection methods to find out what is the event that makes people happy. Then to analyze the events by age, gender, location, marital status, and parental status. This could show what kind of events are important / happy at various ages. We could see what events are considered happy by women, maybe they could be different than what men consider happy events. Cultural events might be different by locations. Married vs. single people might choose different events as important / happy at their current stage in life. Finally, parents could be happy when their children accomplish some developmental milestones, and this kind of events would not show up for people who are not parents.

References

1. Asai, A., Evensen, S., Golshan, B., Halevy, A., Li, V., Lopatenko, A., Stepanov, D., Suhara, Y., Tan, W.C., Xu, Y.: HappyDB: A corpus of 100,000 crowdsourced happy moments. In: Proceedings of LREC 2018. European Language Resources Association (ELRA), Miyazaki, Japan (May 2018)
2. Caruana, R.: Multitask learning: A knowledge-based source of inductive bias. Proceedings of the Tenth International Conference on Machine Learning. (1993)

3. Kim, Y.: Convolutional Neural Networks for Sentence Classification. ArXiv e-prints arXiv:1408.5882 (Aug 2014)
4. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014), http://arxiv.org/abs/1412.6980
5. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014), http://www.aclweb.org/anthology/D14-1162
6. Ruder, S.: An Overview of Multi-Task Learning in Deep Neural Networks. ArXiv e-prints arXiv:1706.05098 (Jun 2017)