Sarina Danaei
Implementation Report
February 8, 2023

Topic-dependent Argument Mining and Classification

UKP dataset classification and analysis

Implementation consists of four main parts:

1. Gathering dataset
2. Dataset transformation, preprocessing and normalization
3. Modeling data on Logistic Regression
4. Modeling data on Neural Network
5. Comparing Machine Learning with Deep Learning

Gathering data set

The UKP ASPECT Corpus includes 3,595 sentence pairs over 28 controversial topics. The sentences were crawled from a large web crawl and identified as arguments for a given topic using the ArgumenText system. The sampling and matching of the sentence pairs is described in the paper. Then, the argument similarity annotation was done via crowdsourcing. Each crowd worker could choose from four annotation options (the exact guidelines are provided in the Appendix of the paper). You can check out the website[1] for more information and cloning dataset.

Dataset transformation, preprocessing and normalization

To ensure the important dataset standards before learning, we need to make sure there is no bias, missing data, noise, and other factors. The UKP dataset is using tab-separated format (tsv) which is not very stable. So we converted the dataset to comma-separated format (csv). This will prevent data mixture and utilizes better structure of selected dataset.

[1] https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/1998

Next, by using libraries such as nltk, preprocessing steps will be as follows:

1. Removing special characters and punctuations
2. Stemming and lemmatization
3. Removing stopwords
4. Removing numbers
5. Label encoding
6. Split dataset into train validation and test

By doing these steps, we make the data ready for modeling.

Modeling data on Logistic Regression

Using TF-IDF method, we try to classify the arguments with ML. After vectorizing prepared data using TfidfVectorizer, in order to get the best results, first we need to find best parameters' values. Grid search method is the way to get the optimal state and by cross-validating them, we get even closer to optimal values.

After compiling the model and run it on test set, the below result has been made:

```
Classification report
                     precision    recall  f1-score   support

        3d printing       0.88      0.88      0.88        16
           Big data       0.93      1.00      0.96        25
      Cloud storing       0.83      0.88      0.86        17
     Cryptocurrency       0.77      0.89      0.83        19
             Drones       0.89      0.94      0.92        18
      Electric cars       0.85      0.85      0.85        20
   Electronic voting       1.00      0.91      0.95        22
           Fracking       0.93      0.82      0.87        17
       Gene editing       0.85      0.85      0.85        13
  Genetic diagnosis       0.91      0.87      0.89        23
     Geoengineering       1.00      0.80      0.89        20
                Gmo       0.85      0.77      0.81        22
  Hydroelectric dams       0.60      0.86      0.71         7
 Hydrogen fuel cells       0.92      0.89      0.91        27
 Internet of things       0.83      0.83      0.83        18
     Nanotechnology       0.88      1.00      0.93        14
     Net neutrality       0.83      0.83      0.83        24
   Offshore drilling       0.86      1.00      0.93        19
     Organ donation       0.88      0.83      0.86        18
 Public surveillance       1.00      0.95      0.97        20
          Recycling       0.88      1.00      0.94        22
     Robotic surgery       0.85      1.00      0.92        22
     Social networks       0.93      0.81      0.87        16
       Solar energy       0.93      0.88      0.90        16
  Stem cell research       1.00      0.95      0.98        22
  Tissue engineering       1.00      0.94      0.97        17
     Virtual reality       0.96      0.96      0.96        27
         Wind power       1.00      0.84      0.91        19

           accuracy                           0.90       540
          macro avg       0.89      0.89      0.89       540
       weighted avg       0.90      0.90      0.90       540
```

Table 1: Logistic Regression Results on Classifying UKP sentences

Although the results are remarkable, we need to build our deep learning model to compare results in a fair environment.

Hyperparameters	Optimal Values
C	~ 0.73
class_weight	None
multi_class	Multinomial
Penalty	L2
Solver	Sag

Table 2: Optimal hyper parameters values

Modeling data on Neural Network

Using keras and tensorflow libraries, we can build, compile and train the model to compare it with ML. Model architecture is basic but the main point is the simplicity.

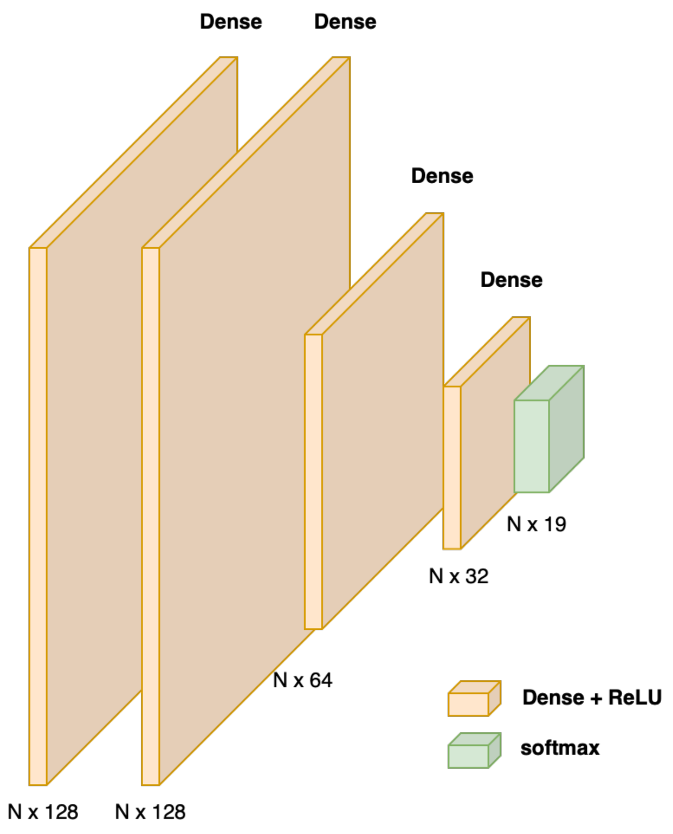

Figure 1: Deep Learning model architecture

Hyperparameters Values	
Epochs	100
Callbacks	On 5
max_features	300

Table 3: Optimal hyper
parameters values

To confirm whether our model overfits the data or not, we train a secondary neural network with several dropout layers and compare the results.

Dropout layer is a tool to regularize neural network, in which a number of layer outputs at random are ignored so that the sparser network will have to adapt to correct mistakes from prior layers. We can grid search the best dropout parameter between the range 0.1-1.0. Here, I used 0.3 as the parameter.

After removing the final dropout layer (the dropout right before the output layer), the performance improves by a few percentage points, and the loss is also reduced across the epochs during training. This is because the output layer can't correct the errors from the previous layers anymore, so adding another dropout layer will end up hurting the model's performance.

```
           0       0.98      0.42      0.59       197
           1       0.96      1.00      0.98        25
           2       1.00      0.82      0.90        17
           3       1.00      0.05      0.10        19
           4       0.47      1.00      0.64        18
           5       0.31      0.85      0.46        20
           6       1.00      0.91      0.95        22
           7       1.00      0.82      0.90        17
           8       0.08      0.85      0.15        13
           9       0.87      0.87      0.87        23
          10       1.00      0.80      0.89        20
          11       1.00      0.45      0.62        22
          12       0.43      0.86      0.57         7
          13       1.00      0.19      0.31        27
          14       1.00      0.83      0.91        18
          15       0.88      1.00      0.93        14
          16       1.00      0.75      0.86        24
          17       1.00      0.95      0.97        19
          18       0.93      0.78      0.85        18

    accuracy                           0.63       540
   macro avg       0.84      0.75      0.71       540
weighted avg       0.91      0.63      0.67       540
```

Table 4: Deep Learning results

Comparing Machine Learning with Deep Learning

- The performance of the model with and without dropout layers are pretty much the same. The diverge between train and test errors don't improve even after using the dropout layers. In fact, with the dropout layer, the model fails to predict the labels for a lot more categories than the model without the dropout layer (many have 0% accuracy). Further analysis can involve fitting models on different dropout values to confirm if the performance doesn't improve because we didn't choose a good dropout rate, or because of the dataset itself.
- The LR, as well as other ML models, don't have that big of a gap between the training and testing errors. It can be said that these ML models don't suffer from overfitting as much as the neural network model, however, these models can hardly be improve further, given that we already use the best parameters. Meanwhile, the neural network stands a good chance of improving beyond this point. Although the results in ML model is way better than the built DL model, and since our interest lies in getting more accurate prediction, we will go with the neural.
- It is observed that the overall accuracy for both models goes up after I added more data points.
- The accuracy shows slight fluctuations across different fitting attempts, but on average, the result is in the range [0.88 - 0.90], which is ~17-34% increase from the average of the results of the deep learning models I fitted (which is ~63-67%). Looking at the confusion matrix, the metrics within each category also improves.
- Another source that might affect the predictive power of the model is the fact that a lot of categories share many words in common, so one of the next steps from here is to identify and develop a collection of stopwords that appear a lot across different categories to reduce the similarity between these categories and see if this increases the performance even further.