CSC 791- 603 Fall'21
Natural Language Processing
Assignment P1 Report

Krishna Prasanna (kbhamid2)

Assignment Task:

There were two labeled text datasets provided in this assignment. The actual sentence/ text comprises of online user reviews for apps. The labels indicate the comparison (explicit/implicit/none) and preference of the user between rival products mentioned in the review sentence.

Problem Approach:

This is a classification problem where the review sentences must be analyzed and classified into their stated labels (Preference/ Comparison).

Baseline Model:

The two datasets provided in assignment description were loaded. Basic understanding of the data-types and columns was gathered. The P1_labeled_dataset was used as the train data and it comprises of 4829 sentences. The P1_test_dataset was used for model testing and performance evaluation and this test data has 1000 sentences. Both train and test datasets have two columns of labels – One for "preferred" and one for "comparison".

The following steps were implemented in baseline model:

- **Data preprocessing:**

  The sentences comprised of unnecessary punctuation marks, white-spaces and inconsistent capitalization.

  The following tasks were performed in pre-processing on each sentence in the train and test datasets:

  1. Punctuations were removed
  2. All words were converted to their lowercase
  3. Extra white-spaces were stripped from the sentences.
  4. The "Preferred" label is a categorical column. For ease of modeling, it has been converted to numerical column using a mapping dictionary.

- **Word and Sentence Embeddings:**

  (A) *Word-to-vector conversion:* The sentences are now clean after the preliminary pre-processing. To be used in classification models, they must be represented in a numeric format, either as a scalar or a numeric vector. This is done by using a word-to-vector like conversion.

  (B) *Word embedding:* Word embedding is a smarter way of converting words into numeric vectors as the words with similar semantic meanings can be designed similar

numeric representation. Word embedding techniques map a word to a fixed-dimensional vector which is a learned representation of that word. The learning happens on a large-corpus of data.

(C) *Sentence embedding:* Sentence embedding is a process of generating a numeric vector representation for the entire sentence. It can be generated by averaging the vector representation of the words present in it.

The following techniques were implemented for generating numeric representation of words and sentences:

1. **Term Frequency- Inverse Document Frequency:** TF-IDF is a basic word-2-vec model that can be run on our training sentence corpus. TF-IDF in this implementation was performed on cleaned sentence. For each sentence it generates a vector (which is equal to the sentence length). TF-IDF was chosen for baseline implementation because unlike the pre-trained models, it generates useful vectors that exploit local structure of corpus.
2. **SpaCy:** Spacy is one of the fastest pre-trained word embedding model which has a pre-trained numeric vector representations for each word. SpaCy package was imported from the web and used to on sentences to generate sentence embedding.

Note: Both the above techniques generate sentence embeddings automatically. We can even extract the individual word embeddings from spacy, although it implicitly averages these word embeddings and generates the corresponding sentence embedding.

- **Dataset preparation for model:**

  The training data comprises only of sentence embeddings. All the other columns are dropped from consideration. The following files were created for both train and test datasets:

  spacy_sentences – An array of arrays which contains spaCy sentence embeddings. The size of outer array is 4829 for train data embeddings and 1000 for test data embeddings.
  tfidf_sentences – An array of arrays which contains tf-idf sentence embeddings. The size of outer array is 4829 for train data embeddings and 1000 for test data embeddings.
  pref_labels – An array for 'Preferred' labels (numeric). The size of array is 4829 for train data and 1000 for test data.

  comp_labels – An array for 'Comparison' labels (numeric). The size of array is 4829 for train data and 1000 for test data.

- **Model selection:**

  The models selected for the classification tasks were:

  1. **Random Forest classifier -** An ensemble-based classification model, for Random Forest classifier the following hyper-parameters were set:
     n_estimators = 10 – Number of weak estimators for our ensemble model
     criterion = 'entropy' - The sentences are classified into buckets based on entropy.

random_state = 42 – Because it is the ultimate answer!

2. **Multinomial Logistic Regression classifier –** The classification algorithm that shows up everywhere.

These models have been selected because they support multi-class classification which is desired for our tasks. All hyper-parameters were selected arbitrarily, no optimization has been performed here.

- **Model Evaluation:**

  The model performances were evaluation on following metrics:

  1. **Accuracy score -** Since we have multi-class classification problem, accuracy is a good evaluation metric to gauge how our models are performing.
  2. **Confusion matrix -** Confusion matrix helps us understand our classification model better and access which classes are being most misclassified.
  3. **Classification report** – This report comprises of some extra evaluation metrics such as F1-score, precision, recall.

- **Model training and implementation:**

  1. There were 8 models run in total – 4 models each for Random Forest, and Logistic Regression.
  2. Out of the four models for each classification technique, two models were covered for each label ('Preferred'/ 'Comparison')
  3. For each label, one model was run for TF-IDF based sentence embeddings, and another model was run for spaCy based sentence embeddings.
  4. All models were evaluated on the above-described evaluation metrics.

- **Baseline model results:**

  The best accuracy metrics have been high-lighted in bolded

  1. Preferred labels – classification:

Accuracy	Random Forest	**Logistic Regression**
spaCY	0.659	0.568
TF-IDF	0.716	**0.753**

  2. Comparison labels – classification:

Accuracy	**Random Forest**	Logistic Regression
spaCY	0.707	0.61
TF-IDF	**0.781**	0.751

From above tables we can infer that TF-IDF word-2-vec based models have a significantly better performance over spaCy based models. Some screenshots of colab results are posted below:

RANDOM FOREST RESULTS

Random Forest Confusion Matrix for Preferences label classification for Spacy embedding:

Predicted	N	O	T
Actual			
N	159	59	72
O	45	219	79
T	37	49	281

Random Forest Accuracy for Preferences label classification for Spacy embedding: 0.659

	precision	recall	f1-score	support
N	0.66	0.55	0.60	290
O	0.67	0.64	0.65	343
T	0.65	0.77	0.70	367
accuracy			0.66	1000
macro avg	0.66	0.65	0.65	1000
weighted avg	0.66	0.66	0.66	1000

Random Forest Confusion Matrix for Comparison label classification for Spacy embedding:

Predicted	0	1	2
Actual			
0	70	39	39
1	18	366	77
2	8	112	271

Random Forest Accuracy for Comparison label classification for Spacy embedding: 0.707

	precision	recall	f1-score	support
0	0.73	0.47	0.57	148
1	0.71	0.79	0.75	461
2	0.70	0.69	0.70	391
accuracy			0.71	1000
macro avg	0.71	0.65	0.67	1000
weighted avg	0.71	0.71	0.70	1000

Random Forest Confusion Matrix for Preferences label classification for TF-IDF embedding:

Predicted	N	O	T
Actual			
N	177	61	52
O	50	248	45
T	37	39	291

Random Forest Accuracy for Preferences label classification for TF-IDF embedding: 0.716

	precision	recall	f1-score	support
N	0.67	0.61	0.64	290
O	0.71	0.72	0.72	343
T	0.75	0.79	0.77	367
accuracy			0.72	1000
macro avg	0.71	0.71	0.71	1000
weighted avg	0.71	0.72	0.71	1000

```
Random Forest Confusion Matrix for Comparison label classification for TF-IDF embedding:
Predicted   0    1    2
Actual
0          69   54   25
1          11  418   32
2           3   94  294


Random Forest Accuracy for Comparison label classification for TF-IDF embedding: 0.781


              precision    recall  f1-score   support

           0       0.83      0.47      0.60       148
           1       0.74      0.91      0.81       461
           2       0.84      0.75      0.79       391

    accuracy                           0.78      1000
   macro avg       0.80      0.71      0.73      1000
weighted avg       0.79      0.78      0.77      1000
```

LOGISTIC REGRESSION RESULTS

```
Logistic Regression Confusion Matrix for Preferences label classification for Spacy embedding:
Predicted    N    O    T
Actual
N          141   86   63
O           76  181   86
T           53   68  246
Logistic Regression for Preferences label classification for Spacy embedding: 0.568
              precision    recall  f1-score   support

           N       0.52      0.49      0.50       290
           O       0.54      0.53      0.53       343
           T       0.62      0.67      0.65       367

    accuracy                           0.57      1000
   macro avg       0.56      0.56      0.56      1000
weighted avg       0.57      0.57      0.57      1000
```

```
Logistic Regression Confusion Matrix for Comparison label classification for Spacy embedding:
Predicted   0    1    2
Actual
0          23   70   55
1          18  324  119
2           9  119  263
Logistic Regression for Comparison label classification for Spacy embedding: 0.61
              precision    recall  f1-score   support

           0       0.46      0.16      0.23       148
           1       0.63      0.70      0.67       461
           2       0.60      0.67      0.64       391

    accuracy                           0.61      1000
   macro avg       0.56      0.51      0.51      1000
weighted avg       0.59      0.61      0.59      1000
```

```
Logistic Regression Confusion Matrix for Preferences label classification for TF-IDF embedding:
Predicted    N    O    T
Actual
N          189   69   32
O           41  267   35
T           31   39  297
Logistic Regression for Preferences label classification for TF-IDF embedding: 0.753
              precision    recall  f1-score   support

           N       0.72      0.65      0.69       290
           O       0.71      0.78      0.74       343
           T       0.82      0.81      0.81       367

    accuracy                           0.75      1000
   macro avg       0.75      0.75      0.75      1000
weighted avg       0.75      0.75      0.75      1000
```

```
Logistic Regression Confusion Matrix for Comparison label classification for TF-IDF embedding:
Predicted   0    1    2
Actual
0          49   47   52
1           9  379   73
2           5   63  323
Logistic Regression for Comparison label classification for TF-IDF embedding: 0.751
              precision    recall  f1-score   support

           0       0.78      0.33      0.46       148
           1       0.78      0.82      0.80       461
           2       0.72      0.83      0.77       391

    accuracy                           0.75      1000
   macro avg       0.76      0.66      0.68      1000
weighted avg       0.75      0.75      0.74      1000
```

Proposed Model:

The proposed model comprises of implementation of a context-driven sentence-embedding scheme that looks leverages the sequential structure of the sentence. In other words, the following improvements were made on the baseline model:

1. Sfew extra preprocessing steps were added
2. Sequence-based word embeddings techniques were used to geenrate sentence embeddings.

I have tried implementing an LSTM based text-classifier model using GloVe word-embeddings in the Keras embedding layer, but LSTM based classifier's results were much poorer when compared to the baseline model. Hence, I stuck with Random Forest classifier which had a better accuracy overall.

The following steps were implemented in baseline model:

- **Data preprocessing:**
  For our baseline model we generated a `cleaned_sentence` column in our dataset. This sentence had no punctuations, whitespaces and capital letters. We perform –
  1. Stop-words removal
  2. Lemmatization of words from cleaned sentence to obtain root word.
- **Context based sentence-embedding:**
  Several embedding techniques were tried in this step. I tried to train my own word-embedding model as a part of LSTM frame-work, and also tried to incorporate GloVe based word-embeddings for the LSTM embedding layer. But the results could not beat the baseline model in terms of model accuracy.
  In the proposed solution, **Google's Universal Sentence Encoder** was implemented to generate context-based sentence embeddings. This encoder exploits the sequential nature of the data to create the embeddings.
- **Model selection:**
  The models selected for the classification tasks were:
  1. **XGBoost -** XGBoost is a boosting based ensemble decision tree classifier. The following hyper-parameters were set:

n_estimators = 200 – Number of weak estimators for our ensemble model

- **Model training and implementation:**
  1. There were 2 models run in total – 1 for each label (‘Preferred’/ ‘Comparison’)
  2. All models were evaluated on the above-described evaluation metrics.
- **Proposed model results w.r.t. baseline results:**

  The best accuracy metrics have been high-lighted in bolded

  1. Preferred labels – classification:

Accuracy	Random Forest	**Logistic Regression**	XG Boost
spaCY	0.659	0.568	-
TF-IDF	0.716	**0.753**	**-**
Google U.S.E.	-	**-**	0.726

  2. Comparison labels – classification:

Accuracy	Random Forest	Logistic Regression	**XG Boost**
spaCY	0.707	0.61	-
TF-IDF	0.781	0.751	-
Google U.S.E.	**-**	-	**0.8**

From the results, we can see that the proposed model (Google U.S.E + XGBoost) performs slightly better than the baseline model for Comparison label classification task. However, for Preferred label classification task, (TF-IDF + Logistic Regression) is a better classifier in terms of accuracy.

```
XGBoost Confusion Matrix for Preferred label classification for Universal Sentence Encoder embedding:
Predicted    1    2    3
Actual
1          289   47   31
2           49  241   53
3           47   47  196
XGBoost for Preferred label classification for Universal Sentence Encoder embedding: 0.726
              precision    recall  f1-score   support

           1       0.75      0.79      0.77       367
           2       0.72      0.70      0.71       343
           3       0.70      0.68      0.69       290

    accuracy                           0.73      1000
   macro avg       0.72      0.72      0.72      1000
weighted avg       0.73      0.73      0.73      1000
```

```
XGBoost Confusion Matrix for Comparison label classification for Universal Sentence Encoder embedding:
Predicted   0    1    2
Actual
0          77   50   21
1          13  403   45
2           7   64  320
XGBoost for Comparison label classification for Universal Sentence Encoder embedding: 0.8
              precision    recall  f1-score   support

           0       0.79      0.52      0.63       148
           1       0.78      0.87      0.82       461
           2       0.83      0.82      0.82       391

    accuracy                           0.80      1000
   macro avg       0.80      0.74      0.76      1000
weighted avg       0.80      0.80      0.80      1000
```

Conclusion:

Hence the classification task has been successfully performed. Some take-away points here are:

1. Sometimes the word-to-vec models that exploit local data structure and corpus perform better over generic pre-trained models. TF-IDF's performance illustrates this. Such an approach is useful when classification task is on a very specific domain or a small dataset.
2. Sometimes simple classification algorithms work well compared to sequence models. LSTM's 40 percent accuracy illustrates this.
3. Ensemble models have a generally higher classification algorithms because they train multiple weak models that specifically focus on getting the wrong classifications right. Both random forest and XGBoost demonstrate this.