Natural Language Understanding: Project 2 - Story Cloze Task

Thomas Brunschwiler **Dario Kneubühler** **Mauro Luzzatto**

1 Introduction

The target of this natural language understanding project was to successfully accomplish the story cloze task. This task was developed to measure advanced commonsense understanding within everyday written stories [1]. To solve the task, one has to determine the correct ending sentence out of two available options, given the four initial context sentences of the story. Two training sets were provided; one consisted of the context sentences with the correct ending only (train_single; train_stories.csv) and the second included also an additional incorrect ending (train_double; cloze_test_spring2016-test.csv). Further, an evaluation set (cloze_test_val__spring2016_cloze_test_ALL_val.csv) with context sentences and labeled endings was provided to determine the accuracy of the classifier. Finally, a test set (test_nlu18_utf-8.csv) was provided to participate in the competition by classifying unlabeled sentences.
In the LSDSem 2017 competition, in which different teams competed to solve the story cloze task, validation accuracies between 0.595 and 0.752 were reached [2]. The major difficulty of this task is to extract semantic information from the story context and use it to determine the correct ending. Additionally, the train_double set is relatively small with 1'871 samples compared to the train_single set with 88'161 samples. Thus, the smaller training set can be used to train a shallow classifier, while the large training set is required to train a deep neural network.

2 Methodology and Model

We trained a binary classifier based on logistic regression to identify true and false endings. The probability to be a true sentence is identified for both given endings of the story. Then, the sentence with the higher probability value is defined as the true sentence and the other as the false sentence. Schwartz et al. showed that this approach is promising [3]. We extended this idea through feature engineering by an universal sentence encoding, which is applied to all the sentences in advance. The sentence embedding is calculated using the universal sentence encoder module from TensorFlow Hub [4]. The module converts text into high dimensional vectors using a deep averaging network (DAN). The features used for the logistic regression are derived by three main methods, which are listed bellow and visualized in Figure 1.

A Sentence Embedding: Sentence embedding of last sentence
The universal sentence encoder is used to embed the last sentences of the story into a 512 dimensional vector, representing semantic content. This embedding vector is used to derive the following feature:

   1. Embedding Vector Feature
   The 512 elements of the vector.

B Word Embedding: Average word embedding distance between context embedding and last sentence embedding
The average value of the word embeddings in the story context (sentence1 to sentence4), as well as of the last sentence was computed to derive a semantic sentence representation. The pre-trained word2vec function [5] was used to create the word embedding. Words that are not in the dictionary are skipped and do not influence the values. This embedding vectors are used to derive the following feature:

1. Difference of Average Word Embedding Feature
   The averaged word embedding of the story ending is subtracted from the context word embedding, which results in a vector with 128 elements.

C Sentence Embedding Prediction: Similarity of a generated last sentence to the given sentence
A generative recurrent neural network (RNN) was trained on the story sentences to later predict the fifth sentense based on the initial story (sentence1 to sentence4). All sentences are represented through embeddings by the universal sentence encoder. The derived features are based on different distance measures between predicted and given final sentence:

1. Variance Feature
   The variance of the component wise difference between the predicted and the given sentence is calculated.
2. Vector Subtraction Feature
   The difference of the vector components of the predicted and given sentence are calculated, resulting in 512 values.
3. Manhattan Distance Feature
   The L1 distance of the predicted and given sentence embedding vector is calculated, resulting in one value.
4. Euclidian Distance Feature
   The L2 distance of the predicted and given sentence embedding vector is calculated, resulting in one value.
5. Cosine Distance Feature
   This feature represents the angular distance between the predicted sentence embedding and the last sentence embedding.

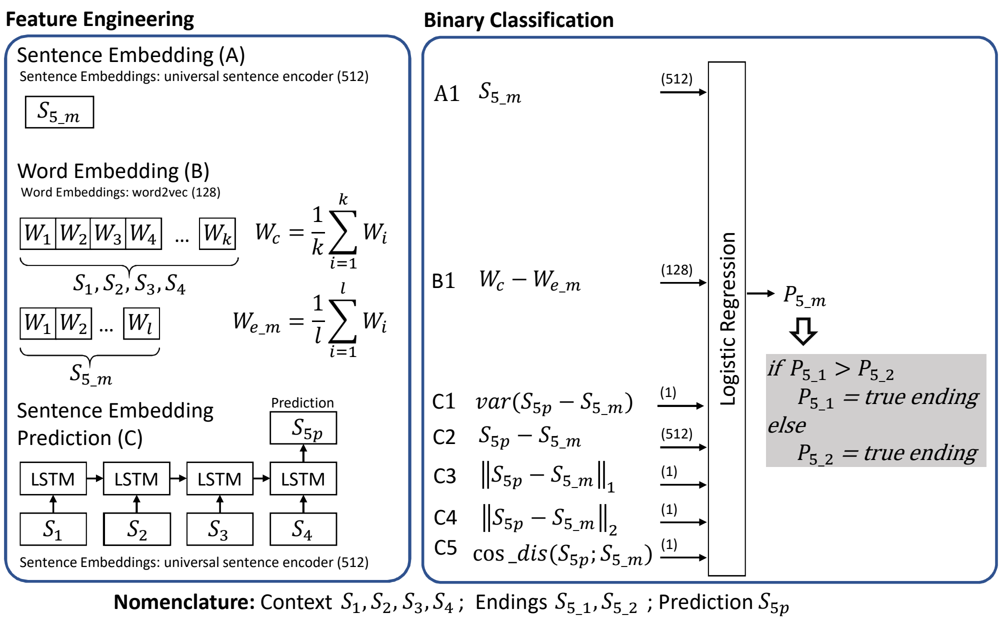
Feature Engineering
Binary Classification
Sentence Embedding (A)
Sentence Embeddings: universal sentence encoder (512)
S_{5_m}
Word Embedding (B)
Word Embeddings: word2vec (128)
W_1 W_2 W_3 W_4 ... W_k
$W_c = \frac{1}{k}\sum_{i=1}^{k} W_i$
S_1, S_2, S_3, S_4
W_1 W_2 ... W_l
$W_{e\_m} = \frac{1}{l}\sum_{i=1}^{l} W_i$
S_{5_m}
Sentence Embedding Prediction (C)
Prediction
S_{5p}
LSTM
S_1 S_2 S_3 S_4
Sentence Embeddings: universal sentence encoder (512)
A1 S_{5_m} (512)
B1 $W_c - W_{e\_m}$ (128)
C1 $var(S_{5p} - S_{5\_m})$ (1)
C2 $S_{5p} - S_{5\_m}$ (512)
C3 $\|S_{5p} - S_{5\_m}\|_1$ (1)
C4 $\|S_{5p} - S_{5\_m}\|_2$ (1)
C5 $\cos\_dis(S_{5p}; S_{5\_m})$ (1)
Logistic Regression
P_{5_m}
$if\ P_{5\_1} > P_{5\_2}$
$P_{5\_1} = true\ ending$
$else$
$P_{5\_2} = true\ ending$
Nomenclature: Context S_1, S_2, S_3, S_4; Endings S_{5_1}, S_{5_2}; Prediction S_{5p}

Figure 1: Model Description: Feature Engineering and Binary Classification

3 Training

To derive the Sentence Embedding Prediction features a generative model based on recurrent neural network architecture with four LSTM cells (one cell for each context sentence) was developed and trained. The network hyperparameters were optimized for performance by the train_single set which contained overall 88'161 samples. 10% of the training data was used to monitor the training progress. In general, an RNN architecture with a small number of variables was chosen to prevent overfitting. The training batch size was set to 32, the epochs to 80, and the learning rate to 0.0001 with a decay step every 40'000 samples. The model loss was defined by the cosine distance between the given true sentence embedding and the predicted last sentence embedding. The ADAM optimizer with a gradient clip of 10 was used to optimize the model weights. Through grid-search, the best performing architecture consisting of a single RNN layer with an LSTM cell size of 64 was identified. It resulted in a loss of 0.378 and a model with 181'000 hyperparameters. Additionally, we tried to reduce the mean squared error instead of the cosine distance. However, this resulted in a similar result with a much larger training time. Furthermore, we also tried to reduce the embedding size of the sentences of 512 digits with PCA. However, this resulted in a worse performance and therefore, that approach was not further investigated.

For the final classification, a logistic regression classifier (penalty L2-norm) was trained based on the derived features extracted to identify the true and wrong ending between two given sentences and the story context. The train_double set was used to train the model and validation set is used to determine the model accuracy of. Grid-search was applied to tune the regularization strength.

4 Experiments and Results

Several experiments with different features and combinations thereof were conducted to determine the feature relevance and the most accurate model. The resulting validation accuracies as reported in table 1 were performed on the validation set.

Identifier	Model features	Validation accuracy
A1	Embedding of last sentence	0.701
B1	Word embedding	0.637
C1	Variance of the last sentence embedding	0.594
C2	Subtraction of the predicted embedding to the last sentence embedding	0.696
C3	Manhatten distance between the predicted embedding to the last sentence embedding	0.612
C4	Euclidean distance between the predicted embedding to the last sentence embedding	0.596
C5	Cosine distance between the predicted embedding to the last sentence embedding	0.600
B1, C1-C5	All features, without sentence embedding feature	0.668
A1, B1	Sentence embedding and word embedding	0.707
A1, B1, C1-C5	All features	0.709
A1, B1, C2	Cosine distance from RNN, word embedding, sentence embedding	0.712

Table 1: Validation accuracy of the trained classifier based on different features for the story clozed task.

5 Conclusion

In this project the authors implemented a natural language processing model to tackle the story cloze task. The final model relies on a combination of features derived from word and sentence embedding, as well as from predicted sentence embedding from a generative RNN. The features are fed into a logistic regression model to perform the classification. The final implementation is able to reach a validation accuracy of 71% which is comparable to state-of-the art solutions [2]. The use of features derived from the universal sentence encoder from TensorFlow Hub applied on the last sentence itself, resulted in a high performance of 0.701%. Interestingly, this feature, derived without considering any of the context sentences results in a higher performance than the word embedding and generative features which included he context sentences. Combinations of the three main feature types resulted only in a marginal improvement of the performance and suggests a strong dependence between the various features. Looking into the future, the current implementation could certainly be improved. Bi-directional RNN architecture with more layers in combination with more data for training are expected to boost the accuracy even further.

References

[1] Nasrin Mostafazadeh et al. A corpus and cloze evaluation for deeper understanding commonsense stories. 2016.

[2] Nasrin Mostafazadeh et al. Lsdsem 2017 shared task: The story cloze test. 2017.

[3] Roy Schwartz et al. Story cloze task: Uw nlp system. 2017.

[4] Daniel Cer et al. Universal sentence encoder, 2018. https://www.tensorflow.org/hub/modules/google/universal-sentence-encoder/1.

[5] The TensorFlow Authors. word2vec - word embedding, 2018.