CSA 250 : Deep Learning Project III
Report

Abhishek Kumar (15648)

May 15, 2020

Project III : Problem Statement

Natural Language Entailment : Given two natural language sentences, a premise P and a hypothesis H, the textual entailment task – also known as natural language inference (NLI) – consists of determining whether the premise entails, contradicts, or is neutral with respect to the given hypothesis (MacCartney and Manning 2009).
Teaching machines how to read, reason, and answer questions over natural language questions is a long-standing area of research; doing this well has been a very important mission of both the NLP and AI communities. In this project we are trying to solve the natural language inference using deep learning methods. But before using these texts as input for our deep-learning model we need to convert it into numbers or vectors. We are comparing the performance of two model :
(1) Tf-Idf feature method with logistic regression
(2) Deep learning model using LSTM or pre-trained model such as BERT, XLNet

Dataset

Stanford Natural Language Inference (SNLI) The SNLI corpus (version 1.0) is a collection of 570k human-written English sentence pairs manually labeled for balanced classification with the labels entailment, contradiction, and neutral, supporting the task of natural language inference (NLI), also known as recognizing textual entailment (RTE). This dataset is provided by the Stanford NLP community.

Dataset Preprocessing
We know that input to deep learning models are the tensors and these tensors are made up of numbers but an english sentence contains word where each word is a collection of alphabets i.e., we cannot directly feed these texts to our deep learning model for training. Hence, these corpus needs pre-processing before it can be used. We followed following steps to preprocess the corpus:

1. **Removal of STOPWORDS** : removed words which are used frequently in a sentence but give little information about the sentence. Some examples are : 'is', 'he', 'the', 'are' etc.

2. **Removal of punctuation and special characters** : similar to stop-words, these words hold little or no information.

3. **Removal of white-spaces and alphanumeric characters :** The idea behind removing these words is similar to step 1 and 2. It is very crucial to remove these stopwords and special characters because they occur very frequent in a sentence and it can make the model's learning very difficult/skewed. Model may give more importance to these because there frequencies is very high.

4. **Tokenization** : Converted the sentences into the tokens because we want to look the sentences at the word level. Also the idea behind this is that sentences are nothing but collection of words

and if we are able to find good representation of words then we can have a good representation for sentences. We can go to character levels also.

TF-IDF

TF : Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length as a way of normalization.
IDF : Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However it is known that certain terms, such as "is", "of", and "that", may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones.

TF-IDF = Product of TF and IDF i.e., TF x IDF

For TF-IDF vectorization : Used sklearn module TfidfVectorizer.
Experimented with different sizes of max features or the size of vocabulary to find the best result.
To use TF-IDF, Combined both premise and hypothesis to apply tf-idf and after that it is passed through a logistic regression.

Results :

1. Max-feature : 100, Accuracy : 35.26
2. Max-Feature : 1500, Accuracy : 36.62
3. Max-Feature : 5000, Accuracy : 57.81

These low accuracy scores are expected because tf-idf only finds the most important term/word from a sentence and it is not suitable for classification.

Deep-Learning Framework

BiLSTM Model

For the deep learning framework, we chose the bilstm layer as the basic building block with attention.
For many NLP tasks an Encoder and Decoder structure is followed where Encoder : encodes the entire sentence/paragraph into a hidden representation and Decoder: decodes the hidden representation to required representation so that it match the labels.
For example : Machine Translation tasks (e.g., English to French) :
Encoder : Encodes the English words into hidden representation such that the whole sentence can be represented by it.
Decoder : Decoder takes the hidden representation as an input and decodes this summarized rep. to required French language.

We have tried to emulate the same architecture for the Natural language inference tasks.

Since, BiLSTM takes a lot of time to train, we need to be very careful of the dimension of the embedding and hidden layer. I tried to experiment as much to get the best result. We hyper-tuned the model using changes in learning rate, dropout ratios etc.

Results

: **Final Model :**
Validation accuracy(20 % training) : 79.923, Test Accuracy : 79.255.
Label-wise accuracy :
Entailment : Accuracy : 84.798
Contradiction : Accuracy : 79.055
Neutral : Accuracy : 73.656

Table 1: Confusion-Matrix

-	Entailment (pred)	Contradiction (pred)	Neutral (pred)	Total
Entailment (gold)	2856	163	349	3368
Contradiction (gold)	255	2559	423	3237
Neutral (gold)	429	419	2371	3219
Total	3540	3141	3143	98241