Recognizing Textual Entailment

Kirils Mensikovs, May 08, 2017

Abstract

Textual Entailments (TE) in Natural Language Processing (NLP) is a directional relation between text fragments. The relation holds whenever the truth of one text fragment follows from another text. TE is designed to focus efforts on general textual inference capabilities, but without constraining participants to use a specific representations or reasoning approach.

1 Introduction

Search engines, chat bots and assistants such as Alexa or Google Home have been using Natural Language Processing (NLP) to better understand the meaning of word and context of the situation to make the intelligent answer to the people.

The characteristic of natural language is that are many different ways to state what you want to say. Several meanings can be contained in a single text and that the same meaning can be expressed by different text. For example can we say sentences “Clearly, California can - and must - do better.” and “California cannot do any better.” are spoken about the same or not? Syntactic structure is quite similar. But the meaning of the second sentence contradicts with the first.

The task we try to solve is to understand words meaning. Textual Entailments (TE) in NLP is a directional relationship between pairs of text fragments, one named *text* (T) and the other named *hypothesis* (H). We say that T entails H if the meaning of H can be inferred from the meaning of T, as would typically be interpreted by people. Usually, hypothesis is a short statement, and text is a longer span of text. Table 1 shows a sample text, hypotheses and label. The label of each entailment pair is determined by multiple human annotators. Each entailment pair is labeled as entailment, natural and contradiction.

It has become feasible to train deep learning (DL) based inference models. Carefully designing sequential inference model based on chain LSTMs can show good performance (Chen et al., 2017). TE is a generalization of many tasks in NLP. If you have a system that is good at recognizing TE, it should be easier to build good systems for Information Retrieval (IR), Question Answering (QA) (Tang et al., 2016), Paraphrase Recognition (PR), Information Extraction (IE) and summarization. Typically entailment is used as part of large system.

We focus on the MultiGenre NLI corpus data, examples from which are shown in table 1. This data set contains pairs consisting of short text fol-lowed by a sentence hypothesis. The goal is to say whether the hypothesis follows from the text and general background knowledge, according to the intuitions of an intelligent human reader. That is, the standard is not whether the hypothesis is logically entailed, but whether it can reasonably be inferred.

2 Preparing the text data

We will use the Multi-Genre NLI Interface (MultiNLI) corpus to create the model.
https://www.nyu.edu/projects/bowman/multinli/

Text	Hypothesis	Label
Well it's been very interesting	It has been very intriguing.	entailment
Drawing a blank?	You can be drawing a blank.	entailment
There was nothing like that emotion now.	There are few emotions that come close.	neutral
Clearly, California can - and must - do better.	California cannot do any better.	contradiction
Gays and lesbians.	Heterosexuals.	contradiction

Table 1: Example from MultiNLP

MultiNLP is a crowd-sourced collection of 433k sentences pairs annotated with textual entailment information. The corpus contains data from ten distinct genres of written and spoken English text (Adina Williams et al., 2016). Dataset contains following groups: *entailment* (130,899), *neutrals* (130,900) and *contradiction* (130,903).

The corpus is available in two formats, tab separated text and JSON Lines (jsonl). We will iterate over the JSON lines, and format them into a list of samples.

The text annotations of the MultiNLP dataset have the following fields:

- gold_label: The label to be used for classification. In examples which are rejected during the validation process, the value of this field will be '-'. These samples is ignored in evaluations.
- sentence1: The premise sentence for the pair.
- sentence2: The hypothesis sentence for the pair, composed by a crowd worker as a companion sentence for sentence1.
- sentence{1,2}-parse: each sentence as parsed by the Stanford PCFG Parser (Klein and Manning, 2003).
- sentence{1,2}_binary_parse: The above parses in unlabeled binary-branching format.
- promptID: A unique identifier for each premise sentence.
- pairID: A unique identifier for each example.
- genre: The genre of the source text from which sentence1 was drawn.
- label[1]: The label assigned during the creation of the sentence pair. In rare cases this may be different from gold_label, if a consensus of annotators during the validation phase chose a different label.
- label[2..5]: The four labels assigned by individual annotators to each development and test example during validation.

We take following fields for training data: *sentence1*, *sentence2* and *gold_label*. Those are main features that allow us to train model. We also can try to use sentence{1,2}-parse features. But we thought that Part Of Speech feature doesn't benefit for our task and make model complex. All other features are informative and can be used during training analyses. For example each corpus corresponds to the domain from which its examples were drawn. Performance of systems varies across tasks, indicating significant qualitative differences between the examples in each.

Premise sentences in MultiNLI tend to be longer than their hypothesis sentences. The longest premise is 401 words and the longest hypothesis 70 words.

3 System

The system we specify here is designed to incorporate existing NLP resources in a uniform way, and to allow systematic development of both straightforward and complex TE systems.

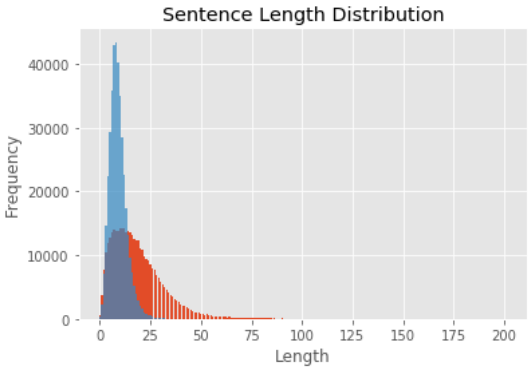

Figure 1: Sentence length distribution (text is red, hypothesis blues).

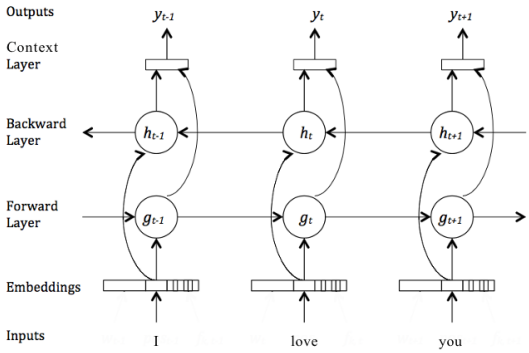

Figure 2: Word and context representation

Given two text fragments, one named *text* (t) and the other named *hypothesis* (h), respectively. The task consists in recognizing whether the hypothesis can be inferred from the text. TE has a three-class balanced classification problem over sentence pairs: *entailment*, *neural* and *contradiction*.

We build the bilateral multi-perspective matching (BiMPM) model (Wang et al., 2017). The model matches sentences and hypothesis in two directions. From the sentence follow the hypothesis and from the hypothesis follow the sentence. In each direction, the model matches the two sentences from multiple perspectives. The model contains five layers.

3.1 Word Representation

To make machine understand words, we will convert all text samples in the dataset into sequences of word indices. A word index simply be an integer ID for the word. We will only consider the top 20,000 most commonly occurring words in the dataset, and we will truncate the sequence to a maximum length of 100 words.

Prepare and embedding matrix which will contain at index *i* the embedding vector for the word of the index *i* in our word index.

We be using GloVe embedding's (Pennington et al., 2014). Embedding technique based on factorizing a matrix of word co-occurrence statistics. Specifically, we use the 300-dimensional GloVe embedding of 2.2m words computed on a common crawl. Example at Figure 2.

By parsing the data of pre-trained embeddings. We compute an index mapping words to known embeddings. We got *2,196,016* words index.

We format our text samples into tensors that can be fed into a neural network. To do this, we vectorizing text and turning text into sequences (list of word indexes, where the word of rank *i* in the dataset (starting at 1) ha index *i*). At the end of transformation we get a list *num_samples* of sequences into a 2D Numpy array of shape (*num_samples*, *num_timesteps*). Sequences that are shorter than *num_timesteps* are padded with *value* at the end. Sequences longer than *num_timesteps* are truncated so that it fits the desired length. During training, we do not update the pre-trained word embeddings. Our data set contains 392,702 samples with vocabulary of 77,556 unique words.

We found that our embedding contains 59,383 words from or data set and 18,173 words are absent. For the out-of-vocabulary (OOV) words, we initialize the word embedding randomly. The max sentence word sequence is 382.

3.2 Context Representation

To fetch contextual information form embedding's we utilize a bi-directional LSTM to encode contextual embeddings for each time-step. Figure 2.

3.3 Matching Layer

Next layer is the core layer it compare each contextual embedding of one sentence against all-contextual embedding of the other sentence.

We define a multi-persective cosine matching function *Fm* to compare two vectors:

$$m = Fm\ (v1, v2; W)$$

where *v1* and *v2* are two d-dimensional vector, W *(belongs to* R^{l*d}*)* is a trainable parameter, *l* is the number of perspectives, ant returned value m is a *l*-dimensional vector *m = [m1, ..., mk,...ml]*. Each element *mk* is a matching value from the k-the perspective, ant it is calculated by the cosine similarity between two weighted vectors

$$mk = cosine(Wk * v1, Wk * v2)$$

where * is the element-wise multiplication, and *Wk* is the *k*-th row of *W*, which controls the *k*-th perspective and assigns different weights to different dimensions of the *d*-dimensional space.

Based on *Fm*, we define two matching strategies to compare each time-step of one sentence against all time-steps of the other sentence.

Each strategy first at the beginning calculates the cosine similarities between each forward and backward contextual embedding.

First, strategy retains only the maximum value of each dimension.

Second, strategy retains only the mean value of each dimension.

We apply all these two matching strategies to each time-step of the sentence T, and concatenate the generated four vectors as the matching vector for each time-step of T.

3.4 Aggregation Layer

Next step we aggregate the two sequence of matching vectors into a fix-length matching vector. We use bi-directional LSTM, and apply it to the two sequences of matching vectors individually. Then we construct the fix-length matching vector by concatenating vectors from the last time-step.

3.5 Prediction Layer

The last layer is evaluation of entailments probabilities. We employ a two layer feed-forward neural network to consume the fix-length matching vector, and apply the *softmax* function in the output layer. The whole model is shown in Figure 3.

3.6 Metric

Accuracy for each instance is defined as the proportion of the predicted *correct* labels to the total number (predicted and actual) of labels for that instance. Overall accuracy is the average across all instances. Accuracy is a common metric for text entailments. It takes into account both true positives and true negative with equal weight.

This metric was used when evaluating the textual entailments because false negatives and false positives both don't make any sense for textual entailment task.

4 Evaluation

The RepEval 2017 Shared task feature two evaluations, a standard in-domain evaluation in which the training and test data are drawn from the same sources, and a cross-domain evaluation in which the training and test data differ substantially. This cross-domain evaluation will test the ability of system to learn representations of sentence meaning that capture broadly useful features.

The task dataset Multi-Genre NLI corpus consist of 393k training examples drawn from five genres of text, and 40k tests and development examples drawn from those same five genres, as well as five more. The labels entailment, neutral, and contradiction, in roughly equal proportions.

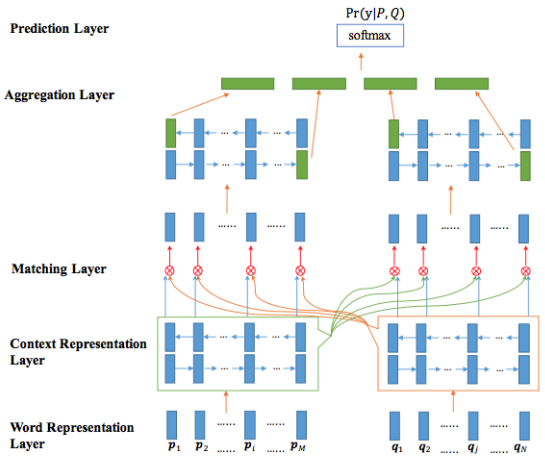

Figure 3: Model.

Model	Matched Test Acc.	Mismatched Test Acc.
Most Frequent Class	36.5	35.6
CBOW	66.2	64.6
BiLSTM	**67.5**	**67.1**

Table 2: Example from MultiNLP

The study found humans to be in agreement on the dataset 95.25% of the time (Boss and Markert, 2005). In principle, then, the upper bound for machine performance is quite high. In practice, however, the TE task is exceedingly difficult for computers. The RepEval 2017 Shared task trained three neural network models; evaluate them and provided base lines (Table 2). The baseline for all experiments is random guessing, which always attains 50% accuracy. We use accuracy as evaluation metric.

Entailments will certainly depend on linguistic knowledge, and may also depend on word knowledge.

5 Experiments

The bilateral multi-perspective matching is the state-of-the art algorithm for the textual entailments. The algorithm outputs an assigned probability for each class. This can be used to reduce the number of false positives using threshold.

The following parameters can be tuned to optimize the classifier:

- Number of epochs
- Batch size
- Learning rate
- Weight decay
- Momentum

During training, both the training and the validation sets are loaded into the RAM. Dataset is shuffled and batches are loaded into GPU memory for processing.

The goal of the training is to minimize the total loss of the model. (but for evaluating model performance, we only look at the loss of the main output). The model is trained in epochs, where the model sees all the input data at least once. During each epoch, batches of size 128 are loaded into the model and evaluated, calculating a batch loss for each; the gradients from the batch are back propagated into the previous layers to improve them.

We set size 100 for all LSTM layers. Apply dropout to every layers and set dropout ration as 0.1. To train the model, we minimize the cross entropy of training set, and use the ADAM (Kingma and Ba, 2014) optimizer to update parameters. We set the learning rate as 0.001. The pre-trained word embeddings was not updated during the training.

The training was done on AWS p2.xlarge instance. For 10 epochs, it took about 23 hour 30 minutes to train the model.

The state of the art bilateral multi-perspective matching achieved a good accuracy, around 84%. On our dataset algorithm showed 79% on matched and 66% on mismatched dataset.

6 Conclusion

We have trained BiMPM model and applied it to large Multi-Genre NLI corpus. We shown that text entailment with decent quality is possible with the architecture we have used.

Our text-entailments algorithm shows 79% accuracy.

Even thought the results are promising. One of the next steps of this project is to improve the accuracy for mismatched text. This can be improved by using part of speech (POS) tagging. There we can extend our word embedding's vector with additional POS embedding's vector.

References

Quian Chen, Zhenhua Ling, Hui Jiang, Xiaodan Zhu, Si Wei and Diana Inkpen. 2017. *Enhanced LSTM for Natural Language Interface*. https:// arxiv.org/pdf/1609.06038.pdf

Ming Tan, Cicero dos Santos, Bing Xiang and Bowen Zhou. 2016. *LSTM-base Deep Learning Models for non-factoid answer selection.* https://arxiv.org/pdf/1511.04108.pdf

Adina Williams, Nikita Nangia and Samuel R. Bowman. 2016. *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference*. https://www.nyu.edu/projects/bowman/multinli/paper.pdf

Jeffrey Pennington, Richard Socher and Christopher D. Manning. 2016. *GloVe: Global Vectors for Word Representation*. https://nlp.stanford.edu/pubs/glove.pdf

Zhiguo Wang, Wael Hamza and Radu Florian. 2017. *Bilateral Multi-Perspective Matching for Natural* Language Sentences. https://arxiv.org/abs/1702.03814

Kai Zhao, Liang Huang and Mingbo Ma. 2017. *Textual Entailment with Structured Attentions and Composition*. https://arxiv.org/pdf/1701.01126.pdf