Entailment in language pragmatics

Alessandro Ballerini

Università degli Studi di Milano

Abstract. In this paper I'll try to explore the problem of generating textual entailments, starting from a text input. My approach consists in the use of recurrent neural networks in an encoder/decoder pattern and the training of network using a supervised learning approach.

1 Introduction

1.1 Definition

Entailments are directional relations between text fragments, where the truth of one requires the truth of the other. The objective of this project was to build a model able to, given an input utterance, generate a meaningful and syntactically correct sentence that is related in some way to the input one.

Text: Two teams are competing in a football match. Hypothesis 1: Two groups of people are playing football. Hypothesis 2: People are running on grass field.

1.2 Language model

Language models are used to determine the probability of a given sequence of words occurring in a sentence.

They may be used to calculate the probability of the next word given the sequence of previous words.

$$P(w_n|w_1, w_2, ...w_{n-1})$$

Statistical Models vs Neural Models When dealing with NLP and language models, we can distinguish two different approaches to the problem: statistic models or neural network-based ones. For this project I choose to follow the neural network-based approach for a series of reasons:

First, neural networks "naturally" work with dense representations of tokens. The input, expressed in some form of encoding (e.g. one-hot encoding, indexes), is automatically embedded through the network more densely and less sparsely. As we will see, the concept of encoding input into a vector space can be applied also to sequences, and this is at the core of the model developed here.

An important application of this principle is word embedding, a technique that allows our model to work with words meanings and words relationships based on their meaning. Thanks to word embedding our model will not deal with tokens but with their representation in an n-dimensional vector space in which the reciprocal position of each word represents "nearness" on a semantic level. There exist other options to achieve word embeddings instead of the use of a neural network (such as co-occurrence, pointwise mutual information, context-based models), but the latter has the advantage to allow to "encode" the tokens into an arbitrary small and dense n-dimensional space, solving the dimensionality and sparsity problem that may affect other techniques.

Moreover, while is true that statistical language models based on N-grams may tend to perform better when N is small, as the value of N increases above 3 or 4 they tend to be more difficult to train, which means they require bigger data sets and displays no significant improvement as opposed to neural language models. This is since N-gram language models are usually based on the counting of the occurrences of sequences of words, and as N (length of the sequences) grows the amount of data required to encounter a reasonable number of sequences increases. This means that N-gram-based models tend to be less reliable when dealing with longer sequences.

One of the cons in the adoption of neural networks is that deep learning techniques make opaque the understanding of what is happening during the computation, as deep models are seen as a black box from an outside point of view and are harder to interpret. This, however, is not alone a good enough reason not to use neural networks.

1.3 Recurrent neural networks

An issue we encounter when we apply neural networks to the task of natural language modeling is that "traditional" feedforward networks make predictions based only on the last element of a sequence, or on their immediately previous output. This is problematic when we deal with natural language since, more often than not, words relations carry on for the whole phrase (if not between sentences) and not only between adjacent words. When we approach the task of sequence learning, Recurrent Neural Networks are the most suited neural network type. RNN predictions depend both on the current data and on the previous prediction of the network itself, allowing to formulate longer meaningful sentences.

More advanced types of RNN such as LSTM and GRU bring further improvements. To avoid the exploding/vanishing gradient problem both LSTM and GRU implement techniques to select and preserve meaningful information for the next iteration.

LSTM introduces the concept of cell state. Cell state consists of an additional hidden vector that is used to store and forget information. Information of earlier states is better preserved through time from the vanishing/exploding gradient problems, allowing the model to learn longer sequences.

GRU are a "simpler" version of the LSTM: while the latter contains three gates on each cell to perform the operations of storing (input gate), forgetting (forget gate) and output (output gate), a GRU cell contains only two gates called Update gate and Reset gate. The Update gate determines the amount of information that needs to pass along the next state; the Reset state decides the amount of the past information that should be forgotten. It is worth noting that the GRU node does not have a cell memory as the LSTM (since there is no output gate).

1.4 Semantic and Syntax

Handling and evaluating the semantics of a text is less intuitive for a machine than handling its syntax. Syntax of a sentence can "easily" be evaluated through statistical models (by calculating how probable is for a certain word to follow the previous one(s)) or ruled-based ones, many techniques are developed to achieve this. Since we are dealing with two different objectives (semantic relevance and syntax correctness) different evaluation methods must be adopted.

2 Research question and methodology

As stated in the introduction, in this project I decided to opt for a neural network-based approach. The main reason resides in the ability of advanced types of RNN such as LSTM and GRU to remember longer sequences of words, and the possibility to encode those longer sentences into a vectorial space.

This way, by having a big enough dataset composed of pairs of entailed sentences, it should be possible to build a model able to generate appropriate comments to given input texts.

Encoding entire sentences also help when dealing with the syntax of generated texts, since if the encoded sentence is correctly phrased, in most cases, the decoded sentence will be too.

2.1 Encoder decoder model

What briefly introduced above represents the strategy I adopted to build the model. I utilized a known architecture made of an encoder and a decoder to pair texts with the corresponding hypothesis. This strategy is usually used when dealing with machine translation, question answering or dialogue generation, or more in general sequence learning.

In the current project, the encoder and the decoder both utilize GRU networks to learn the whole sentence. I opted for GRU instead of LSTM networks because the first tend to perform as well as the other when dealing with short sentences but are simpler and easier to train.

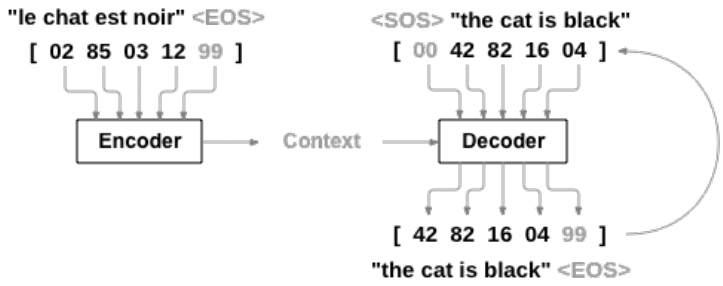

Fig. 1. Encoder/Decoder pattern, often used in machine translation.

3 Experimental results

3.1 Dataset

The dataset for this project consists of a Textual Entailment Graph in which each node represents a natural language sentence and each edge represents an entailment relation between two nodes. To increase the number of examples I decided to "relax" the definition of entailment by generating two examples for each edge. In total, more than 7400 pairs were used for the model training and evaluation in a supervised learning approach.

The dataset is focused on customer dissatisfaction and was constructed from a text collection of email feedback sent by customers of a railway company. It is composed of short sentences about a few topics and usually, their sentiment is negative about their subject. This provides both pros and cons to the development of the model: Having sentences focused on a few topics and always on negative aspects could limit the ability of the model to generate meaningful hypotheses on a broader field of subjects and with a different sentiment. But the limits of the datasets provide an opportunity to answer some useful questions that could help improve the model. This shows us what are the limits of such an approach and how to remedy those.

Interesting questions about the performance of the model, given the TEG dataset, could be: How much dependent is the model on the sentiment of the examples? How able is the model to generalize concepts and apply them to other subjects or contexts? Is the model able to generate new, meaningful hypotheses when the text is proposed with a different formulation?

Just answering those questions will result in useful information to expand the study on the subject.

3.2 Evaluation

The evaluation of the result cannot be performed exclusively by looking at an eventual "accuracy" or a "loss" value since we are dealing with more abstract subjects, like the meaning of a word or sentence or its underlying context. Since the core of our problem relies on the ability of the model to understand the underlined context and generate an appropriate "comment" our evaluation method must account for that.

Due to the scope of this project, I opted to manually evaluate the results of the model mainly for two reasons: By analyzing some samples of texts generated by our model, it was clear that it preserves the language syntax quite well. The syntax correctness appears to be a non-issue for the model developed. The second reason is that, concerning the semantics of the generated texts and their relations with the input utterance, since an entailment is not necessarily unique in its formulation and there is no restriction for its subject, no automated systems could have been able to judge the model for this particular aspect.

3.3 Results

During the training pairs of text/hypotheses have been sampled from the dataset and submitted to the model in total 75000 times. Every 7500 iteration the current loss value has been collected to evaluate the performance of the training.

To evaluate the performance of the trained model I examined by hand 100 examples taken from the training set and 100 examples taken from the test set. Each sample consisted of an input text, a paired hypothesis provided by the dataset, and the hypothesis generated by the model.

	Correct		Wrong		
	Total	Copy	Total	Semantic	Syntax
Train	68	13	32	10	23
Test	74	4	26	7	19

I annotated whether the hypothesis proposed by the model was correct, meaning it was syntactically well formulated and meaningful by itself and in association with the input text, or whether it was lacking in one of both syntax or meaning. I also annotated whether the formulated hypothesis was identical to the one provided by the dataset or the model made changes on its formulation while preserving, in some way, the meaning and correlation with the input text.

Input t: It could be improved by making the portions of the meals larger.
Example h: I wished the food would be more consistent in terms of quantity.
Model h: Food portions are small.

Input t: Trains could use some updating.
Example h: Trains need refurbishment.
Model h: Trains need refurbishment.

Input t: Food is not good.
Example h: Time for the food improved.
Model h: Food is food.

Lastly, I evaluated some completely original sentences to test the ability of the model to generalize both subject or topic, or whether it was able to interpret and adapt to the sentiment of the new text.

Input t: There is enough leg room.
Model h: The seats could do with more legroom.

Input t: Food is good enough.
Model h: Food is be.

Input t: There is not enough time.
Model h: food is average.

Input t: I would like a fruit.
Model h: Food food be with more choice.

The model performed well when dealing with texts and concepts already seen in the examples of the training set. The low number of syntactical errors show that the model is able to learn (encode) and apply correctly the syntax of the language of the examples, and the fact that the vast majority in both training and test samples is not a copy-paste of the dataset pairs indicates that it can understand similar contexts "behave" accordingly. The most obvious issues are related to the inability of the model to generalize between different concepts that are not present in the dataset. Also, changing the sentiment of a sentence which subject is included in the training set won't give the expected results.

An improvement could surely be provided by a larger and more "general purpose" dataset which would include more topics, scenarios, and relations between texts. But even in this case, I'm afraid that the current model architecture would still encounter limits due to the inability to generalize and apply the associations learned in one scenario to another.

A proposal of solution could be the application of masks to key elements of the sentences (for example masking the subjects, object, abjective, predicate

etc) in order to increment the number of examples and produce an effect similar to the data augmentation seen in fields such as image recognition.

4 Concluding remarks

For this project, I applied an encoder/decoder model using GRU recurrent neural network to solve the problem of entailment generation. The dataset provided more than 7400 examples of entailed pairs of sentences about a specific topic and sentiment.

The results showed that the model was able to correctly apply the syntax of the language used in the examples and correctly generate meaningful sentences. The model nevertheless reflects the limits of the dataset it is trained on: correct "answers" are given when the input sentence reflects and does not expand the topics, scenarios, and sentiment seen in the examples.

To conclude, the encoder/decoder model architecture applied to the entailment generation could be further explored with a more inclusive, larger dataset or by experimenting with data augmentation techniques such as word masking.

References

[Pol20] Adam Poliak. *A Survey on Recognizing Textual Entailment as an NLP Evaluation*. 2020. arXiv: 2010.03061 [cs.CL].