Long Short Term Memory Based Models for Negation Handling in Tutorial Dialogues

Dipesh Gautam, Nabin Maharjan, Rajendra Banjade, Lasang Jimba Tamang, Vasile Rus
Department of Computer Science and Institute for Intelligent Systems,
The University of Memphis, Memphis, TN, 38152
dgautam@memphis.edu

Abstract
Negation plays a significant role in spoken and written natural languages. Negation is used in language to deny something or to reverse the polarity or the sense of a statement. This paper presents a novel approach to automatically handling negation in tutorial dialogues using deep learning methods. In particular, we explored various Long Short Term Memory (LSTM) models to automatically detect negation focus, scope and cue in tutorial dialogues collected from experiments with actual students interacting with the state-of-the-art intelligent tutoring system, DeepTutor. The results obtained are promising.

Introduction
All human languages have negation as a key linguistic feature. Negation reverses the polarity of entire statements or of parts of statements. Givón (1993) categorized negation broadly into two classes: morphological and syntactic negation. The morphological negation follows a specific structure where a root word is modified by prefixes such as "non", "un" or suffixes such as "less". On the other hand, in syntactic negations, explicit negation cues are used to affect the meaning of a single word or a group of words.

Negations are marked by a set of cue words (or negation words) such as "no", "none", "not", "neither", "nor" and their variants(such as " 'nt"). Statements might be negated either explicitly with explicit cues, e.g. "no" or "not", or implicitly such as "avoid", "prevent", "prohibit" and so on. In the latter case, we have implicit negation. The negation cue can affect one or more parts of the statement in which it occurs - the affected parts are called the scope of negation. The constituent in the scope that is prominently negated is called the focus of the negation (Huddleston and Pullum 2002).

An example of negation is shown below. The negation clue is delimited by <<>>, within square brackets ([ ]) we show the scope of negation, and within curly brackets {} we show the negation focus.

[There are] <<no>>{[forces] acting upon the puck} because it is at rest.

Many studies showed that negation accounts for a significantly large part of both spoken and written human language. In one study, it is reported that negation occurs twice as often in speech as in writing (Tottie 1993). Some domain-specific corpus linguistics studies showed that negation occurs most frequently and represents a major portion of the information within such domain specific texts. For example, in a corpus of medical textual documents and biological scientific papers developed by Vincze and colleagues (2008), 10% of the sentences contain negation. A customer review corpus of movies, books and consumer products (Konstam-nova et al. 2012 ) includes 18.1% negated sentences. Also, an analysis of student utterances in dialogues collected from experiments with actual students interacting with the intelligent tutoring system DeepTutor (Rus et al. 2013) showed that 9.36% of student utterances included explicit negation.

In this work, we focus on handling negation in dialogues. Linguistics phenomena, such as ellipsis and pragmatics that are more prevalent in dialogues, make negation more complex to automatically analyze in this case as the meaning of a sentence relies on previous dialogue turns, i.e. previous context. The example below shows five possible answers with explicit (A1-A3) or implicit (A4 and A5) negation for the same tutor question (Q). It should be noted that in this work, we limit our analysis to explicit negation.

Q: Do a ping-pong ball and a bowling ball falling from the same height hit the ground with same force of impact?

A1: No
A2: They do not hit the ground with the same force of impact.
A3: The respective impacts of the two balls is never the same.
A4. The impact of ping-pong ball will be smaller.
A5. The impact of bowling ball will be greater.

Previous automated approaches to negation handling rely on the key observation that both written and spoken texts can be regarded as sequences of events, i.e. words, in which a word occurrence depends upon the long sequence of the previously occurred words. An analysis of such sequences of observations, e.g. to detect the negation scope, could be approached as a sequence labeling task in which each word is labeled as being part (or not) of the negation scope associated with a negation cue word. For instance, probabilistic approaches such as Hidden Markov Models (HMM) or Conditional Random Fields (CRF) have been used for negation handling (Prollochs, Feuerriegel and Neumann 2016; Banjade, Niraula and Rus 2016). In these approaches such as HMM, the model becomes complex when long term de-
pendency is taken into account. And in the case of CRF it also requires a set of hand crafted features (in addition to labeled data). Because of recent successes in training deep neural networks, and in particular recurrent neural networks (RNN) which are relevant to our task, and because of their ability to handle sequential data, it makes sense to approach the task of negation handling using a recurrent neural network approach. In this work, we will explore one version of RNNs that account for long term dependencies, namely, Long Short Term Memory (LSTM) RNNs (Hochreiter and Schmidhuber 1997). Neural networks can perform equally well or sometimes better than other complex sequence labeling models with comparatively large number of parameters or human constructed features. The only human involvement in a neural network based model is the labeling of data.

Specifically, we propose a deep learning approach to identify negation scope, negation focus and negation cues in tutorial dialogues. We explore various LSTM network architectures and compare their performance on a negation handling task using actual student utterance from several DeepTutor experiments. Furthermore, we explore a better input-output strategy to train and test the model in order to performance, i.e. better handle negation of unseen sequences of words which are future dialogue utterances in our case. While the negation scope and focus detection task is similar to prior work (Banjade, Niraula and Rus 2016), it should be noted that the primary motivation and novelty of this work is to explore deep learning methods. Our model detects negation cues as well, which is an additional contribution of our work compared to Banjade and colleagues.

In the next sections, we discuss related works, various models, experiments and results. The paper ends with conclusion and future work.

Related Works

Negation in natural language has been studied since ancient time (Wedin 1990). It is worth mentioning Horn’s work (1989) on the effect and influence of negation in languages. Horn described the construct, usage and cognitive processing of negation. It should be noted that in computational linguistics, negation handling methods were initially studied in the context of medical documents. Mutalik and colleagues (Mutalik, Deshpande and Nadkarni 2001) developed a tool called Negfinder to detect negated concepts in dictated medical documents. They hypothesized that the negated concepts could be detected using a lexer and parsers that are generally used to analyze programming languages. In another work, Rokach and colleagues (Rokach, Romano and Maimon 2008) proposed a pattern learning method for the automatic identification of the negative context in clinical narrative reports. They proposed several steps including corpus preparation, regular expression pattern learning, and training classifiers to predict negation.

Councill and colleagues (Councill, McDonald and Velikovich 2010) developed a Conditional Random Fields (CRFs) based state-of-the-art sentiment analysis system using features from an English dependency parser. In this work, they limit their study to explicit negation within a single sentence. They also developed a negation corpus of English product reviews obtained from the open web.

Banjade and colleagues (Banjade, Niraula and Rus 2016) proposed a sequence labeling model using CRFs to detect the negation scope and focus in dialogues. They also collected and manually annotated about 1,000 dialogues, i.e. student and tutoring system interactions collected from actual students using DeepTutor (Rus et al. 2013). Their work is the first to use CRFs to automatically detect the negation scope and focus both within the current utterance and previous utterance. That is, in their case the negation scope could include parts of the previous utterance (previous dialogue context) besides parts or the whole current student utterance. Though we performed our experiment on DeepTutor dialogues, our approach is different from theirs in two aspects: firstly, we proposed a deep learning method using LSTM and, secondly, the LSTM based models we proposed do not rely on labor intensive feature engineering.

LSTM-based approaches have been successfully used in various natural language processing tasks. Cho et al. (2014) proposed a recurrent neural network (RNN) model to encode sequence of input words (a phrase) into a fixed dimension vector. A variant of Cho’s model was explored by Sutskever and colleagues (Sutskever, Vinyals and Le 2014). They proposed a deep LSTM approach for sequence to sequence learning for machine translation. In their method, a multi-layer LSTM (encoder) mapped a sequence of inputs onto a fixed dimension vector and another multilayer LSTM (decoder) constructed the target sequence from the vector. They demonstrated that even with limited vocabulary, their model outperformed statistical machine translation models that use a large vocabulary to translate from English to French. They also found that reversing the input data improved the performance of the model. Since we model our negation handling task as sequence labeling problem, we explore an encoder-decoder model similar to Sutskever and colleagues.

Recurrent neural networks have been used in predicting sequences such as generating words by using models that predict next letters (Sutskever, Martens and Hinton 2011). Motivated by the success of recurrent neural network in sequence generation, Bi-directional LSTM (B-LSTM), a variant of LSTM, have been used in sequence tagging tasks (Huang, Xu and Yu 2015). Huang proposed various LSTM based models for sequence tagging. Their experiments on NLP benchmark sequence tagging tasks and using standardized data sets showed that hybrid models that combines B-LSTM with CRF produce state of the art accuracy on parts-of-speech tagging (POS), chunking, and named entity recognition (NER).

In many applications that need automated natural language understanding such as intelligent tutoring systems or question answering systems, challenges arise when dealing with negation such as accounting for context. Widdows and Peters (2003) proposed a model of vector negation in which the portion of the negation vector was subtracted from the vector representation of the document, which captures in some ways the full context of the document. RNNs, on which our approach relies, account for previous context using memory elements.
A critical ingredient in machine learning approaches to negation handling is collecting relevant data. There were several attempts to develop negation corpora in different domains. Vincze et al. (2008) developed the BioScope corpus which consists of biomedical texts. BioScope contains annotations at token level for negative and speculative keywords and their scope within a sentence. Konstantinova et al. (2012) developed an annotated corpus for negation and speculation in review texts. The corpus consisted of 400 documents of movie, book and consumer product reviews. They annotated negative and speculative keywords at token level and their scope (at sentence level). Morante and colleagues (Morante, Schrauwen and Daelemans 2011) published a comprehensive guideline for annotating negation cues and their scope. Blanco and Moldovan (2011) proposed a method to semantically represent negation using focus detection. They extracted and annotated negation focus on texts extracted from PropBank. The annotated data set was used in one of the shared tasks held in 2012 (*SEM 2012 ) which was about resolving the scope and focus of negation (Morante and Blanco 2012). These data sets do not fit our goal of handling negation in dialogue. Instead, we will use the data set prepared by Banjade and colleagues (Banjade, Niraula and Rus 2016) which contains annotated tutorial dialogues.

Model Description

We model the negation handling task as a sequence labeling task where a sequence of input tokens, i.e. words, need to be labeled within scope/focus and cue tags. Unlike previously proposed sequence labeling methods such as HMM or CRF, our LSTM-based approach is better suited for identifying the scope and focus of the negation cue in dialogues because the scope and focus could extend to previous dialogues utterances relative to the utterance where the negation cue is. In such cases, a model able to take into account long term dependencies should be preferred; that is, a long term memory based model such as LSTM.

More specifically, in our approach a sequence of tokens provided as input is processed and a sequence of tags, one for each input token, is produced as output. During training, input sequences are constructed from labeled conversation data by sliding a specific sized window over each utterance. Since the LSTM requires training sequences to be of equal length and the average size of our corpus is roughly 10 tokens per utterance, we set the window size to be of 5 tokens. In order to preserve the continuity of the context tokens that are in the original dialogue, we slide the window one step at a time so that each consecutive window overlaps to the previous window. Also, we slide the window over the labels exactly in the same way to obtain a sequence of labels for the corresponding tokens in the input utterance so that each input sequence is paired with the corresponding label sequence. For sequences shorter than the window size, a special padding symbol is appended both to the input and label sequences.

Figure 1 & 2 show our two different LSTM network architectures for negation handling. In the figures, the LSTM blocks from left to right show the unrolled network over time (t1 through tn). The output sequence is seen as the probability (given by softmax layer) of each label once the labels for all the tokens are processed (confirmed by time distributed dense layer). The labels (tags) corresponding to the input tokens are predicted by decoding the probability distribution given by the softmax layer.

In the following subsections, we describe in more detail the two different model architectures namely Sequence to Sequence Tagger and Tag Sequence Generator we designed for negation handling.

Figure 1: LSTM network of encoder-decoder model for predicting label of token sequence.

Figure 2: LSTM network of label sequence generator.

Sequence to Sequence Tagger

Similar to various sequence to sequence architectures proposed previously, our sequence to sequence architecture uses two different layers of LSTMs, one for encoding the input sequence into an embedding vector and the other to decode the embedding representation onto an output sequence. In general, such encoder-decoder architecture could be used to map an input sequence to an output sequence whose length may not be equal to the input. However, our task is to label each word with a negation tag, hence the output tag sequence is same length as input sequence. Instead of feeding the previous output back to its input, as in (Sutskever, Vinyals and Le 2014; Cho et al. 2014), we feed copies of the vector from the encoder to each of the LSTM units of decoders. Since each LSTM unit uses its state from the previous time step, simply feeding a copy of encoder output simplifies our model without loss of generality of the model. The Repeat layer (Figure 1) does the job of copying the final output of encoder layer to the decoder layer at every time step of decoding.
Table 1: Configurations of models.

<table>
<thead>
<tr>
<th>Model</th>
<th>Configuration</th>
</tr>
</thead>
</table>
| M1, M2 | • Single layer of both encoder and decoder with 150 LSTM units in each  
          • No dropouts  
          • M1 has training batch size = 100, M2 has batch size of 50 |
| M3, M4 | • M3: Single layer of both encoder and decoder with 150 LSTM units in each  
          • M4: Two layers with 150 LSTM units in each layer of both encoder and decoder  
          • Dropouts = 0.3 for each unit  
          • Training batch size = 50 |
| M5 | • Two layers with 150 LSTM units in each layer  
       • Dropouts = 0.3 for each unit  
       • Training batch size = 50 |

Note: M1, M2, M3 and M4 are Sequence to Sequence (encoder-decoder) model and M5 is Sequence Generator model.

Tag Sequence Generator

Recurrent neural networks were successfully used in language models to generate next letter or word based on long term context (Sutskever, Martens and Hinton 2011; Graves 2013). Unlike sequence to sequence architecture, where the input sequence is mapped to output sequence of arbitrary length, this architecture generates one tag for each input word. Since this architecture does not have a decoding LSTM layer, it generates a label immediately after receiving an input token. Similar to previous architecture, a time distributed dense layer is used (Figure 2) to ensure that a sequence of labels is obtained once all the tokens in the input sequence have been seen.

Experiments and Results

We performed two categories of experiments with the above architectures for various model configurations presented in Tables 1. In the first category, we used one-hot-encoding vector representation of each token in the dialogue utterances. In the second category, we used word2vec (Mikolov et al. 2013), a distributed word vector representation (word embedding) for each token in the dialogues. In order to represent words that are not present in the word2vec model, we used a vector average of the synonyms of the corresponding word. Also, if synonyms are not available, we randomly initialized the vector for that word. The performance of the models were evaluated using standard performance measures such as precision, recall and F-1 scores obtained using a 10-folds cross-validation methodology. Results were obtained for identifying the cue, scope and focus, respectively. While annotating our tutorial dialogues, we followed annotation guideline proposed by Morante and colleagues (Morante, Schrauwen and Daelemans 2011; Morante and Blanco 2012), however, in constrast to their data, which was extracted from PropBank and consists of non-dialogue texts, our dataset contains dialogue data.

Dataset

As mentioned, we use an annotated corpus extracted from the set of dialogues between a computer tutor (DeepTutor\(^1\)) and high-school students. During the tutor-tutee interactions, students are challenged to solve conceptual physics problems and if they struggle then a scaffolding dialogue is initiated in which the computer tutor is trying to help the student solve the problem based on socio-constructivist theories of learning. As previously noted, 9.36% of the student utterances in the dataset contain at least one explicit negation cue word such as no and not.

The DT-Neg corpus consists of 1,088 instances of student utterances with at least one negation cue word. Each of these utterances were manually annotated with negation cue word, negation scope and negation focus. As shown in the examples below, the scope and focus could be in same dialogue utterance (Example 1) or in the previous dialogue utterance (Example 2).

Example 1: Q: According to Newton’s first law, if the parachutist moves with constant velocity what is the net force acting on the parachutist?  
A: \{\text{no forces}\}

Example 2: Q: According to Newton’s first law, if the parachutist moves with constant velocity what is the \{net force\} acting on the parachutist?  
A: \{\text{no} \}

About 42% of the instances in the DT-Neg corpus have the scope and focus located in the previous dialogue utterance (i.e., dialogue context).

Results

Tables 2, 3 and 4 show the average precision, recall and F-1 scores based on 10-folds cross validation for five different model configurations. In average, M4 has the highest F-1 scores for the detection of focus (0.839), scope (0.857) and cue (0.955) when one-hot-encoding is used for input sequences. Also M4 exhibited best F-1 scores when word embeddings for input sequences were used. From Table 3, 2 & 4, it can be seen that the F-1 measure improves from first model to the fourth when one-hot-encoding was used. In fact, we experimented with more configurations, however we present only the best ones here. The improvement in the F-1 score for scope and focus detection for M3 shows the positive effect of using dropout on the generalization power of the underlying model. Adding one more layer for encoder-decoder slightly improved performance. For focus detection, the F-1 score (0.839) is higher than that obtained (0.826) in previous work by Banjade and colleagues (Banjade, Niraula and Rus 2016).

\(^1\)www://deeptutor.org/
Figure 3: F-1 scores of 10-folds cross validations of different models for focus, scope and cue detection using one-hot-encoding.

Figure 4: F-1 scores of 10-folds cross validations of different models for focus, scope and cue detection using word embedding.

Table 2: Detection of scope.

<table>
<thead>
<tr>
<th>Model</th>
<th>One Hot</th>
<th>Word Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>M1</td>
<td>0.824</td>
<td>0.802</td>
</tr>
<tr>
<td>M2</td>
<td>0.829</td>
<td>0.821</td>
</tr>
<tr>
<td>M3</td>
<td>0.860</td>
<td>0.847</td>
</tr>
<tr>
<td>M4</td>
<td>0.863</td>
<td>0.851</td>
</tr>
<tr>
<td>M5</td>
<td>0.697</td>
<td>0.551</td>
</tr>
</tbody>
</table>

Table 3: Detection of focus.

<table>
<thead>
<tr>
<th>Model</th>
<th>One Hot</th>
<th>Word Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>M1</td>
<td>0.812</td>
<td>0.791</td>
</tr>
<tr>
<td>M2</td>
<td>0.819</td>
<td>0.788</td>
</tr>
<tr>
<td>M3</td>
<td>0.832</td>
<td>0.812</td>
</tr>
<tr>
<td>M4</td>
<td>0.839</td>
<td>0.843</td>
</tr>
<tr>
<td>M5</td>
<td>0.473</td>
<td>0.315</td>
</tr>
</tbody>
</table>

Table 4: Detection of cue.

<table>
<thead>
<tr>
<th>Model</th>
<th>One Hot</th>
<th>Word Embedding</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>M1</td>
<td>0.994</td>
<td>0.985</td>
</tr>
<tr>
<td>M2</td>
<td>0.991</td>
<td>0.991</td>
</tr>
<tr>
<td>M3</td>
<td>0.991</td>
<td>0.995</td>
</tr>
<tr>
<td>M4</td>
<td>0.992</td>
<td>0.999</td>
</tr>
<tr>
<td>M5</td>
<td>0.621</td>
<td>0.493</td>
</tr>
</tbody>
</table>

Four of our models, i.e. all except M5, were able to predict the cues almost perfectly. Since our corpus consisted of limited number of unique cue words, the models were able to see those words most often during training. Among the five models, M5 differs with respect to the other models in its architecture.

It is interesting to see that most of the models (except M5) trained with one-hot-encoding outperformed the model trained with word embedding for negation focus and scope detection. However, in the case of cue detection, models with word embeddings performed better. This could be explained by the fact that the word embedding vectors have been learned from a huge collection documents and our models were confused to some extent to generalize the DeepTutor data which is limited in size and diversity, whereas in the case of M5, the absence of separate encoding and decoding phases could be the reason behind its poor performance as it had to predict the label immediately without seeing the full input sequence.

Figures 3 and 4 show the F-1 scores (10-fold cross validation) for focus, scope and cue obtained with our models trained, respectively, with one-hot-encoding and word embeddings. From the figures, it can be seen that the the performance of all models except M5 remained consistent during each fold. Among the five models we presented, M4 performed best in average as well as in individual run on 10-folds cross validation. The best performance of M4 is confirmed by its topmost position for the F-1 score plots of 10-folds cross validation. Also after through analysis of each run of 10-folds cross validation, we found that M4, had a best (among 10 different runs) F-1 score of 0.885 with and kappa of 0.822 for focus detection and F-1 of 0.889 with Cohen’s Kappa of 0.840 for scope detection.
Conclusion and Future Works
We explored various LSTM models and configurations for handling negation in tutorial dialogues. We have experimented with five models, broadly with two distinct network architecture, namely sequence to sequence tagger and tag sequence generator. We experimented and validated our models using real dialogues between student and an intelligent tutoring system. Our experiments suggests that the sequence to sequence tagger can handle well the subtasks of negation scope, focus and cue detection, outperforming a previous method based on CRF that relied on human engineered features. A good choice of hyper-parameters of LSTM such as dropouts, number of units and layers could result in a competitive model for negation handling.

In this work, we have considered only the previous context while training the model. However, experiments have shown that LSTM RNNs trained with both past and future context perform better. In future work, we plan to build models that could be trained with both past and future contexts. We also plan to experiment with a hybrid model that uses word embeddings as well as human engineered features.

Acknowledgments
This work was partially supported by The University of Memphis and a contract from the Advanced Distributed Learning Initiative of the United States Department of Defense.

References