Assignment 4: Fourth and Final Obligatory Exercise

Filip Stefaniuk
filipste@student.matnat.uio.no

November 15, 2018

1 Introduction

TODO

2 Preliminaries: Making Sense of the EPE File Format

2.1 Multiple negation instances

Approach suggested in this exercise states that when more than one negation cue appears in the sentence, it should be copied that many times witch each negation cue and scope. To practice this approach, first exercise was to rewrite a toy example in that manner. The result can be found in `./data/toy_multiplied.tex`.

2.2 Working with EPE structure

To make working with the EPE structures simpler I have written Python class that embedds sentences and allows to easily extract needed information. Note that no more additional space or conversion is needed since the class stores all the data in the epe dictionary like structure and the information is extracted dynamically. Class also have static method that allows creating instances from json strings, thus loading epe file is simply calling this method on each line of the file. Sentences may be converted to `.tt` format by calling `to_tt()` method.

2.3 Summary Statistics Extractions

I have extracted requested statistics from all the available sentences (using both training and developement set). Statistics below are computed for all the types of negation cues and for multi-token and affix cues separately. General negation scope has on average length of 8. Number of negation tokens drops extremely quickly, to the point where most of the negations occur only once.

When analysing the multi-token negation cues it is worth noticing, that some of those have length equal 0, this is the case when the negation cue makes the whole sentence, in example "*By no means.*"

3 A Joint Cue and Scope Sequence Tagger

3.1 End-to-end Negation Resolution System

The Negation Resolution System that I have build uses Estimator class as a center point. This class is responsible for building the model and provides interface to train and evaluate (it is convinient to use `train.py` and `eval.py` scripts). To transform EPE sentenceces to data format that may be feed to keras classifier, estimator makes use of preprocessor and postprocessor.

Preprocessor extracts relevant information from the list of sentences, builds numpy arrays of encoded input tokens, cue information and pos tags, with coresponding labels. It does that in memory efficient manner by creating numpy arrays of fixed size and filling them with relvant information.

	tokens	type	mean scope len	count
0	not	CUE	8.55	398
1	no	CUE	5.68	258
2	n't	CUE	7.86	85
3	nothing	CUE	7.15	71
4	never	CUE	8.97	69
5	without	CUE	4.39	31
6	none	CUE	3.58	12
7	impossible	AFFIX	8.09	11
8	nor	CUE	14.80	10
9	unable	AFFIX	7.25	8

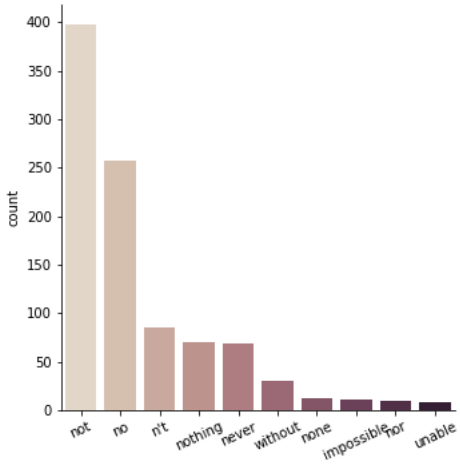


Figure 1: General statistics for negation cues.

	tokens	mean scope len	count
0	neither/nor	8.0	4
1	by/no/means	2.0	4
2	on/the/contrary	0.0	2
3	not/not	14.0	1
4	rather/than	8.0	1
5	nothing/at/all	0.0	1
6	not/for/the/world	0.0	1
7	no/nor	8.0	1
8	no/more	8.0	1

(a) multi-token negation cues

	tokens	mean scope len	count
0	impossible	8.09	11
1	unable	7.25	8
2	unknown	2.86	7
3	unhappy	4.83	6
4	unlikely	11.50	4
5	unpleasant	5.25	4
6	useless	7.25	4
7	motionless	3.00	3
8	unambitious	3.33	3
9	imprudent	5.33	3

(b) affix negation cues

Figure 2: count and mean scope length for 10 most frequent cues

Postprocessor creates new list of sentences updated with list of given predicted labels. To maintain compatibility with evaluation system, following heuristics are applied:

- when there is no negation cue in predicted tags for a given sentence, all the tokens are set to out of negation scope token **T**
- when all the tokens are predicted as **T** but there was a cue in input sentence, `negations` is set to 0 and `negation` is removed from every node.
- when token is predicted as affix tag **A**, lookup for this token is performed in the dictionary of known affix cues. If the token is found, **negation** information is filled accordingly otherwise the whole token is treated as a *cue* and *scope* is set to be an empty string.
- *event* value is copied from the input data.

3.2 Evaluation of Baseline System

I have evaluated baseline system using both LSTM and GRU recurrent neural network architectures. I have used following setting:

- input was padded to length **100**
- I have used randomly initialized word embeddings with size **300**.
- Adam optimizer with learning rate **0.0001**

- **200** neurons in recurrent neural network layer.
- early stopping after **10** epochs without loss improvement on validation data.

There wasn't much of a difference when training with GRU compared to LSTM. On the plot to the right it is shown that the loss value was almost identical. Training with LSTM was slightly slower (one epoch took 50s compared to 40s with GRU). Scope token detection were better captured by GRU, but LSTM had better score on cues detection.

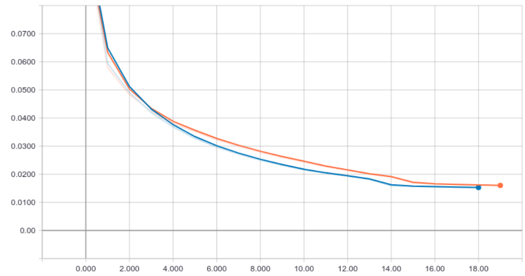


Figure 3: loss during training with LSTM (orange) and GRU (blue)

Tag accuracy because the classes were very unbalanced model had problems with learning how to classify classes other that **T**. This is particulary visble with Affix **A** class, where there were only 33 examples and not even one was classified correctly.

***SEM 2012 Scorer** evaluation using official *SEM 2012 scorer yelded very similar results. Score of *Scope Tokens* that corresponds to *False* if similar, I assume the 3% difference is caused by postprocessing heuristics. *Cues* and *Cue* tag have score difference of more than 10% this is caused by the fact that in tag evaluation affix cues are treated as a different class, and in official scorrer they are counted as cues. Model predicts those cues very poorly, thus they lower the overall score.

	precision	recall	f1-score	support
True	0.95	0.98	0.96	12758
False	0.75	0.53	**0.62**	1335
Cue	0.73	0.85	**0.78**	146
Affix Cue	0.00	0.00	0.00	33
avg(micro) / total	0.93	0.93	0.93	14272
avg(macro) / total	0.61	0.59	0.59	14272

(a) LSTM

	precision	recall	f1-score	support
True	0.95	0.98	0.97	12758
False	0.80	0.55	**0.65**	1335
Cue	0.73	0.82	**0.77**	146
Affix Cue	0.00	0.00	0.00	33
avg(micro) / total	0.93	0.94	0.93	14272
avg(macro) / total	0.62	0.59	0.60	14272

(b) GRU

Figure 4: tagging accuracy

	gold	system	tp	fp	fn	precision (%)	recall (%)	F1 (%)
Cues	173	142	94	18	79	83.93	54.34	**65.97**
Scope tokens	1348	908	684	224	664	75.33	50.74	**60.64**

(a) LSTM

	gold	system	tp	fp	fn	precision (%)	recall (%)	F1 (%)
Cues	173	132	87	15	86	85.29	50.29	**63.27**
Scope tokens	1348	907	715	192	633	78.83	53.04	**63.41**

(b) GRU

Figure 5: Evaluation results using official *SEM 2012 scorer

4 Zooming in on Negation Scope

4.1 Differences between systems

I have read the paper by Fancellu et al. and their code from github. It seems that their experimental setup was slightly different than what we were supposed to do. First of all, they used only sentences with at least one negation, which makes sence since it makes the token classes a bit more balanced. Second they treated affix cue differently, instead of adding a separate class they split them into *cue* and *scope* parts. Finally they focused on predicting scope tokens and not cues, they added cue information to input data in form of an embedding.

4.2 Evaluation of Negation System

I tried to replicate the results achieved by Fancellu by making necessairy changes in my system. I filtered the sentences and was left with **983** training samples and **173** validation samples. Data split is a bit different to the one used by Fancellu (**848** training, **235** validation) but I believe it is similar enough to produce comparable results. I have added cue information as an embedding of size **50**. I used the same hidden size, learning rate, optimizer, max input length as in paper. I have left affix cues as a separate class as instructed in the assignment. I have trained and evaluated the system with different settings:

Cue info (C) word embedding matrix randomly initialized and updated during training.

External embeddings (E) usage of external embeddings in non-static mode, I used 840B glove vectors.

PoS information (PoS) separate embedding for pos tags, initilized randomly and updated during trainig.

	gold	system	tp	fp	fn	precision (%)	recall (%)	F1 (%)
BiLSTM - C	1348	1156	1054	102	294	91.18	78.19	84.19
BiLSTM - C + PoS	1348	1168	1069	96	279	91.76	79.30	**85.08**
BiLSTM - C + E	1348	1215	1075	149	273	88.48	79.75	83.89
BiLSTM - C + E + PoS	1348	1245	1084	161	264	87.07	80.42	83.61

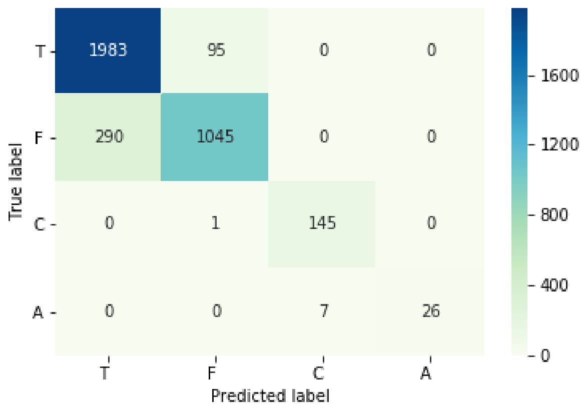
Figure 6: results of the scope detection

In comparison to Fancellu's system, mine performed worse (Fancellu achieved best F1 score **88.72**). I believe the difference in the final results is caused by multiple minor differences, namely:

- the fact that in my systemaffix cues are treated differently
- the datasets are not exactly the same
- different word embeddings.
- different postprocessing heuristics

5 Error Analysis: Looking on our Data

I decided to perform error analysis on best model that is **BiLSTM - C + POS**. I started with plotting confusion matrix on tag level classification. Not suprisingly the model can distinguish perfectly between negation cues and scopes, since we provide this information explicitly. It also looks that it can distinguish quite well between affix and normal negation cues, and since the 3rd part of this assignment was focused on detecting solely negation scopes I want to focus on that aspect in my further analysis. Separately for **F** misclassified as **T** and **T** misclassified as **F** I performed following analysis: I have selected sentences in which such misclassification occurs and sorted them by the number of misclassified tokens. Then I highlighted all the scope tokens (or out of scope tokens in the second case) from the gold standard in blue, then I highlighted correct predictions in green and misclassification in red.



Scope tokens missclassified as out of scope tokens occured more frequent, one class of errors that seemd to be very popular is when the negation scope continues after conjunction or punctuation:

> I desire you to spare no expense and no pains to get at the truth.
> I desire you to spare no expense and no pains to get at the truth.

Suprisingly in multiple cases subject was excluded from the negation scope:

> The crime was ascribed to Nihilism, and the murderers were never arrested.
> The crime was ascribed to Nihilism, and the murderers were never arrested.

Out of scope tokens classified as scope tokens TODO

6 Further work

TODO