Assignment 4: Relation Extraction

CSE 538 Fall 2019
Released: Nov 15
Due: Nov 29 11:59pm

1 Overview

In this assignment you will implement multiple models that accomplish relation extraction within the definition of SemEval task 8 (2010). The first model will be similar to what you've worked with throughout the semester, where you will be given some skeleton code and you need to fill in the gaps. However, the second model is left *entirely* up to you. The goal is for you to apply what you've learned and create a model that beats the performance of the baseline model.

2 Dataset and Task

For this assignment we will be working with the data released for SemEval's relation extraction task from 2010. Ultimately, it's broken down into 8,000 training examples and 2,000 testing. In total there are 10 relations (cause-effect, instrument-agency, product-producer, content-container, entity-origin, entity-destination, component-whole, member-collection, message-topic, and other). Each individual relation will have extra documentation available with some examples in the dataset directory.

Generally each sentence has 2 elements tagged as $< e1 >< /e1 >$ and $< e2 >< /e2 >$ and your goal is to define the relation **and** direction of the relation, i.e. cause-effect(e1,e2) is not the same as cause-effect(e2,e1). It's highly suggested to start by reading the (short) paper that outlines the task linked in the resources section[Hendrickx et al., 2009].

The specific files you will use for train/validation/test are described in the README of the assignment.

3 Getting Started

The template code you will be given follows a similar structure to previous assignments. The code you are receiving will handle the following:

- Load pretrained glove embeddings
- Setup PoS tag embeddings
- Parse shortest dependency path between 2 marked elements
- Load the dataset, generate instances, batches, etc
- Setup (basic) train/test loops
- Outlines your first model
- Generates test pred file

Instructions on the individual files can be found in the README file associated with the assignment. It's important to keep in mind that the train_lib.py file contains the training loop that is used in train_basic.py and train_advanced.py. For your second model you may need to tweak the code inside of train_advanced.py to meet your needs.

3.1 Features

As like the previous assignments you have similar information about the data, plus some new ones. In this assignment you have access to:

- pretrained word embeddings (glove 6B)
- pos tag embeddings (randomized/learnable NOT pretrained)
- the shortest dependency parse between the 2 entities of interest[1]

3.2 Baseline Model [25 pts]

The first model you will need to implement is a GRU with attention. You must implement the attention layer from scratch and also write the call function for this model. Basic variable definitions will be given to you this time in the init function. For this implementation I loosely followed the paper Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification [Zhou et al., 2016]. I mostly followed what they implemented except I did not experiment with dropout as well as opted to use a GRU cell rnn. The areas to note from the paper are as follows:

- Implement a standard bi-directional GRU (instead of their LSTMs) [10 pts]
- Implement an attention layer as described in section 3.3 [10 pts]
- Implement L2 regularization as described in section 4.1 [5 pts]

[1]The input sequence is then reduced to this shortest path. If for some reason it fails to find a path between the entities the entire sentence is used

If your implementation is correct you should score .57 - .60 F1 on validation set using default params and using word, pos, and dependency features. On my laptop (3yr old, i7-7500u) each epoch took around 45 seconds so this is doable without a GPU. Note: I used the 100D pretrained glove embeddings to match the input dimension of the above paper.

3.2.1 GRU Experiments

Once you have a working implementation we want you to explore a bit of the feature spaces with the following experiments (tips on how to easily turn on/off these features are in the README):

1. Run with only word embedding features
2. Run with word + pos features
3. Run with word + dep structure features

3.3 Advanced Model [50 pts]

This is the core of assignment. This is where you can apply everything learned over this course and come up with a custom model. There will be no template code for this, although a skeleton class exists in the model.py file for you to add your code. The goal is to create a better model that outperforms the biGRU with attention.

You have complete freedom for this section, your only limitation will most likely be compute resources. There is no restriction on libraries or functions to use. However, if you **need** to import a new library that does not exist within the environment please update your requirements.txt so we are aware of what you changed.

Additional to the justification of your architecture in your report, we require everyone also submit a brief (1 - 2 paragraph(s) max) readme explaining your architecture blueprint (i.e. I have 3 conv layers, a max pooling, then a linear classifier, etc) as well as a figure that summarizes your architecture.

The grading breakdown is as follows:

- Model Implementation [30 pts]
- Model Performance on test set [10 pts]
- README overview [5 pts]
- Figure [5 pts]

3.4 Evaluation

After you are pleased with your models you should run them on the given test instances. You can submit **THREE** test predictions for your custom model (This allows you to play around a bit with hyper parameters). We will compare

your test predication file to the *hidden* test set. Which means your submissions are mostly blind, beyond picking your best implementation on the validation set.

For the baseline GRU model just submit a single test prediction file for evaluation. The specific files required for submission are outlined in the README of this assignment.

4 Report [25 pts]

Your report should contain the following:

- Explanation of your GRU + attention implementation [5 pts]
- Observations of GRU experiment 1 [3 pts]
- Observations of GRU experiment 2 [3 pts]
- Observations of GRU experiment 3 [4 pts]
- Justification/Motivation of advanced model choice [10 pts]

The suggested length of the report should be 2 - 4 pages, where the bulk should be your justification of model implementation. The experimental observations should be around 1 paragraph each and the GRU/Attention implementation should be no longer than 1 page.

5 Resources

Below are the (clickable) links to the papers necessary for this assignment

- SemEval-2010 task 8: multi-way classification of semantic relations between pairs of nominalsr
- Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification

References

[Hendrickx et al., 2009] Hendrickx, I., Kim, S. N., Kozareva, Z., Nakov, P., Ó Séaghdha, D., Padó, S., Pennacchiotti, M., Romano, L., and Szpakowicz, S. (2009). Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions*, pages 94–99. Association for Computational Linguistics.

[Zhou et al., 2016] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., and Xu, B. (2016). Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212.