CSE599 - M.S Thesis

Progress report

Dhaval Bagal

Codebase

Week 1

August 23, 2021 - August 29, 2021

Codebase

All possible variations of NER

- ***Flat entities:*** These are regular entities. There are tons of papers for this.

- ***Nested entities:*** Entities completely encapsulating other entities. There are many state of the art techniques for this one too.

- ***Overlapping entities:*** These can occur in rare cases where two entities are not nested completely within each other, rather they have some common overlapping portion. Papers which have addressed this problem have mostly used some variation of exhaustive region listing. But this is still computationally not the best.

- ***Two entities starting/ending with the same token:*** This variation is addressed by some papers using sequence labelling models. They label every token according to BILOU scheme (or any other scheme), and then each B and L tags are paired to get a region which is then classified by the classifier.

Variations summed up: Basically we should possibly design a technique that'll have the freedom to label every token with multiple positional as well as categorical tags. E.g: Every token has some probability of it being B,I,L,O,U tag and it belonging to multiple entity categories.

Observation: None of the papers have addressed all of these variations at once.

What we can do?
Possibly find out a method that addresses all these variations.

Codebase

Proposed method

- ***Solution:*** Flat Sequence Labelling
- Let ‘m’ be the number of entity types. For every token, we output a probability vector of length m+5 (5 for the positional tags - BILOU).
- Thus every token can have multiple category types as well as multiple positional tags.

TODO FOR NEXT WEEK

Since we expect the model to give us a probability vector with multiple possible outputs, we need to make sure that the model doesn’t bluff the probabilities, i.e it learns from the data and then outputs the them accordingly.

So the task for next week is to study some feature engineering techniques starting from CRFs, some text encoding methods and some popular architectures like BERT.

Codebase

Week 2

August 30, 2021 - September 5, 2021

Codebase

Insights from research papers

- Predicting tags by looking at the context and masking the current word.

- Using CNNs for generation of embeddings.

- **Pooled contextualised embeddings** which concatenates the original word embedding along with the a pooled embedding which is obtained by pooling over all previous embeddings for every unique word.

- ***NER Evaluation Metrics:***

- ***Exact match evaluation*** implies that entity is recognised correctly only if both boundaries and type match ground truth. ***Macro averaged F-score*** computes the F-score independently for each entity type and then takes the average. ***Micro averaged F-score*** aggregates the contributions of entities from all classes to compute the average.

- ***Relaxed match evaluation*** implies that a correct type is credited if an entity is assigned its correct type regardless of the boundaries as long as there is some overlap with the ground truth.

Codebase

- **CharNER** considers a sentence as sequence of characters. It outputs a tag distribution for each character and then word level tags are obtained from the character level tags.

- **Rei et al.** combined character and word embeddings using gating mechanism so that the model dynamically decides how much information to use from a character or word level component.

- Adding other features to word embeddings apart from word and character embedding (e.g: spelling features, morphology features, etc) boosts tagging accuracy.

- **Pointer networks** first identify a chunk and then label it. E.g: For the sentence “Michael Jeffrey Jordan was born …”, given the start token <s>, the segment “Michael Jeffrey Jordan” is first identified and then labelled as PERSON. (Thus we get completely rid of positional information).

TODO FOR NEXT WEEK

- Identifying datasets suitable for our problem statement.
- Developing PyTorch data loaders for those datasets.

Codebase

Week 3

September 6, 2021 - September 12, 2021

Codebase

Looking into the datasets

- NNE seems to be the best dataset for nested NER, because it has upto 6 levels of nesting. However, it is based on the Treebank dataset which needs LDC membership.

- ACE 2004 and 2005 datasets too require LDC membership.

- We are left with GENIA dataset which is open sourced and free.

- We'll be specifically using GENIA-NER dataset v3.02 for nested NER. The dataset is one single XML file. This week was little busy and therefore managed to understand the structure of the dataset.

Codebase

Week 4

September 13, 2021 - September 19, 2021

Codebase

GENIA Dataloader complete!

- GENIA dataset for nested NER (v3.0.2) is a single xml file with a particular hierarchical structure.

- For our statement, what is important to us is the text within each <sentence></sentence> tag.

- Using regular ***xml*** library provided with python leads to following issue

Let's see the results of the ***xml*** library for the example shown. Extracting the text for the <*sentence*> tag, gives us the text *'Activation of the'.* This is clearly incorrect. Even if we fetch the elements recursively, the library will still miss the text at the end i.e *'lead to …'*. Thus, there is a need to parse this file recursively and label individual tokens according to the tags.

```
<sentence>
Activation of the
<cons sem="protein">
    CD28 receptors
</cons>
lead to …
</sentence>
```

Codebase

Given the XML file as input to ***GENIADataLoader*** module, the output is a dictionary with keys as labels and values as the location of instances of those labels in the entire dataset.

TODO FOR NEXT WEEK

- Checking the correctness of the data loader in case of any errors
- Extending this data loader in Pytorch

```
XML:
----
<sentence>Activation of the <cons lex="CD28_surface_receptor" sem="G#protein_family_or_group"><cons lex="CD28"
sem="G#protein_molecule">CD28</cons> surface receptor</cons> provides a major costimulatory signal for <cons
lex="T_cell_activation" sem="G#other_name">T cell activation</cons> resulting in enhanced production of <cons
lex="interleukin-2" sem="G#protein_molecule">interleukin-2</cons> (<cons lex="IL-2" sem="G#protein_molecule">IL-2</
cons>) and <cons lex="cell_proliferation" sem="G#other_name">cell proliferation</cons>.</sentence>


Annotations:
------------
{'O': [[0, 32]], 'protein_family_or_group': [[3, 6]], 'protein_molecule': [[3, 4], [21, 23], [25, 27]], 'other_name':
[[13, 15], [30, 31]]}

 ===
| O |
 ===

['Activation', 'of', 'the', 'CD', '28', 'surface', 'receptor', 'provides', 'a', 'major', 'costimulatory', 'signal',
'for', 'T', 'cell', 'activation', 'resulting', 'in', 'enhanced', 'production', 'of', 'interleukin', '-', '2', '(',
'IL', '-', '2', ')', 'and', 'cell', 'proliferation', '.']

 ========================
| protein_family_or_group |
 ========================

['CD', '28', 'surface', 'receptor']

 =================
| protein_molecule |
 =================

['CD', '28']
['interleukin', '-', '2']
['IL', '-', '2']

 ============
| other_name |
 ============

['T', 'cell', 'activation']
['cell', 'proliferation']
```

Codebase

Week 5

September 20, 2021 - September 26, 2021

Codebase

GENIA dataloader changes!

- As per the suggestion of the professor, instead of regular expressions the data loader now uses BeautifulSoup for extracting the data from the XML content.

- The issue mentioned in the Week 4 slides is now completely eliminated with the introduction of BeautifulSoup library. However, the annotation format needs to be rethought.

- This week I managed to modify the data loader, but the annotation logic still needs to be designed.

- Apart from the data loader, a recent state-of-the-art is ***Reformers***. Found a GitHub repository which implements reformers in pytorch.

- Planning to use Reformer architecture as the first model in the experiment

TODO FOR NEXT WEEK

- Finalise the data loader and begin with the reformer model

Codebase

Week 6

September 27, 2021 - October 3, 2021

Codebase

Dataloader still on!

- Performed batching, data cleaning and XML extraction and processing.
- The code for data loader is ready, however, there is a minor bug in fetching the annotations.
- Challenges:
  - Consumed the entire RAM and the program timed out because of the huge size of the dataset after processing. The solution that worked then was to split the one XML file into batches with each file containing just one abstract (batch_size=1).
  - To further optimise the size of the annotations, I converted the labels into indices and stored both the tokenised input and its integer labels in a JSON file as a result of preprocessing. Thus, the data preprocessor first converts xml to json and then the data loader will read the json files to load the data into torch tensors.

TODO FOR NEXT WEEK

- Start with the training script for experiment 1

UPDATE

- The bugs in the data loader are resolved.
- Correctness of the data loader verified.

Codebase

Week 7

October 4, 2021 - October 10, 2021

Codebase

Dataloader done, modelling begins!

- Dataset preprocessing done.
- Resolved some major errors in data loader.
- Dataloaders in pytorch are fully functioning.
- Yet to start with the training.
- Also, for reformers, ***reformer_pytorch*** library is available. So building the model won't take any time.

TODO FOR NEXT WEEK

- Training reformer model and analysing the results

Codebase

Week 8

October 11, 2021 - October 17, 2021

Codebase

Took a break because of mid-sem exams!

Codebase

Week 9

October 18, 2021 - October 24, 2021

Codebase

Took a break because of mid-sem exams!

Codebase

Week 10

October 25, 2021 - October 31, 2021

Codebase

Took a break because of mid-sem exams!

Codebase

Training the model!

- Completed the training script and started training the model.

TODO FOR NEXT WEEK

- Testing the results on the test dataset

Codebase

Week 11

November 1, 2021 - November 7, 2021

Codebase

Model evaluation!

- Evaluated the model on the test dataset.
- Got around 99% accuracy on the test dataset after training for just 10 iterations. However, this seemed little fishy so analysed the model further.
- On analysis, it appears that all the probabilities generated by the model after training for 10-20 iterations lie between 0.1 and 0.35.

TODO FOR NEXT WEEK

- Maybe train for more iterations. Not sure thought, since the loss is already very less.

Codebase

Week 12

November 8, 2021 - November 14, 2021

Codebase

Model evaluation metrics updated!

- Designed a Metrics class that prints all the metrics including tp, tn, fp, fn, num_examples, accuracy, precision, recall, f1_score.

- Reduced the number of output categories from 77 to 11 to figure out the obstacles in the training.

- Looking at the evaluation results after few (60) iterations, here is the conclusion:

  - Class 0 which is the outside tag, has a non-zero f1 score in the initial few iterations, however, for all the other classes the f1 score is not defined.

  - This is because of the class imbalance problem. Model is giving more weightage to the outside tag and hence probabilities are quickly learned for outside tag at the cost of other tags.

TODO FOR NEXT WEEK

- Discussing the following proposal with Professor and working on it if he approves: Since outside tag is causing class imbalance, why not change the scheme from BIO to BI. This will be beneficial for the model, but BI scheme is not a standard, so not aware of the potential problems that might come in future.

Codebase

Evaluation metrics											
metric	class_0	class_1	class_2	class_3	class_4	class_5	class_6	class_7	class_8	class_9	class_10
true_positives	0	0	0	0	0	0	0	0	0	0	0
true_negatives	1568840	1689580	1689580	1689440	1689480	1689560	1689520	1689300	1688660	1689300	1688780
false_positives	0	0	0	0	0	0	0	0	0	0	0
false_negatives	120760	20	20	160	120	40	80	300	940	300	820
num_examples	1689600	1689600	1689600	1689600	1689600	1689600	1689600	1689600	1689600	1689600	1689600
accuracy	0.9285	1.0	1.0	0.9999	0.9999	1.0	1.0	0.9998	0.9994	0.9998	0.9995
precision	0	0	0	0	0	0	0	0	0	0	0
recall	0	0	0	0	0	0	0	0	0	0	0
f1_score	None	None	None	None	None	None	None	None	None	None	None

Week 13

November 15, 2021 - November 21, 2021

Codebase

Week 14

November 22, 2021 - November 28, 2021

Codebase

Week 15

November 29, 2021 - December 5, 2021

Codebase

Week 16

December 6, 2021 - December 12, 2021

Codebase

Week 17

December 13, 2021 - December 19, 2021

Codebase

Codebase

Codebase

Codebase