NLP homework 3: Coreference Resolution

Anonymous ACL-IJCNLP submission

1 Introduction

Coreference Resolution (CR) is the task of determining linguistic expressions that refer to the same real-world entity in natural language. It is an important step for a lot of higher level NLP tasks that involve natural language understanding such as document summarization or question answering. CR consists of two main tasks which are Mention Proposal (identify all the candidate entities in a text) and Mention Linking (understand which pairs of mentions correspond to the same entity). It can be approached at three different complexity levels: End-to-End Coreference Resolution, Entity Identification and Resolution, Entity Resolution; my work implemented the last approach over the GAP dataset (Webster et al., 2018). I started by implementing a baseline model composed of contextualized embeddings + BiLSTM, then I progressively added one new component at a time on top of the model and only if it brought improvements on the validation set then it was kept. The sections of this report follow the chronological order of the extensions that I added. The baseline model was then compared with my implementations of some work that has already proven effective for this type of task. (Attree, 2019; Chada, 2019).

2 Data preprocessing

First of all I tokenized the given sentences by splitting on whitespaces. The GAP dataset presents the Entity Resolution problem in a "gold-two-mention" format, formulating it as a classification problem where the model must resolve a given pronoun to either of the two given candidates or neither. So the classes that I defined are: A, B, NEITHER. Neither instances are the most difficult to resolve since they are the least represented class. For each sentence, I also labeled each token with its mention role in the sentence: pronoun, entity A, entity B, none; this was done in order to always know the position of the most relevant tokens of the sentence, of course if an entity is a span of words more than one token will belong to it. For the ProBERT approach it was also necessary to enclose the labeled span of mentions with an associated tag, i.e. <P> for pronoun, <A> for entity A and <B> for entity B. An example tagged sentence can be: *"<A> Bob Suter <A> is the uncle of <B> Dehner <B>. <P> His <P> cousin is Minnesota Wild's captain."*. The primary reason for doing this is to provide the positional information of the labeled mentions implicitly within the text instead of explicitly through additional features.

3 Baseline model architecture

This section describes the baseline model developed for Entity Resolution. All the reported accuracy scores are to be intended on the Entity Resolution task on the validation dataset.

3.1 Contextualized embeddings + BiLSTM

Given an input sentence as a sequence of words, each word needs to be transformed into a meaningful numerical representation. For this purpose I used contextualized word embeddings produced by a Transformer-based model. The contextualized embedding for a given word is very powerful because it will be different based on the context of the sentence, as oppose to non-contextualized embeddings (GloVe, Word2vec, ...) that will produce the same vector representation regardless of the context. In this way, different senses of a word are not collapsed into the main sense, but are all captured based on the surrounding words in the sentence. Since the advent of the Transformer (Vaswani et al., 2017), large pre-trained language models have become the de facto standard for obtaining contextualized word embeddings. I compared different models exposed through the Hugging Face library

[1], as can be seen in table 1, but the final choice was to use a fine-tuned RoBERTa (Liu et al., 2019) model. It is an encoder-only Transformer based on the same architecture as BERT (Devlin et al., 2018), but trained on a lot more data (10x). To obtain an embedding for each word, I averaged the last 4 layers of the Transformer model and averaged the sub-tokens belonging to the same words, using the Transformers Embedder library [2]. After the word embedding layer, I concatenate to each of them a 300-dimensional trainable mention embedding (i.e. an embedding representing if the token is part of pronoun, entity A, entity B or none of them). This was done in order to give to the model a better understanding of which are the most important tokens in the sentence. The concatenated embeddings are then fed to a 2-layer bidirectional LSTM (BiLSTM) (Graves and Schmidhuber, 2005). BiLSTMs stack two LSTMs on top of each other: one will process the input sequence in forward order encoding the context before every token, the other will process it in reverse order encoding the context after each token. The two hidden representations obtained for each token are then concatenated in output, in this way the network knows both the context before and after each token. I also used Dropout (Hinton et al., 2012), a regularization technique that randomly zeroes neurons of the network during training, after each layer because the number of parameters is quite big and I wanted to avoid overfitting. After the BiLSTM I extracted from each sample the embedding produced for the last token and considered it as a “sentence embedding”, then I fed it to a 2-layer MLP for the final classification. This model reaches an accuracy score of **83.92%**, other experiments can be seen in table 1.

3.2 Part of speech tags

Part Of Speech (POS) Tagging is a main task in NLP that can often improve many downstream tasks. Adding POS tags to the input tokens could be useful to add some syntactic information. I mapped each POS tag extracted with Stanza (Qi et al., 2020) into a trainable embedding and concatenated it to the respective word $\circ$ mention embedding ($\circ$ is the concatenation operator), the resulting vector is then fed as input to the BiLSTM. Since adding the POS information to the model didn't bring any improvement in accuracy, this path will not be carried forward.

[1] https://huggingface.co/docs/transformers/index

[2] https://github.com/Riccorl/transformers-embedder

3.3 Pronoun embedding

Here I switched from using the sentence embedding to using the pronoun embedding in order to produce the final classification. So, after the BiLSTM, instead of taking for each sample the embedding produced for the last token I took the average embedding of the tokens belonging to the pronoun, and this is fed to the MLP classifier. This change was done because of the intuition that the feature vector representing the pronoun should contain more informations on whether it refers to entity a, b or neither, than a feature vector representing the entire sentence, which might instead lose some of this information. With this change the model reached an accuracy of **85.92%**.

4 Training

After all the additions I performed one last coarse-grained grid search over hyperparameters as reported in table 2, the final model still reaches an accuracy score of **85.92%** and you can see its confusion matrix in image 2. It is trained using Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 and no L2 regularization. In order to reduce overfitting, every experimented model is trained for 70 epochs using early stopping with 15 epochs of patience, actually causing a model to never be trained for more than 47 epochs. Moreover, the saved weights are the one that produced the best validation accuracy score during the training, this avoids saving noisy weights and further reduces overfitting, as you can see in image 1. Batch size was fixed to 8 and all the models were trained on a local GPU. The loss to be minimized is the Cross Entropy.

5 Other models from literature

At this point I decided to compare this baseline model with my re-implementations of some works that reached the state of the art in CR's literature.

5.1 CorefSeq

In the CorefSeq model presented in (Chada, 2019) we have a BERT model that takes as input a tokenized sentence and produces word embeddings. I replaced the BERT model with a RoBERTa due to its better performance, then sequence features are extracted by concatenating the 3 separated token embeddings corresponding to entity A, B and the pronoun spans. I computed each of these span embeddings by averaging the token embeddings belonging to it. The token embeddings are the average of the last four encoder layers of the fine-tuned RoBERTa. These concatenated features are then fed to a single hidden layer linear neural network with a ReLU activation. This hidden layer has 512 hidden units, the final layer then gives us the classification. Dropout is used to reduce overfitting. The tests on this model are reported in table 3, the best one achieves an accuracy score of **88.22%** and actually made a relevant improvement over my baseline model. Its confusion matrix can be seen in image 3.

5.2 ProBERT

Pronoun BERT (ProBERT) was presented in (Attree, 2019). It uses a fine-tuned BERT language model (which I replaced with a fine-tuned RoBERTa for performance reasons) with a classification head on top. The given text is augmented with mention level tags (as explained in section 2) to capture positional informations, then fed to the language model. After this, the average token representation corresponding to the pronoun tokens is extracted from the last layer of RoBERTa for each sample. This embedding is then classified into the three classes A, B and NEITHER through a single linear layer. This simple architecture only adds $H \cdot 3$ new parameters over RoBERTa (where H = embedding dimension of the language model) allowing the model to use training data more efficiently, and in fact it was the best performing. Tests on this model can be seen in table 4, the best model achieves an accuracy score of **89.86%** and its confusion matrix is reported in image 4.

References

Sandeep Attree. 2019. Gendered ambiguous pronouns shared task: Boosting model confidence by evidence pooling. *ArXiv*, abs/1906.00839.

Rakesh Chada. 2019. Gendered pronoun resolution using BERT and an extractive question answering formulation. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 126–133, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. Cite arxiv:1810.04805Comment: 13 pages.

A. Graves and J. Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052 vol. 4.

Geoffrey Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint*, arXiv.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the gap: A balanced corpus of gendered ambiguou. In *Transactions of the ACL*, page to appear.

Language Model	**BiLSTM layers**	**MLP layers**	**Fine-tuning lr**	**Accuracy**
RoBERTa	3	1	•	82.38%
BERT-base-cased	2	2	•	80.83%
BERT-base-uncased	2	2	•	81.93%
RoBERTa	2	2	•	83.25%
RoBERTa	2	2	1e-5	**83.92%**
RoBERTa	1	2	•	80.83%
RoBERTa	1	1	•	81.93%

Table 1: Different experiments for the initial model with relative results. Where • is present, it means it was not fine-tuned.

Hyperparameter	**Values**
BiLSTM's layers	1 \ **2**
Fully connected layers	1 \ **2**
Mentions embedding size	150 \ **300**
L2 regularization	**0** \ 1e-5

Table 2: Final additional grid search for the baseline was performed on these hyperparameters. Final model's hyperparameters are highlighted in bold.

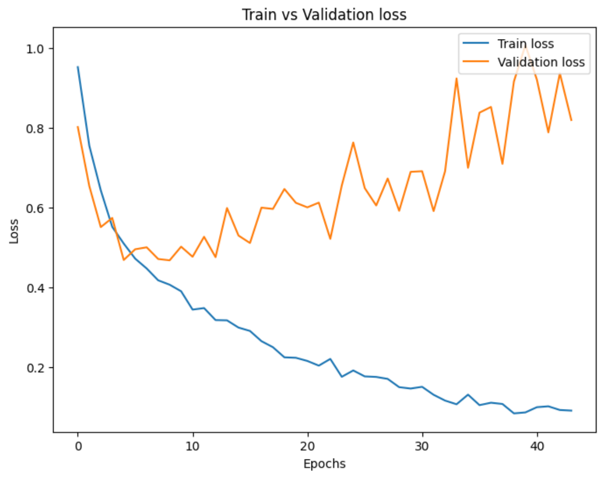

Figure 1: Train vs validation loss of the final baseline model. The best performing weights were saved at epoch 23, patience was consumed at epoch 44.

MLP learning rate	Transformer learning rate	MLP L2 regularization	Accuracy
0.001	1e-05	0	87.32%
0.001	1e-05	1e-05	86.78%
0.001	4e-06	0	**88.22%**
0.001	4e-06	1e-5	87.66%
0.0001	1e-05	0	88.04%
0.0001	1e-05	1e-05	87.88%
0.0001	4e-06	0	88.10%
0.0001	4e-06	1e-5	88.10%

Table 3: Different experiments for the CorefSeq model with relative results.

MLP learning rate	Transformer learning rate	MLP L2 regularization	Accuracy
0.001	1e-05	0	88.98%
0.001	1e-05	1e-05	88.64%
0.001	4e-06	0	87.88%
0.001	4e-06	1e-5	88.86%
0.0001	1e-05	0	88.64%
0.0001	1e-05	1e-05	88.20%
0.0001	4e-06	0	88.86%
0.0001	4e-06	1e-5	**89.86%**

Table 4: Different experiments for the ProBERT model with relative results.

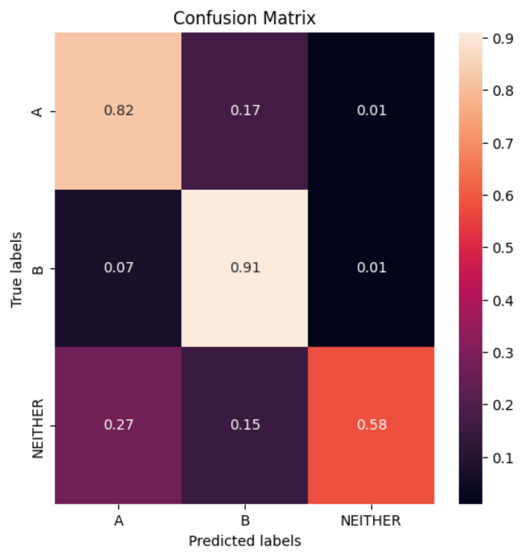

Figure 2: Confusion matrix of the final baseline model, we can see that the NEITHER class is the harder to predict for the model, since it is the less represented.

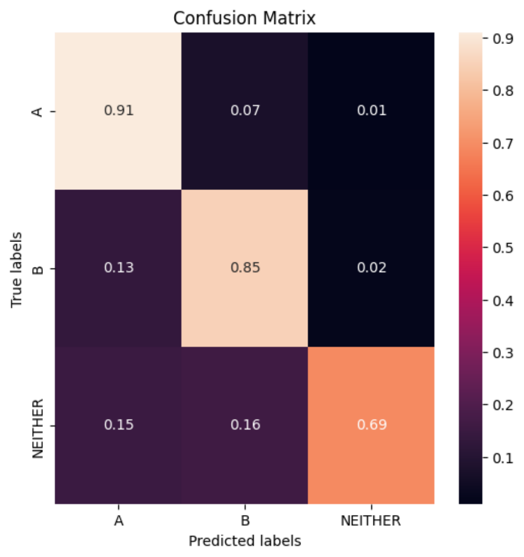

Figure 3: Confusion matrix of the CorefSeq model, we can see that with respect to the baseline the NEITHER class had a big improvement, and the recall of classes A and B switched.

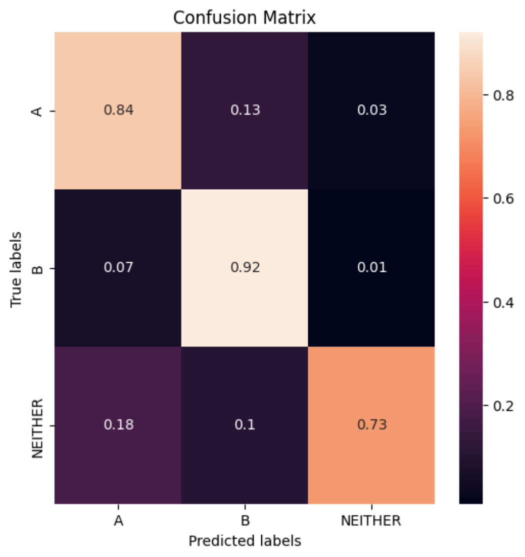

Figure 4: Confusion matrix of the ProBERT model, we can see that with respect to the baseline model all classes are better predicted.