NLP homework 3: GAP Coreference Resolution

Giorgio Strano
strano.1809528@studenti.uniroma1.it

1 Introduction to the task

This project tackles the problem of coreference resolution for gendered ambiguous pronouns, as presented in (Webster et al., 2018). Here I report my solutions to two versions of the task: in the *gold-two-mention* format, the model is given a sentence, an ambiguous pronoun and two cadidate coreferences, and has to determine which one, if any, is correct. In the complete task, the objective is the same, but the model is not given any candidates, so it has to predict the exact portion of text that is referenced by the pronoun. In the original paper, the prediction is considered correct if it either contains the correct label or it is a substring of it. Here, instead, only an exact match will be considered correct. Lastly, an *end-to-end* version of the task was presented, in which the pronoun is not given and has to be predicted by the model. I did not attempt to solve this version of the task due to the limited time available, however it is important to note that most of the sentences in the dataset contain more than one ambiguous pronoun, so a result strongly better than random is hard to foresee.

2 Dataset and preprocessing

The very limited size of the dataset makes the task very hard to solve, as the complex models that are needed to extract enough information from the data are extremely prone to overfitting.

2.1 Supersampling *neither* instances

The aforementioned problem is even more noticeable for the sentences that do not contain a correct coreference (labeled as *neither* in the *gold-two-mention* task), which are extremely underrepresented. To partially face this issue, i followed the path presented in (Attree, 2019), integrating the training dataset with a superset of *neither-instances* that they generated.

2.2 Mention Tags

To encode into each sentence information about the position of the pronoun and, if needed, the two candidates, I enclosed the relevant text between special tokens `[P]`, `[A]` and `[B]`. This removes the need to encode and pass any additional information to the model, other than the sentence itself.

Example:

... and her mother, [A] Ellen [A] , ran a boarding house in Brixton; [B] Kathleen [B] was their third daughter. [P] She [P] was educated at Mary Datchelor Girls' School...

3 Gold-two mention task: baseline model

To solve the *gold-two-mention* task, I used a transformer-based architecture, which allows to leverage the power of complex pre-trained models. Since I only needed the *encoding* step to obtain contextualized embeddings for each word, I decided to only use BERT-based transformers (Devlin et al., 2018). As suggested in (Attree, 2019) with the ProBERT model, for my baseline I stuck to a BERT encoder followed by a linear transformation layer, which takes as input the embedding of the pronoun, and outputs logits for 3 classes (*neither*, *A* or *B*).

3.1 Frozen vs. fine-tuned

It is important to consider, as also noticed in (Joshi et al., 2019) and in (Chada, 2019), that, for this type of task, a fine-tuned BERT model always performs substantially better than one with frozen weights (up to 15% improvements in accuracy), even if at the cost of much slower training and significant memory requirements. For this reason, I used the DistilBERT pretrained model presented in (Sanh et al., 2019), which is 40% lighter than a traditional BERT, with minimal losses of performance.

4 Improvements over the baseline

4.1 Choice of the best embeddings

It is important, when using a pretrained transformer, to pay attention to which layers of the encoding process to use, as the last one is not necessarily the best choice. I found the best results when taking the outputs from the last 4 layers and concatenating them, before feeding the resulting vector to the classifier.

A second change I introduced was to not only classify the embedding of the pronoun, but rather the concatenation of the embeddings of various tokens. In the end I found the most intuitive combination to be the one leading to better performances: I averaged together the vectors coming from each candidate, and used as an input for the classifier a concatenation of the pronoun vector, the average A vector and the average B vector [1].

4.2 More complexity: LSTM and MLP

As a last improvement, I changed the classifier module, adding a hidden layer with 1024 neurons, and, most importantly, added a recurrent module (in the form of a Bi-LSTM) after the BERT contextualized embeddings. This was done with the idea of giving an added layer of relevance (other than the positional embeddings used by BERT) to the relative position of the tokens, since it is clearly a crucial factor when establishing references between parts of a sentence.

5 Results and comparisons

All the results mentioned below, with exception for the accuracy, are computed with the official scorer available from the original repository[2].

5.1 Results: baseline model

Even the baseline, simple as it is, performs very well, reaching an accuracy of 79.96% and overall F1-score of 81.7%. This is a much stronger performance than any of the models mentioned as baselines in the original paper (Webster et al., 2018), even the transformer-based ones, surpassing all of them by 8.8% or more. For an interesting comment on the *"unreasonable effectiveness"* of this simple model I would suggest to read the paragraph 6.1 of (Attree, 2019). In that paper, the baseline reaches an overall F1-score of 87.8 for the model that uses BERT-base and 89.7 when using BERT-large, which is an improvement over my model that is expected when moving from DistilBERT-base to the much more complex BERT-base.

5.2 Results: improved version

The best model I obtained, including the improvements mentioned earlier, reached the performances listed below, improving by 3.6 points the overall F1 score of the baseline.

Acc.	Overall F1	Masc. F1	Fem. F1	Bias
83.48%	85.3%	82.4%	88.1%	1.07

Interestingly, this is the only model I have seen in any publication on this topic which has a positive bias, meaning that it performs better on feminine pronouns than on masculine ones: taking into account only the feminine F1-score, this model even surpasses GREP (Attree, 2019) (the current state of the art model) in its base version, while being able to be trained on much lighter hardware.

I believe that switching from DistilBERT-base to BERT-large, this model could easily improve ProBERT and potentially challenge the current state of the art GREP-large.

6 Complete task: without candidates

In this version of the task, the model is not given any candidates, and has to chose the correct span of the sentence that is referenced by the pronoun. It is immediate to notice that this change makes the challenge extremely harder, as it combines coreference resolution with named entity recognition; we can see, indeed, that the performance of a semi-random model (which chooses a random capitalized word in the sentence as coreference) floats around 6%, as opposed to 33.3% in the *gold-two-mention* variant.

6.1 A first baseline using NER

It is clear that, in this case, the *Named Entity Recognition (NER)* step plays a fundamental role: it is very improbable to imagine a joint approach in which a transformer architecture learns to detect the exact span of the named entity referenced, without having learned beforehand how to recognize a named entity. For this reason, I decided to structure the model as two separate components that can be, however, fine-tuned jointly: a named entity recog-

[1]This operation of averaging vectors belonging to each entity is not needed for the pronoun, because the pronouns are always words small enough to never get split by the tokenizer.

[2]https://github.com/google-research-datasets/gap-coreference

nizer, that finds the candidates, and a selector that picks its favourite one.

For the named entity recognizer, I found the best results with, once again, a pretrained transformer from HuggingFace[3], based on BERT-large-cased. Unfortunately, this model was absolutely impossible to fine-tune on either my computer or Google Colab, so I had to keep it frozen.

After improving its performance with a series of basic logic rules (to avoid predicting I-tags without a preceeding B-tag) it performed very well, and in 95% of the sentences the correct coreference was predicted as a candidate.

6.2 The coreference solver

After choosing the candidates, the taks becomes similar to the previous one, with the exception that the number of candidates to chose from (and therefore labels) is variable for each sentence, so I kept the same structure of contextualized embeddings and LSTM, changing the structure of the classifier as needed.

At first I decided to classify each token into one of four labels (*nothing*, *pronoun*, *B-coref*, *I-coref*), and then pick between the candidates based on the last 2 classes. Unfortunately the approach of classifying each token did not find any success, achieving performances only slightly better than random.

The best approach i found was to treat the position of the *B-coref* tag as the label for each sentence: to do that, I had to pad every sentence to the same length (400) and have 401 labels, ranging from 0 to 400 included, to indicate the position of the coreference in each sentence (label 400 means that there is no coreference). Once detected the first token corresponding to the correct entity, the following I-tags were taken from the named entity tagger. This idea worked much better, although it had its downside in the extreme complexity of the model: to output the exact position of the correct token, the classifier needs to receive a concatenation of every embedding in the sentence, meaning that it would have an input size of $400 * hidden\_vector\_dim$. This led to the need to make many compromises to fit the model in the 8 GB of GPU memory I had available, such as using the LSTM as an artificial bottleneck to reduce the dimensionality of the embeddings from 1024 to 512, aggregating the last 4 layers of the transformer using a sum instead of a concatenation, and removing the hidden layer from the classifier.

7 Results and comparisons

Here it is much harder to make a proper comparison, as the model is divided in two very different components, and the performance of the second depends on the first.

7.1 Results: NER step

The NER module, even without being fine-tuned, worked very well at identifying the correct entity as a candidate ($> 95\%$ accuracy), but, even after filtering by only entities recognized as *Person*, it couldn't narrow down the selection very much. Both because of its nature that favoured recall over precision, and because of the particular composition of the dataset, which was very dense in terms of named entities, the average number of *Person entities* detected was around 6.7 per sentence, giving a baseline accuracy (with the enitity chosen at random) of around 15%. This is in sharp contrast with the results presented in the original paper (Webster et al., 2018), where is achieved an F1-score of 41.5[4] just in the candidate selection step by using the Google Cloud NL API as a named entity recognizer and a series of structural cues that determine whether the entity is in a syntactic position which precludes coreference with the pronoun. Unfortunately, that section was not explained with enough detail to extrapolate more useful information.

7.2 Results: full model

The full model, composed of a frozen BERT-large named entity tagger, DistilBERT-base contextualized embeddings, LSTM and classifier, reached an accuracy of 42.73%, which improved on the random baseline by around 27 points. Here, the hardware limitations were even more challenging than before, since not only I had to use DistilBERT, but I could not fine-tune the NER module, and had to reduce to an absolute minimum every hidden size. It would be very interesting to train this model on proper hardware, and using a candidate identification step as good as the one proposed in the original paper, in which case I think this model would vastly surpass the baselines set by the original paper, just like the previous one.

[3]https://huggingface.co/dbmdz/bert-large-cased-finetuned-conll03-english

[4]Accuracy was not provided, but it usually is slightly lower than F1-score.

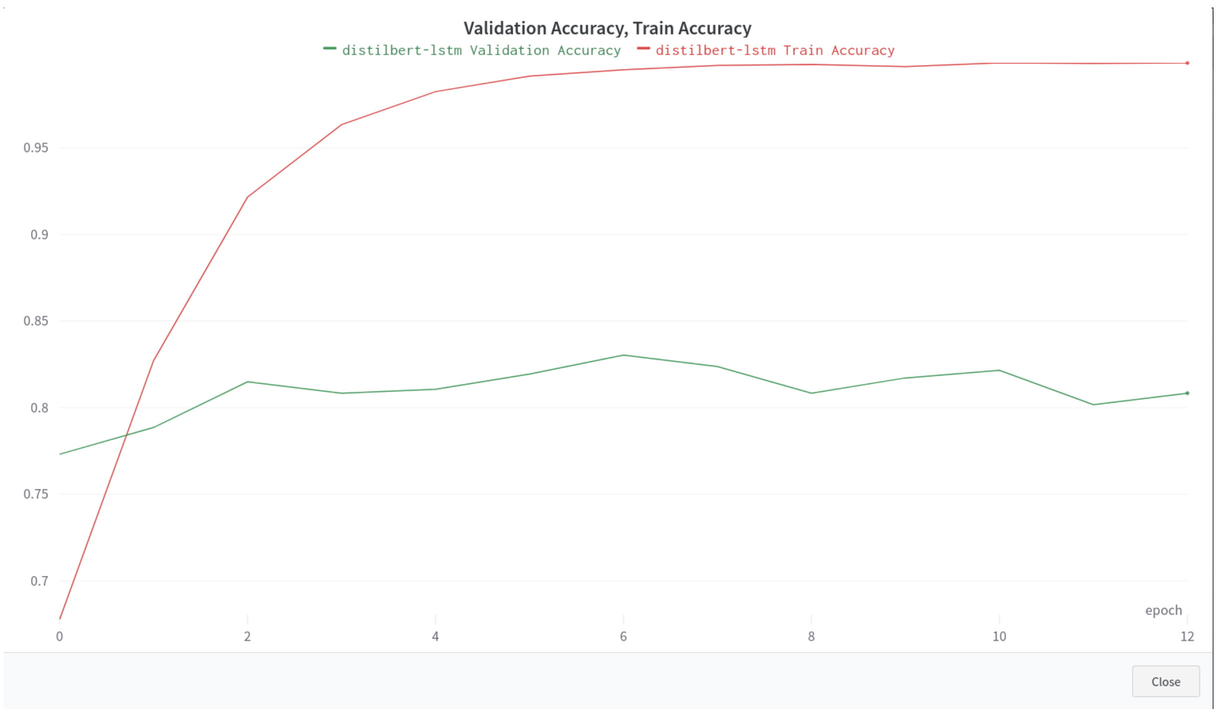

Figure 1: Validation accuracy, train accuracy for the *gold-two-mention* task.

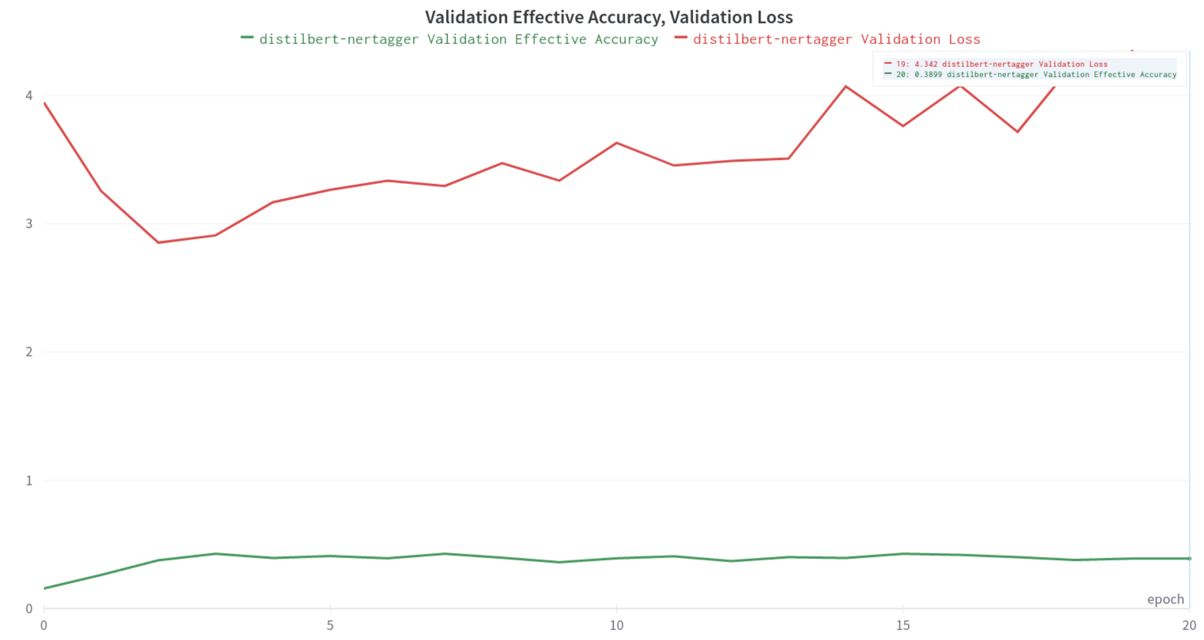

Figure 2: Validation loss, validation accuracy for the complete task.

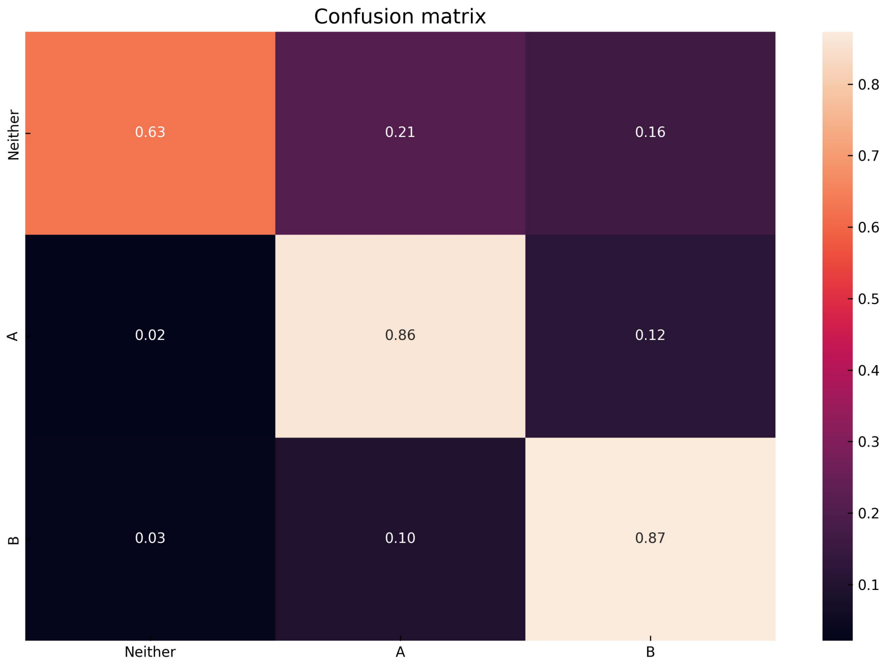

Figure 3: Normalized confusion matrix for the *gold-two-mention* task.

References

Sandeep Attree. 2019. Gendered ambiguous pronouns shared task: Boosting model confidence by evidence pooling. In *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*, pages 134–146, Florence, Italy. Association for Computational Linguistics.

Rakesh Chada. 2019. Gendered pronoun resolution using BERT and an extractive question answering formulation. *CoRR*, abs/1906.03695.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Mandar Joshi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2019. BERT for coreference resolution: Baselines and analysis. *CoRR*, abs/1908.09091.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

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.