On GAP coreference resolution shared task: insights from the 3rd place solution

Artem Abzaliev
GfK SE
Global Data Science
Bamberger Strasse 6, Nuremberg 90425, Germany
artem.abzaliev@gfk.com

Abstract

This paper presents the 3rd-place-winning solution to the GAP coreference resolution shared task. The approach adopted consists of two key components: fine-tuning the BERT language representation model (Devlin et al., 2018) and the usage of external datasets during the training process. The model uses hidden states from the intermediate BERT layers instead of the last layer. The resulting system almost eliminates the difference in log loss per gender during the cross-validation, while providing high performance.

1 Introduction

The GAP coreference resolution shared task promotes gender fair modelling with its GAP dataset (Webster et al., 2018). GAP is a coreference dataset for the resolution of ambiguous pronoun-name pairs in real-world context. GAP has a particular focus on the balance of masculine and feminine pronouns and allows for gender-specific evaluation. The challenge was hosted by Kaggle and consisted of two stages. Stage 1 attracted 838 participants, and stage 2 involved 263 participants.

GAP training examples look the following way:

> Burnett Stone (Peter Fonda) is **Lily's** grandfather and **Lady's** caretaker. He keeps **her** in Muffle Mountain.

where **her** is ambiguous pronoun, **Lily** is candidate mention A, and **Lady** is the candidate mention B. The data was extracted from Wikipedia, so, in addition to the text, the source URL of the article is given. The goal is to predict whether the ambiguous pronoun refers to the mention A, to the mention B or to NEITHER of them. The problem was treated as gold-two-mention task, where the model has the access to the position of the mentions.

2 The data

The GAP dataset is split into training, validation and test set. Training and test set contain 2000 examples each, validation includes 454 examples. During the stage 1 of the competition, the test set was used to evaluate the model performance, while train and validation sets together were used for training with 5 fold cross-validation scheme. This choice initially was made because of the relatively little amount of data and the instability of the predictions (the score on one fold can significantly differ from the other fold). There were several errors in labels of all three GAP dataset, reported by the competition participants[1]. The current solution employs the GAP data with manual corrections.

During the stage 2 of the competition, all three datasets with resulting 4454 observations were used for the training, while the new test set with 12,359 examples was used for the prediction.

2.1 Additional data

There are several coreference datasets available for training and evaluating. Besides GAP data, the presented solution uses four external data sources: Winobias (Zhao et al., 2018), Winogender (Rudinger et al., 2018), The Definite Pronoun Resolution (DPR) Dataset (Rahman and Ng, 2012; Peng et al., 2015) and Ontonotes 5.0 (Pradhan et al., 2012). Each of them was processed to be compatible with the GAP format. After the cleaning this resulted in 39,452 training examples for Ontonotes 5.0, 360 for Winogender, 3162 for Winobias and 1400 for DPR. In this paper, this external data (Ontonotes 5.0, Winobias, Winogdender, DPR) will be called *warm-up data*, because it was used to fine-tune the BERT embeddings, and the weights learned from this data served as 'warm

[1] https://www.kaggle.com/c/gendered-pronoun-resolution/discussion/81331

up' for the training on the GAP dataset.

There was one more candidate to the additonal datasets pool, namely PreCo (Chen et al., 2018), but despite many efforts this dataset did not provide any score improvement. Presumably, this is mainly due to the different structure of the data, and the high amount of noise. For instance, some training examples contained the same word as both mention and pronoun, which may have worsen the model performance.

There were several attempts to include all the additional datasets to the training procedure. The naive attempt to concatenate GAP data and additional data into one big training set did not work, because the additional data has a different structure and does not have the URL feature. The second attempt was to use a two-step approach:

1. Warm-up step: pretrain the part of the model (namely the head, see section 3 for the explanations) on the external data only. Then, select the weights from the model that performs best on the warm-up validation set.
2. GAP step: continue training on GAP data, using the weight from the best-performing warm-up model instead of randomly initialized weights.

The warm-up data was randomly split into training and validation set with 95%-5% proportions. This strategy slightly improved the model performance, showing that warming-up on external dataset can be a promising direction. One possible explanation is that starting with pretrained weights allowed the model to reach flatter optimum and generalize better. In addition, the external data contained many more training examples for the category NEITHER (see Table 1), resulting in better performance for this group.

During the third attempt, the validation set was not randomly chosen, but replaced by Winobias only. It was done to ensure that gender fair representation will be chosen as initialization weights for the GAP step. This action further provided small improvement in the evaluation metric. However, the negative effect of choosing Winobias as validation data was the complete exclusion of the class NEITHER from the validation data (see Table 1). Surprisingly, this effect was not detrimental to the performance, most likely because the training data contained enough training examples for this class.

	GAP train	Warm-up train	Warm-up val
A	43.71%	31.91%	50%
B	44.65%	31.40%	50%
NEITHER	11.63%	36.68%	0

Table 1: Class distribution for the datasets used. GAP train includes all the gap datasets available (gap development, gap val, gap test). Warm-up train includes Ontonotes, DPR and Winogender. Warm-up val only includes Winobias.

The final version of the model also fine-tunes the particular layers of BERT embeddings, in addition to the warm-up of the head. For a full description, see section 3.

2.2 Evaluation metrics and class distributions

Class distributions Table 1 shows the class distributions for the three final datasets used: GAP train, which includes all GAP data, warm-up train, which includes Ontonotes 5.0, Winogdender, DPR and warm-up validation, which is Winobias. As can be seen, warm-up train has the most balanced distribution of classes, while GAP train has a lower portion of the category NEITHER.

Evaluation metrics Solutions were evaluated using the multi-class logarithmic loss. For each pronoun, the participants had to provide the probabilities of it belonging to A, B, or NEITHER. The formula to evaluate the performance of the model is:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij})$$

where N is the number of samples in the test set, M is number of classes, 3, log is the natural logarithm, y_{ij} is 1 if observation i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that observation i belongs to class j.

2.3 Features

Besides the direct textual input, the current solution uses some manually constructed features. The majority of them were already mentioned by Webster et al. (2018) as single baseline models. The following features were used:

- **Token Distance**. Distance between mentions and the pronoun, and also between the mentions themselves.

- **Syntactic distance between mentions and pronoun**. The distances were extracted with the StandfordCoreNLP.
- **URL**. Whether the Wikipedia URL contains the mention.
- **Sentence of the mention**. Index of the sentence, where the mention is located, divided by total number of sentences in the snippet.
- **Syntactic distance to the sentence root**. The distance between the mention and its syntactic parent.
- **Character position**. The relative character position of the mention in the text.
- **Pronoun gender**. Gender of the pronoun. It was noticed that in some examples, mentions were of different gender, so the hope was that this feature could help. It did not help, but it did not hurt either. The fact that this feature did not affect the performance can be a good indicator of gender-neutral learning.

The features that provided the biggest improvements were URL (0.07 decrease on log loss) and syntactic distance between mentions and pronoun (0.01). The contribution of other features was very limited.

3 The system

The final solution uses an ensemble of four neural networks. They are: fifth-to-last-layer with cased BERT, fifth-to-last-layer with uncased BERT, sixth-to-last-layer with uncased BERT, sixth-to-last-layer with cased BERT. The explanation is in the next subsection of the paper. Each network consists of two main parts:

- BERT part: contextual representations of the text with fine-tuned BERT embeddings
- Head part: using the embeddings together with manually crafted features to produce softmax probabilities of the three classes

All networks have the same architecture for the head part and the only difference is in the BERT part.

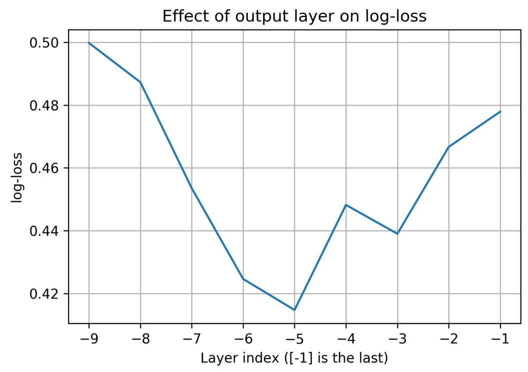

Figure 1: The effect of the output layer choice on the performance of the model. Layer index [-5] means fifth-to-last output layer. Log loss is reported on the stage 1 test dataset, at the beginning of the competition, keeping all the other parameters fixed.

3.1 BERT part

BERT is general purpose language model, pretrained on Wikipedia and BookCorpus. It leverages high amount of unannotated data on the web and produces context-aware word embeddings. The current solution uses pytorch-pretrained-BERT[2]. Besides fine-tuning BERT weights, the set of possible parameters for the pretrained BERT is limited. The possibilities are:

- Amount of layers in the Transformer model: 12-layer (bert-base) or 24-layer (bert-large)
- Cased or uncased model
- Multilingual or single-language model

One additional peculiarity, discovered by several contestants independently, is that using the output of the last layer may be an inferior option compared to the deeper ones. For the architecture presented in this paper, optimal layers were fifth-to-last ([-5]) and sixth-to-last ([-6]). Figure 1 shows the log loss during stage 1 for different ouput layers, keeping all other parameters of the network fixed.

One possible explanation for this phenomenon is that the last output layer specializes on predicting the masked words, while the intermediate layers contain more general information. The U-shaped curve also shows that taking much deeper layers negatively affect the model performance.

Fine-tuning. As mentioned in the section 2.1, initially only the head part was fine-tuned on the

[2]https://github.com/huggingface/pytorch-pretrained-BERT

warm-up dataset and the learned weights were used as initialization weights for the GAP training. The logical step would also be to fine-tune the BERT embeddings themselves. It was done in the following way: first, the head was trained for 16 epochs, with the parameters of BERT being frozen. Afterwards, all of the parameters of the head were frozen, and one particular layer (either fifth-to-last or sixth-to-last) of BERT was fine-tuned. The small learning rate was very crucial at this step, because the number of parameters is high (12,596,224). The current solution uses $4e^{-5}$ as learning rate, trained for 16 epochs with batch size of 32. Every 400 steps the network performance was estimated on the evaluation data, and only the best performing model was used after the training was done.

3.2 Head part

The head network always takes BERT contextual embeddings of shape [batch_size, seq_len, 1024] (for BERT-large). These embeddings are processed with the 1d-convolutional layer of size 64 and kernel=1 in order to reduce the dimensionality. Interestingly, increasing kernel size to 2 or 3 deteriorates the performance. This may be due to the fact that the context just around the mentions was not that informative.

Because the positions of mentions and pronoun are known in advance, the embeddings of only those three phrases are extracted. This is done by using SelfAttentiveSpanExtractor from AllenNLP (Gardner et al., 2017). This span extractor will generate 3 vectors of size 64 - for A, B and Pronoun. For single token mentions the span representation is just the original vector itself, while for mentions with more than two tokens, the span extractor will produce weighted representation by using the attention scores. Other span extractors from AllenNLP did not perform as good as the self-attentive span extractor.

The resulting three vectors of size 64 are concatenated and processed with the standard fully-connected block: BatchNorm1d(192) $\rightarrow$ Linear (192, 64) $\rightarrow$ ReLU $\rightarrow$ BatchNorm1d $\rightarrow$ Dropout (0.6). This output is concatenated with all the manual features mentioned in the section 2.3, which results in the vector of length 96. Adding features directly to the last layer is important, otherwise they do not bring any improvement. Finally, Linear (96, 3) layer on top produces the logits. The softmax probabilities are computed in the numerically stable way in the loss function.

3.3 Training details

For the GAP training, Adam optimizer with the learning rate $2e^{-3}$ is used. The batch size is 20. For both BERT fine-tuning and GAP training the triangular learning rate schedule is used (Smith, 2017). The loss used is CrossEntropyLoss, which combines numerically stable computation of softmax probabilities with negative log-likelihood loss function.

The predictions for each of the four networks are done in 10 fold cross-validation stratified by the class distribution, i.e. the model is trained on 90% of the data and the other 10% is used for the evaluation. The final predictions is the average across all folds and all models, overall of 40 models. Final predictions were clipped to be in the interval $(1e^{-2}, 1-1e^{-2})$, because log loss penalizes the predictions heavily as they drift away from ground truth.

The training runs approximately one day on single NVIDIA Tesla P100. This can be substantially reduced with proper code optimization (for instance, removing BERT computations for all layers after the fifth-to-last). The framework used for the implementation is PyTorch[3]

4 Results

The described solution provides the log loss of 0.1839 on the test stage 2 data, which results in the third place on Kaggle leaderboard. The results of single models are presented in the Table 3. As can be seen, the cased version performs generally better. One reason may be given by the variety of personal names in the GAP, and the cased version is able to recognize them better.

On the *cleaned* stage 1 test data, the best-performing single model (cased BERT, fifth-to-last layer) provided the log loss of 0.23819. Because the true labels are available for the test stage 1 data, the loss for the masculine and the feminine pronouns was estimated separately. For masculine pronouns the log loss was equal to 0.24014, while for feminine it was equal to 0.23623. The difference is $3e^{-3}$, which can be considered insignificant.

The error matrix for the whole stage one dataset (10-fold cross-validated) is presented in the Table

[3]https://pytorch.org/

	A pred	B pred	NEITHER pred
A	1849	72	26
B	57	1908	24
NEITHER	58	59	401

Table 2: Confusion matrix on the whole stage 1 data with 10 fold cross validation.

Model name	Log loss
Cased [-5]	0.20592
Cased [-6]	0.20429
Uncased [-5]	0.22834
Uncased [-6]	0.21088
Ensemble	0.18397

Table 3: Performance of the models on the test stage 2 data.

2. As can be seen, the performance for the category NEITHER is worse than for other categories. Most likely this is because of the lower amount of training examples. Even though the warm-up data contained many observations with this category, these examples were quite trivial, and the final model still struggles on the GAP data. This indicates a potential area for the improvement of the model. For the wrong predictions, some additional metrics were also examined, like the length of the text or the number of words in the mentions. These properties are not significantly different from those for the correct predictions. After manual examination, it appears that the model makes mistakes on the examples that are also quite challenging for humans.

5 Discussion

One of the weaknesses of the presented system is the training and prediction time. Because there are four networks, training takes a long time, and the prediction on the test set requires almost 2 hours. The attempt to concatenate several intermediate BERT layers, or to use a linear combination of them did not work, although it was reported to have a positive effect (Tenney et al., 2018). Using the information from the whole Wikipedia page also did not provide any improvements.

During the early stages of the competition, when only GAP data was used, the sources of model mistakes were analyzed. It was found, that despite the best efforts of the authors, there are still some mislabelled examples in the GAP data itself.

dataset	errors female	errors male
gap_development	35	31
gap_val	12	9
gap_test	45	28

Table 4: Number of mislabelled examples in the datasets per gender.

Other participants reported errors as well[4]. Some of these errors are quite simple, but the majority require substantial human efforts and sometimes were impossible to detect without reading the corresponding Wikipedia article.

The number of erroneous labels for different GAP datasets separated by gender is reported in Table 4. This list is based on the mistakes reported on the forum, as well as own single checks, but it is not comprehensive. It can be seen that mislabelled examples represent less than 5% of all cases. They are usually equally distributed between genders, besides gap_test, where mislabelled examples for female cases are 30% higher.

6 Conclusion

This paper presents the solution for the coreference resolution on GAP shared task. The solution utilizes the pretrained contextual embeddings from BERT and fine-tunes them for the coreference problem on additional data. One of the findings is that the output of BERT's intermediate layers gives better representation of the input text for the coreference task. Another contribution is that the gender bias in external data can be mitigated by using gender-fair datasets as validation data during the pretraining phase.

References

Hong Chen, Zhenhua Fan, Hao Lu, Alan L Yuille, and Shu Rong. 2018. Preco: A large-scale dataset in preschool vocabulary for coreference resolution. *arXiv preprint arXiv:1810.09807*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer.

[4]https://www.kaggle.com/c/gendered-pronoun-resolution/discussion/81331

2017. Allennlp: A deep semantic natural language processing platform.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 809–819.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in ontonotes. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 1–40. Association for Computational Linguistics.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: the winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789. Association for Computational Linguistics.

Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. Gender bias in coreference resolution. *arXiv preprint arXiv:1804.09301*.

Leslie N Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2018. What do you learn from context? probing for sentence structure in contextualized word representations.

Kellie Webster, Marta Recasens, Vera Axelrod, and Jason Baldridge. 2018. Mind the gap: A balanced corpus of gendered ambiguous pronouns. *Transactions of the Association for Computational Linguistics*, 6:605–617.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2018. Gender bias in coreference resolution: Evaluation and debiasing methods. *arXiv preprint arXiv:1804.06876*.