Gendered Pronoun Resolution

Dhruv Patel(14528)
IISc, Bangalore
dhruvpatel@iisc

Prateek Sachan(15754)
IISc, Bangalore
prateeksachan@iisc

Abstract

In this project we try solve the problem of Gendered Pronoun Resolution. To train neural networks with less data we show how augmentation can help to reduce overfitting. We have experimented with different architectures including and excluding attention. We observe that attention has no advantage over simple mean. Our model scores 0.2462 (cross entropy) on Kaggle competition, which is better than solution occupying 26th position on the leaderboard.

1 Introduction and Problem Statement

Gendered Pronoun Resolution is a subset of co-reference resolution problem. Here we study problem in a setting where we are given a particular male(he, his, him) or female(she, her, hers) pronoun and two candidate nouns in a sentence that match gender of the given pronoun. Task is to figure out which of these two candidates this particular pronoun refers to. Our architecture can be extended to more than two candidates without any modifications. This problem was posed on Kaggle as a competition by Google AI. Below we show one example sentence of the input. Boldface words are candidate nouns. Underlined noun is the correct noun. Italic boldfaced word is the pronoun.

> **Kathleen** first appears when **Theresa** visits ***her*** in a prison in London.

2 Related Work

Earlier co-reference resolutions approaches using neural networks were proposed by Wiseman et al. (2016) and Clark and Manning (2016). However they used syntactic parsers to hand engineer mention proposal algorithms. Current state-of-the-art model is proposed by Lee et al. (2017). In their model, they would consider all spans of some maximum length and calculate mention score for that span using a neural network. Than for each span i they would consider all spans j before it and calculate a score for pair (i, j). They also consider special span ϵ and fix score of (i, ϵ) to 0. ϵ span denotes that i is the first co-referent, and nothing before it refers to same entity as i.

To find embeddings of spans Lee et al. (2017) used BiLSTM, with GloVe (Pennington et al., 2014) embeddings as input. At the end, to calculate fixed size span embedding, they used attention mechanism to find weights and than used weighted combination of embeddings of words in the span. Later Peters et al. (2018) improved results of this architecture by using ELMo embeddings.

By using transformer network (Vaswani et al., 2017), Devlin et al. (2018) have trained BERT model to give contextualized embeddings. These embeddings have got state-of-the-art results in eleven NLP tasks.

3 Method

Earlier we started with ambitious goal. Instead of just two candidates, our earlier models used Flair embeddings to find all possible person entities as possible candidates using named entity recognition. However for each sentence we only knew one correct answer. There could have been other entities with some variation to correct name, which we would treat as negative example. It turned out that none of our models worked better with GAP dataset(section 4.1). So instead we focused on more simpler setting, where two candidates are given and at most one of them is true. Now our problem was reduced to three class classification.

Our starting point was Lee et al.'s model. First we get BERT embeddings for candidate A, candidate B and a pronoun. We used simple mean to get

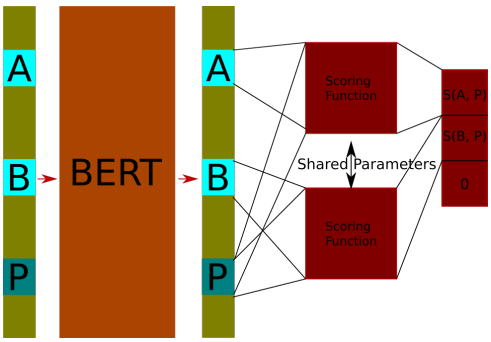

Figure 1: Architecture

fixed size vector for each candidate.

In their model they had to find mention score for each span. However in this simpler setting, since candidate spans are given, we modified their architecture to not use mention score. Now we only needed to find two scores, one for each possible candidates. We pass pair consisting a candidate and a pronoun through three layer fully connected neural network (called scoring function). Then we concatenate these two scores and 0 to get 3d vector. Here 0 is dummy fixed score for case when neither A nor B is correct answer. Probabilities of classes are given by softmax over this vector.

We have also experimented with bilinear variation of scoring function, but that didn't perform well. In one variation, we ran single layer LSTM on the output of BERT, and then we applied scoring function as above, on the output of LSTM.

However, while training we found that, since GAP is a small dataset, our models overfitted. Instead of looking for other datasets with similar characteristics of GAP, we used augmentation. Details of augmentation are explained in section 4.2. But in summary, each time network sees a sentence, with probability 0.6 it will see random different candidates instead of original candidates. Thus network will have no time to memorize candidates and instead it has to learn other useful features.

Figure 1 shows our final architecture with respect to which our results are reported in section 7.

4 Datasets and Metrics

4.1 Datasets

We have used two datasets in our experiments.

1. **GAP** GAP co-reference dataset (Webster et al., 2018) is gender-balanced dataset divided into three sets for development, test and validation. Both development and test sets contain 2000 sentences each. Validation dataset contains 454 sentences. In each sentence there are two possible candidates denoted by A and B respectively. There is one pronoun per sentence. This pronoun can refer to either A or B or neither.

2. **DPR** Definite Pronoun Resolution (Rahman and Ng, 2012) dataset is divided into two sets for training and testing. There are 1886 sentences in total. Although the original dataset has only one candidate per sentence, there are two sentences having same actors in common(i.e. there are 943 pairs of sentences). So we combined two actors to play as candidates. The resulting dataset is similar to GAP. Below is an example pair.

    - James asked **Robert** for a favor, but ***he*** refused.
    - **James** asked Robert for a favor, but ***he*** was refused.

We have used 2000 sentences from GAP to train, while keeping others aside for validation and test. When we used DPR in addition to GAP, we used both train and test sets of DPR for training. Validation in this case was still done on GAP validation set.

4.2 Data Augmentation

Earlier we tried our models without any data augmentation. But since GAP has only 2000 sentences in development set, our models overfitted. An SVM trained on these 2000 sentences outperformed neural network architectures that we tried. To compare our later modifications, we will use SVM as a baseline.

To mitigate the situation we applied data augmentation. Our hypothesis is that, since input to our network is just a pair of candidate nouns, it doesn't matter what these nouns are. If all occurrences of noun 'Firstname Lastname' were to be replaced by some other plausible pair of first name and last names, sentence should make perfect sense. To neural network "Jon Snow doesn't know anything." should be similar to "John Wick doesn't know anything."

To augment data we applied simple rule. If both candidate A and candidate B had less than four words then with probability 0.6 we would pick

random noun with same number of words. That is if A had three words and B had two words, we would pick alternative A and B with three words and two words respectively. If pronoun is male(female), then only male(female) names are proposed as alternatives. Alternative name for B was chosen such that no word of it was a substring of alternative A. Also none of the alternatives had any overlap with original nouns. Below are the examples of augmented sentences. First sentence is an original, other are augmented. Here Margaret Ray is candidate A and Betsy is candidate B.

- Tony Markham, a high school senior and the "Tall Dark Stranger" Betsy fell in love with as a freshman, who has since become a good friend not only to Betsy but the entire Ray family. Mrs. Ray, Betsy's mother. Mr. Ray, Betsy's father, who owns a shoestore. **Margaret Ray**, **Betsy**'s sister who is five years younger than she is.

- Tony Markham, a high school senior and the "Tall Dark Stranger" Booth fell in love with as a freshman, who has since become a good friend not only to Booth but the entire Delgado family. Mrs. Delgado, Booth's mother. Mr. Delgado, Booth's father, who owns a shoestore. Pam Delgado, Booth's sister who is five years younger than she is.

- Tony Markham, a high school senior and the "Tall Dark Stranger" Alyssa fell in love with as a freshman, who has since become a good friend not only to Alyssa but the entire Jolie family. Mrs. Jolie, Alyssa's mother. Mr. Jolie, Alyssa's father, who owns a shoestore. Angelina Jolie, Alyssa's sister who is five years younger than she is.

The pool for alternative names was extracted from dataset for stage2 of Kaggle competition. It has around 12K sentences. Figure 2 shows distribution of names for both genders. One word names are most common in dataset followed by two word names.

4.3 Metrics

To compare our results with baseline proposed by Webster et al. (2018), we use micro average of F1-scores. To compare our results with other Kaggle competitors, we used cross entropy loss $\mathcal{L}$.

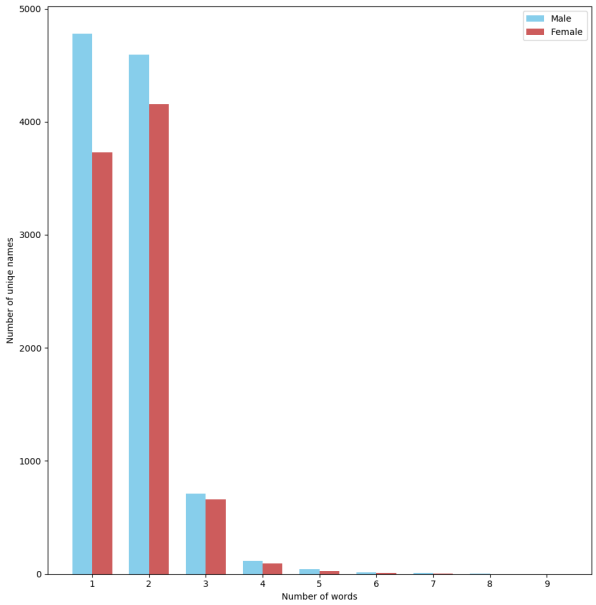

Figure 2: Distribution of names.

	M	F	O
Wiseman et al. (2016)	68.4	59.9	64.2
Lee et al. (2017)	67.2	62.2	64.7
SVM-8	**77.10**	**79.10**	**78.10**

Table 1: Baselines

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j\in\{A,B,N\}}(y_i^j * log(\sigma(\hat{y_i^j}))).$$

Where y_i^j is 1 if j is correct candidate for i^{th} example. N denotes neither case. $\hat{y_i^j}$ denotes predicted probability for class j for i^{th} example.

5 Baselines

Table 1 shows baselines we have used. First two rows were reported by Webster et al.(2018). **M** column denotes F1 score for male, same for female and **O** denotes score for all. However these baseline models were not trained on GAP dataset[1]. For example Lee et al. (2017) was trained on onto notes.

In last row we note the baseline that we established. That score was obtained by SVM trained on 768×3 dimensional vectors. These vectors

[1]Authors have reported results on development set and not test set. So to be fair, we have trained our models actually on test set and reported results on development set. On Kaggle too published test set was actually development set in Github repository

were obtained by concatenating outputs of 8th layer BERT(base model) for A, B and P. Where P denotes a pronoun.

6 Experiments

We have ran different experiments. In one set of experiments we varied different layers of BERT base model and BERT large model. There were $24 + 12$ different experiments done. Initially for base model, our best score(cross entropy) was 0.59, which was obtained by 8th layer and scoring function explained in section 3. This model was unable to pass the baseline that we had established (i.e. SVM). The cross entropy of SVM-8 is 0.51.

Our scoring function is multi layer perceptron with three linear layers with ReLU activation. First layer has 64 units, second layer has 32 units and final layer has one output unit. We pass $[C(V(A))|V(P)]$ and $[C(V(B))|V(P)]$ through this function and obtain scalar score. Here $V(w)$ is an output of certain layer BERT for token w. When w is a list of tokens $V(w)$ is a list containing output for every input token. Since BERT uses WordPiece tokenizer, sometimes even one word noun gets tokenized into multiple tokens. C function combines these multiple vectors into single fixed size vector. We have used two different types of C. In one case we just take a mean of all these vectors. That is bag of tokens. In other model we learn attention on these tokens and then combine proportional to the attention. We used Multiplicative attention.

For LSTM, we used single layer with 256 hidden units. In these experiments, output of LSTM was passed to scoring function with two linear layers, with layer of size 32 removed.

We have implemented our models in PyTorch. We used pretrained BERT models from Hugging Face[2]. We used Adam optimizer with default parameters. We used patience value of 20, that is if no improvement in validation loss happened within 20 epochs, we would stop training. Weight decay of 0.001 was used. Dropout value of 0.5 was used. The best results were obtained by 20th layer of BERT large model.

	M	F	O
SVM-8	77.10	79.10	78.10
MLP	89.10	88.00	88.55
MLP-attn	89.50	88.40	88.95
MLP-dpr	88.90	88.70	88.80
MLP-dpr-attn	90.10	87.90	89.00

Table 2: F1 score Results

	M	F	O
SVM-8	0.5127	0.5077	0.5102
MLP	0.2669	0.3412	0.3041
MLP-attn	0.2828	0.3252	0.3040
MLP-dpr	0.2752	0.3187	0.2969
MLP-dpr-attn	0.2706	0.3367	0.3036

Table 3: Cross Entropy Loss on stage 1 test set

7 Results

Table 2 shows results on our models. When model has `-attn` suffix, model used attention instead of simple mean. When model has `-dpr` in it's name we have also used DPR dataset in addition to GAP dataset. We can see that there is no discernible difference in overall F1-score. We discuss implications and our hypothesis in section 8.

Table 3 shows cross entropy loss on Kaggle test set of stage 1. Other models are performing better. In this test set there are 2000 examples.

Table 4 shows cross entropy loss on Kaggle test set of stage 2. This test set contains around 12K examples. Since correct answers are not in public domain yet, we obtain these results by uploading our results on Kaggle competition page. Here we see that our SVM baseline fails miserably. However other models performs good. For reference we also list results of composition winner and 25th

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

	O
SVM-8	0.7809
RNN-MLP	0.3529
MLP	0.2462
MLP-attn	0.2545
MLP-dpr	0.2727
MLP-dpr-attn	0.2517
Kaggle First	0.1366
Kaggle 25th	0.2403

Table 4: Cross Entropy Loss on stage 2 test set

	precision	recall	f1-score	support
A	0.8635	0.9533	0.9062	856
B	0.9219	0.8930	0.9072	925
Neither	0.8113	0.5890	0.6825	219

Table 5: Precision-Recall of MLP

spot to compare our position. This table also includes result on RNN. Using LSTM didn't improve results.

Table 5 shows Precision Recall for different classes on MLP model(2nd row in Table 3) on test set for stage 1. It is interesting to note that for class "Neither" recall is 0.58, which is too low. That is our model will say, either A or B often, even though neither of them is true. We believe this is due to class imbalance in training set. Assigning different weights to different classes in loss function could mitigate this situation, but we haven't tried that approach yet.

8 Discussion

8.1 Difference(or lack of) between attention and mean

Looking at results in table 3, we don't see discernible difference between attention mechanism and simple mean mechanism. To analyze further we compared cosine similarity of the vectors generated after both mechanisms. About 23% vectors from validation set had cosine similarity less than 0.999. And none of the vectors had cosine similarity less than 0.99. This implies that the learned attention was close to uniform. To confirm this, we manually analyzed few random attention weights and they were close to uniform. Although We don't know the side effects of WordPiece tokenization, our hypothesis is that, looking at Figure 2, most nouns are single word or double word long. Thus resulting tokenization would not be too long either. And so attention has no advantage in this case.

8.2 Bias

Again, looking at table 4, columns **M** and **F**, model MLP has most bias, even though it is best performing model on stage 2 test set. We expected there to be more bias in last two models than first two, since they were trained on DPR dataset as well, which has unbalanced pronouns. GAP dataset on the other hand is balanced dataset. We think this bias stems from BERT embeddings and not from the dataset.

8.3 Fine tuning BERT

Even after augmentation, we think that it is not a good idea to fine tune BERT on GAP dataset. Although this may reduce bias, we are afraid that model might learn to remember answer by looking at the context. Augmentation happens only on nouns, context doesn't change. So it might be possible to remember answer based on context.

8.4 Extending to more than two candidates

Although during training we have two candidates, there is nothing stopping us to have more than two candidates during test time. That is since parameters for scoring function in figure 1 are shared, we can have final vector of arbitrary size. The next step is softmax over this vector. It doesn't matter the dimensions of this vector, softmax essentially selects top scoring element from input vector. However we were unable to combine NER from Flair and our model, since both use different tokenization. To combine, we would have had to make few changes in our preprocessing code as to incorporate different tokenizations.

9 Future Work

For test stage 2, many top ranking users on Kaggle actually retrained their model with 4000 sentences. That is they also included test set of stage 1 during training. First task for future would be to train our model on combined dataset and observe the effects.

We believe, once test set for stage 2 becomes available, we can fine-tune BERT model also. Or we can induce BiLSTM in between BERT and MLP.

As noted in section 8.4, it would be interesting to check our hypothesis that our architecture should work on more than two candidates.

References

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262, Austin, Texas. 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. *arXiv preprint arXiv:1810.04805*.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proc. of NAACL*.

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.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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.

Sam Wiseman, Alexander M. Rush, and Stuart M. Shieber. 2016. Learning global features for coreference resolution. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 994–1004, San Diego, California. Association for Computational Linguistics.