Transformers for Coreference Resolution

Ryan Elliott
ryelliot@calpoly.edu
California Polytechnic State University
San Luis Obispo, California

ABSTRACT

Coreference resolution systems notoriously struggle resolving long-distance referents. Further, state-of-the-art systems are unfit to achieve human-level accuracy, unable to surpass an F1 score of 85 on the CoNLL-2012 shared task. The transformer neural architecture is designed to process an entire document in parallel, thereby lending itself kindly to resolving distant referents. Using this, we apply a transformer architecture alongside an experimental transformer-pointer network to achieve 99.3% validation accuracy on the PreCo Coreference Resolution dataset.

KEYWORDS

coreference resolution, transformer neural network, language modeling

ACM Reference Format:
Ryan Elliott. 2020. Transformers for Coreference Resolution. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 7 pages. 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Transformer networks have not only advanced the state-of-the-art in machine translation tasks in recent years, but also demonstrated their versatility in various natural language processing (NLP) tasks. Transformers are capable of modelling long-distance relationships between independent word tokens in expansive documents of text – an imperative strength for coreference resolution systems to possess. Similar state-of-the-art transformer models struggle with semantic word relationships over extended distances, a fault the transformer has the strengths to solve.

Consider an extended text document similar to a short story. Often times, characters are introduced in the beginning of the document, not to be mentioned again until half-way through the story. Humans are able to easily recognize these near-singleton-like characters, subconsciously creating an internal graph of character relations, mapping character names to referential pronouns. Transformer networks possess the unique ability to process an entire document of text in parallel – through the power of positional encoding – and have reduced the reliance upon memory loss prone gated recurrent neural networks.

We apply the encoder-decoder transformer model, while experimenting with an encoder-linear transformer and transformer-pointer network to coreference resolution, finding a subset of the architectures are able to model a simplified coreference resolution problem limited to 200 tokens. BERT word embeddings [4] have been found to achieve substantial performance improvements in coreference resolution, but are known to struggle resolving entity relations that require world-based (opposed to semantic-based) knowledge. While our proposed model does not resolve this issue, we opted to use learned embeddings to determine if this pitfall can be avoided via a full encoder-decoder resolution system.

The utilization of an encoder-decoder transformer was a logical choice. Transformer networks have shown remarkable results in machine translation tasks, and in an abstract manner, coreference resolution itself can also be framed in a machine translation problem mold. It's easy to view the task of coreference resolution in this frame, especially when the target indices have been encoded through an identical process as sentences in an arbitrary language. The source language stands as English, while the output language is the encoded indices of a token's referents.

2 RELATED WORK

We give an overview of the three most common model paradigms in coreference resolution: mention-ranking, entity-based, and recurrent-based models. Each of the models aim to solve a unique problem propagated by another, all the while unable to fully solve the coreference resolution task.

2.1 Mention-Ranking Models

Recent advances in span-scoring-based models have shown great potential in achieving further state-of-the-art results. In these score-based models, the typical process involves the generation of a query for each candidate mention resolution using surrounding semantic context. These queries are then compared against the key span, wherein the most similar span is selected as coreferent. Ultimately, these models extract mentions from a given document, create an expansive ranking of each mention against others, and select the highest ranking mention as the referent for a given mention.

2.2 Entity-Based Models

Entity-based models resolve the issue of mixing two or more unique referents. Consider a document summarizing court proceedings containing three entity mentions, "*he*", "*Mrs. Rodgers*", and "*Rodgers*". A mention-ranking model is prone to labeling "*he*" and "*Rodgers*" as coreferent, and further labelling "*Mrs. Rodgers*" and "*Rodgers*" as coreferent, creating an implication stating that "*he*" and "*Mrs. Rodgers*" are the same entity, where in reality the document's context informs the reader that "*Rodgers*" alone refers to "*Mrs. Rodgers*"'s spouse, while "*Mrs. Rodgers*" and "*she*" are indeed coreferent. Recent papers employ models designed to circumvent these issues, finding excellent results in euclidean-near entities as demonstrated by Lee et al. [6], however the models tend to struggle resolving long-distance references.

These entity-based models are perhaps the most widely used paradigm in coreference resolution. Many state-of-the-art systems utilize a model similar to that found in a paper from Lee et al. [5], modifying either the encoding or decoding modules. The base model utilized in the aforementioned paper belongs to the same family as many previous state-of-the-art models, also branching from an earlier system by Lee et al. [6]

2.3 Recurrent-Based Models

Another popular coreference resolution paradigm incorporates bidirectional long-short term memory (Bi-LSTM) networks. Perhaps most influentially, the model proposed by Joshi et al. [9], which utilizes BERT word embeddings in conjunction with a Bi-LSTM, took language models in coreference resolution a step further; the authors implemented the BERT transformer into an encoder-decoder coreference resolution network, effectively replacing the previous LSTM-based encoder which utilized ELMo and GloVe embeddings [7] [8] as the proxy between raw tokens and the network. While this work took the step to implement a transformer in a coreference resolution network, the authors did not attempt to replace the inner LSTM architecture with a transformer.

Even further, the original transformer paper by Vaswani et al. [1] notes that the transformer alone performs a niche type of anaphora resolution, illustrating the relation between the two words "*Law*" and "*application*" being used to generate the word "*its*" which is correctly referent to both "*Law*" and "*application*". The authors tracked the activation's in attention heads to find this correlation, illustrated in the paper's appended attention visualization section.

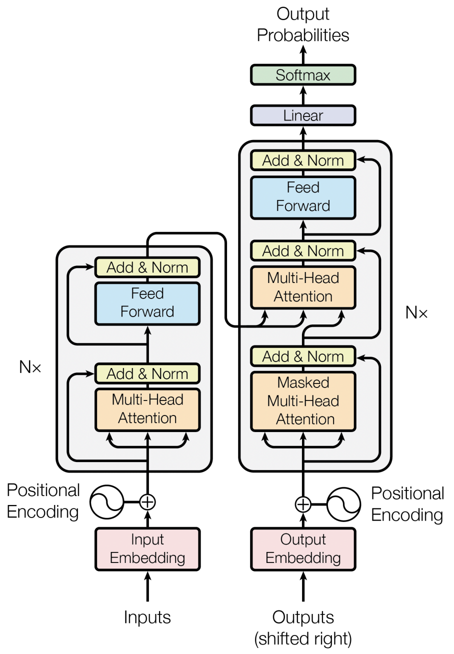


Figure 1: Transformer network, as depicted by Viswani et al. [1]. The network is composed of an encoder and decoder, both of which are preceded by word embeddings transformed by a positional encoder. This architecture allows for the processing of text in parallel with the use of token masking.

3 SYSTEM

Most every state-of-the-art coreference resolution system makes use of an encoder-decoder structure. Our gold model follows in similar footsteps.

3.1 Encoder-Decoder Transformer

The gold model proposed in this paper was trained using an encoder-decoder architecture, assuming the shape of a relatively standard full transformer network. The transformer network accepts token IDs as input, converting these IDs into learned word embeddings using a standard 768 neuron embedding layer [1]. Distinctly different, the encoder and decoder learn separate embeddings from each other. The source text was converted from raw strings to token IDs independently of the target "text". In reality, the target texts are indices into the source document (see Figure 2), simply constructed into a standard vocabulary to allow for consistent training.

3.1.1 Positional Encoding. In both the encoder and decoder, the word embeddings also underwent a positional encoding transformation, encapsulating crucial information regarding the positioning of the source and target inputs. This positional encoding represents the full power of a transformer processing an entire document in parallel; the ability to embed a token's positional information into the word embedding itself allows the encoder to provide context up to a given word while the decoder is able to understand what tokens it has seen at a given point, despite the parallelization of the data. The positional encoding $\overrightarrow{p_t}^{(i)}$ is given by a function, $f(t)^{(i)}$, where d stands as the embedding dimensions and $t \in \mathbb{R}^d$ denotes the index:

$$\overrightarrow{p_t}^{(i)} = f(t)^{(i)} := \begin{cases} sin(\omega_k \cdot t), i = 2k \\ cos(\omega_k \cdot t), i = 2k + 1 \end{cases} \tag{1}$$

$$\omega_k = \frac{1}{5000^{\frac{2k}{d}}} \tag{2}$$

This positional encoding layer feeds to the respective encoder or decoder, as depicted in Figure 1. All encoder/decoder hidden layers are composed of 768 neurons in the feed-forward layers, fed by 12 attention heads. Both the encoder and decoder modules are repeated 4 times. The decoder finally leads to a linear projection layer producing a log softmax probability distribution over the label (target) vocabulary.

3.2 Encoder Transformer

We tested the feasibility of strictly using a transformer encoder leading directly to a linear projection layer throughout development. The rationale for this was mainly driven by the increased computational speed; omitting a decoder greatly increased batch through-put – a large issue throughout training. While the results from this architecture were not nearly as promising as the encoder-decoder transformer, the data we did gather proved the viability of this architecture on limited machines such as embedded systems.

[1]Both the encoder and decoder utilized an embedding layer.

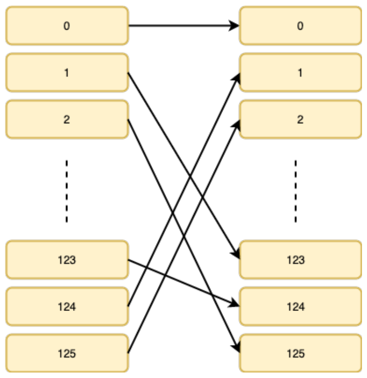


Figure 2: Mapping from indices to target IDs, depicting the non-linear nature of the target vocabulary representation. An identical process to the source mapping, each word (in reality an integer index) receives a unique integer id to be used throughout training and inference. These IDs are used by the model to produce word embeddings.

3.3 Transformer-Pointer Network

The leading model that was supposed to be proposed in this paper is an architecture that uses a slightly truncated encoder-decoder transformer, wherein the final attention layer in the transformer's decoder leads to a pointer network decoder, allowing the model to refer (point) directly back to the referential token(s). The failed architecture plan for this model follows the typical transformer model flow leading up to the decoder's mutli-head attention layer. At this point, a pointer neural network's decoder consumes the attention values and performs principal component analysis (PCA) – or a derivation of PCA – on the attention values to directly produce references to the system's input.

This architecture was never completed, as the multi-head attention mechanism found in the transformer architecture proved incompatible with the standard pointer network. Admittedly, this issue can be resolved, however, we do not possess the resources necessary to convert and normalize a multi-head attention output into a conventional self-attention layer to be used in decoding.

4 EXPERIMENTAL DESIGN

We utilized the PreCo Coreference Resolution dataset [3], processing the data to smooth the integration with a transformer network.

4.1 Pre-processing

The data underwent fairly thorough pre-processing; the raw dataset contains 37,600 documents, each document ranging in size from 500 tokens to slightly under 1,400 tokens. In total, the data, test, and validation sets contain 12.1 million cumulative tokens.

Our compute power (and time) were not capable of processing this much information in a timely manner. We were forced to truncate each document to a maximum of 200 tokens, reducing the dataset by ~71.4%. This size reduction took place after a week of training efforts, ultimately finding the network was unable to model relations appearing later in documents, most significantly unable to model past the 800 token mark.

4.2 Sources and Targets

While this implementation of a coreference resolution model does not incorporate anything novel in terms of source vocabulary and target vocabulary, the clarification between types is imperative. The source document is first converted from raw tokens parsed via the SpaCy library, then converted to token IDs, as depicted in Figure 3. These token IDs are then used to create word embeddings.

The expected targets are created in a very similar manner with one distinct difference: rather than raw words converted into token IDs, we convert the raw *coreferent indices* into token IDs, as depicted in Figure 2. This distinction disallows easy conversion to a transformer-pointer network.

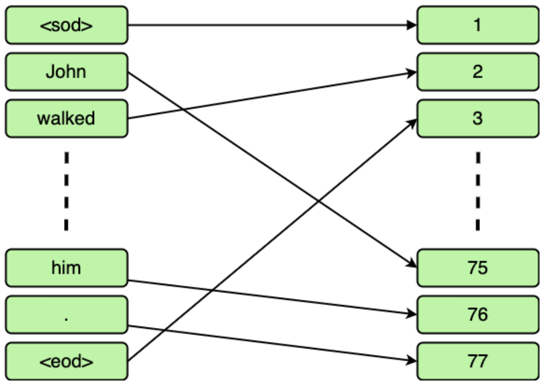


Figure 3: Mapping from source words to token IDs, depicting the non-linear nature of the target vocabulary representation. Each word receives a unique integer id to be used throughout training and inference. These IDs are used by the model to produce word embeddings.

4.3 Dataset Issues

The PreCo dataset is relatively new, not yet finding its footing in research. Furthermore, as is natural in coreference resolution, most non-singleton entities have multiple referents. While this is usually beneficial to a coreference resolution system, our unique model proved unable to take full advantage of the multi-referential nature of language. Each token is mapped to exactly one single token in a transformer model. This means, in the example document,

John walked his dog to the park. His friend, Billy, played tag with him.

the token "*his*" (2) refers to "*John*" (0), "*His*" (8), and "*him*" (16), meaning the model *should* be able to model "*his*" (2) as referring to indices 0, 8, or 16. In reality, the transformer model architecture forces us to decide a single correct token for the model to map to, typically the previous referent. In this example, the target token would be "*John*" (0).

5 EVALUATION

We trained on a truncated version of the PreCo dataset consisting of 36,620 documents, using a batch size of 1 on the gold model, and validation size of 500 documents.

Table 1: Results from tested models.

Model (Encoder-)	Epochs	Val. Loss	Val. Accuracy
-Linear	10	3.12	45.6%
-Decoder Ptr. [2]	10	-	-
-Decoder	15	**0.037**	**99.3**

Table 2: Hyperparameters associated with each model.

Model (Encoder-)	Epochs	Hid. Dim.	Hid. Layers	Attn. Heads
-Linear	10	512	6	8
-Decoder Ptr. [2]	10	-	-	-
-Decoder	15	768	4	12

5.1 Impact of Batch Size

Training a model of this size on such an extensive dataset proved computationally expensive, bringing to light the limitations of RAM allotment on GPUs. Setting aside the issue of parallelizing training data, the network was unable to model coreferent tokens on large batch sizes.

5.1.1 Analysis of the Dataset. The vast majority of tokens in a given document are not coreferent. Further, of the tokens that are coreferent, 50.8% are singleton mentions. This vast data imbalance was initially glossed over, but later presented problems during training. We began training using batch sizes varying between 4 and 8, however we found that the network was unable to learn, converging at a training loss of 3.1. After extensive investigation and debugging, we soon found the reasoning for this performance: the gradients for an index that is coreferent in one example were being "out-weighed" by the other training examples in the batch whose indices were not coreferent in that same index. This artificial masking of true error prevented an accurate gradient propagation across the network. Theoretically, decreasing the learning rate should resolve this issue, but our experiments found decreasing the learning rate past 0.0001 $(1x10^{-4})$ per step prevented the model from converging in a reasonable time.

5.2 Results

The gold model trained and presented in this paper is a full encoder-decoder transformer, as described in Section 3.1. The results depicted in the table are accurate to the PreCo dataset, but likely are not extractible to true coreference resolution systems, especially not systems trained on the CoNLL-2012 shared task. As mentioned previously, this dataset is weighted against non-referent tokens. Whenever the model predicts a token isn't referential, it's accuracy is likely to increase. Larger evaluation tasks utilized in state-of-the-art systems are designed in a manner such that non-referent tokens are not marked as true positives if correctly predicted. Rather, they will only be marked as false positives if incorrectly predicted, an objectively stronger, more comprehensive metric than that used in these experiments. An example of the model's output compared to the expected output can be found in the appendices.

[2]Not accurately implemented to produce results.

5.3 Reproducibility

While our encoder-decoder system was able to achieve 99.3% accuracy on the validation set, we were unable to reproduce these results on custom input sentences. We assured the accuracy of the results by reproducing the model's performance using a standard seed and manually reviewing the model output versus the expected results, among other metrics. Throughout debugging we found a couple key issues with the model, talked about briefly, prior to finding the effective reason for our issues.

5.3.1 Data Leak in Transformer. This was, in fact, the original root of our inability to reproduce the results. The targets fed into the transformer were parallel to the results we were asking of the transformer, essentially reducing the problem (during training) to a copying task. This was promptly corrected, shifting the targets down an index by omitting the "<sod>" token. With this, the model was correctly aligned, meaning the first word in the document received the target's "<sod>" token as input, preventing the copy task.

5.3.2 Incorrect Token Masking. A key step in training a transformer is masking future tokens. Transformer networks process an entire document in parallel, so ensuring the model cannot "see" future tokens at an earlier timestep is imperative to training a model capable of performing on new data. This task differs from the token shifting; to perform token masking, we create an upper triangular matrix, with the diagonal offset by +1, filled with $-\infty$ values as seen in Figure 6, effectively reducing the attention in those indices to some negligible value. While we were unable to find any issues with our token masking, we suspect masking the source sequence (a transformation which we did not perform) may improve the real-world results.

5.3.3 Initial Inaccuracies. Debugging led to the unearthing of the root issue: inaccuracies in the initial predictions propagated through the extended document. We found through experimentation that using the "true" labels through inference leads to effective final model output. The below document and extended vectors shown in Equation 3 depict this relationship, outlining an example of how feeding correctly masked tokens into the model results in valid output.

John walked his dog to the park. He ran into his friend, Ben.

$$\overbrace{\begin{bmatrix}0\\4\\0\\7\\4\\9\\8\\4\\5\\4\\4\\11\\4\\11\\4\\3\end{bmatrix}}^{Target} \longrightarrow \overbrace{\begin{bmatrix}2\\31\\4\\4\\7\\7\\9\\8\\4\\7\\4\\9\\11\\9\\11\\4\\4\end{bmatrix}}^{Output} \longrightarrow \overbrace{\begin{bmatrix}<sod>\\23\\<nr>\\<nr>\\3\\3\\5\\6\\<nr>\\3\\<nr>\\5\\8\\5\\8\\<nr>\\<nr>\end{bmatrix}}^{Raw\,Indices} \longrightarrow \overbrace{\begin{bmatrix}-\\-\\-\\-\\dog\\dog\\the\\park\\-\\dog\\-\\the\\he\\the\\he\\-\\-\end{bmatrix}}^{Referential\,Tokens} \underbrace{\begin{bmatrix}John\\-\\John\\dog\\-\\the\\park\\-\\his\\-\\-\\he\\friend\\-\\friend\\-\\-\end{bmatrix}}_{Expected\,Tokens} \tag{3}$$

Evidently, the model's output, even with the correct targets available, is still not indicative of the training and validation sets' accuracies. Unfortunately, these findings have cast more doubt on the results gathered through our experimentation.

6 CONCLUSION

We found the encoder-decoder transformer network applied to coreference resolution is feasible, however its real-world applicability is still questionable. In our testing and training environment, the network successfully modeled a simplified coreference resolution problem limited to 200 tokens on an extensive dataset, spanning 7.3 million tokens. The model's performance was considerably greater in the beginning of documents, degrading in performance as the document length grew.

Given our inability to reproduce the experimental results by feeding in new sentences for the model to perform inference upon, we would like to reiterate that the validity of these results cannot be proven, at least with the prediction loop we implemented. Despite this shortcoming, we are of the belief that transformer networks are capable of performing on a coreference resolution task.

7 FUTURE WORK

Moving forward, we would like to modify the data pre-processing and system architecture. Notably, we believe there are substantial improvements to be gained when multiple correct references are allowed per token, as discussed in Section 4.3. Allowing the model to learn with multiple correct answers would require significant adjustments to the system's architecture. Such adjustments would likely lean kindly into a full implementation of a transformer-pointer network.

7.1 Multi-Referents

The most glaring issue we've found to be impacting the results from this model is our data pre-processing methods. As noted in Section 4.3, we're drastically limiting the model's ability to generalize to any sentence structure by limiting the correct referents per token to a single index. While the route to fix this issue is not clear, we believe it can be alleviated via an intermediary loss function; rather than using standard cross entropy on a single expected tensor, it may be beneficial to have an intermediary function that performs token matching, pulling from acceptable output tensors to create a modified expected tensor. In this way, the model can correctly predict one of many acceptable target IDs for a given token. The expected tensor would transform into an expected matrix of shape $(Num\,Tokens, Max\,Refs)$.

More concretely, assuming the presence of the following document in training data:

John walked his dog to the park. His friend, Billy, played tag with him.

The expected tensors may assume a shape of $(17, 4)$ and be composed of the raw referent indices as tokens rather than mapped token IDs.

$$Y = \begin{bmatrix} 0 & 1 & 8 & 16. \\ <nr> & <nr> & <nr> & <nr> \\ 0 & 1 & 8 & 16 \\ 2 & <nr> & <nr> & <nr> \\ <nr> & <nr> & <nr> & <nr> \\ 5 & 5 & 6 & <nr> \\ 6 & 5 & 6 & <nr> \\ <nr> & <nr> & <nr> & <nr> \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4}$$

7.2 True Transformer-Pointer Network

The lack of a transformer-pointer network implementation was the largest shortcoming of this research. As briefly touched on in Section 3.3, this failure stemmed from our inability to convert the transformer's multi-head attention layer into an attention layer compatible with a standard pointer network. One resolution to this problem involved digging even deeper into the internal structure of a transformer, modifying the decoder module to encourage compatibility with a pointer network. Another, more practical resolution relates to the encoder-linear transformer model proposed in this paper.

Rather than creating a model that is 3/4 transformer network and 1/4 pointer network, creating a firm divide between the two models may improve performance, debugging, and modularity. Applying this, we would like to develop a model utilizing a transformer's encoder and a pointer network's decoder. However, we have one main hesitation when it comes to implementing a network designed in this manner: transformer networks are so powerful not only due to their encoder, but also their decoder and the token masks. Removing this capability from transformers in problems that do not require decoding has shown negligible performance degradation, oftentimes even improving performance. Further, we are uncertain whether a pointer network is able to be parallelized in this manner, although it's an endeavor we wish to pursue.

7.3 Batch Size

The final improvement we believe the model would benefit from encompasses batch size tuning. As noted in Section 5.1, the model faced difficulty converging on large batch sizes, greatly increasing training time while reducing performance. Batch size has been studied extensively, with the general conclusion finding that reducing batch size improves generalization capabilities in models [11]. However, recent studies have shown opposing results, as presented by Smith et al. [10], advocating for the avoidance of learning rate decay in favor of increased batch size. This most recent study appears to directly oppose our personal findings on a sparse dataset. We are unable to make conclusions regarding the validity of our counter-results against Smith et al. [10], but we feel there is room for further research in efforts to find the optimal batch size to learning rate ratio.

7.4 Multi-Lingual Models

Taking future work a step further, we would also like to experiment with creating multi-lingual-capable models. Taking inspiration from a paper showcasing machine translation capabilities [12], we would like to experiment with adding a language token at the beginning of the source sentence and targets. An immediate challenge we face by proposing a task like this is the lack of datasets; as it stands, coreference resolution datasets are fairly sparse, mainly attributed to the fact that they all require extensive manual labelling. While datasets for varying languages do exist, they are not nearly as exhaustive as the PreCo dataset. In spite of this, it may be feasible to train an English coreference resolution model to later fine tune using language tokens.

REFERENCES

[1] Vaswani Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need *arXiv preprint arXiv:1706.03762v5*, 2017

[2] Oriol Vinyals, Merire Fortunato, and Navdeep Jaitl. Pointer Networks *arXiv preprint arXiv:1506.03134v2*, 2017

[3] Hong Chen, Zhenhua Fan, Hao Lu, Alan L. Yuille, and Shu Rong. PreCo: A Large-scale Dataset in Preschool Vocabulary for Coreference Resolution *arXiv preprint arXiv:1810.09807*, 2018

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding *arXiv preprint arXiv:1810.04805v2*, 2019

[5] Kenton Lee, Luheng He, Luke and Zettlemoyer. Higher-order Coreference Resolution with Coarse-to-fine Inference *arXiv preprint arXiv:1804.05392*, 2018

[6] Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. End-to-end Neural Coreference Resolution *arXiv preprint arXiv:1707.07045*, 2017

[7] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep Contextualized Word Representations *arXiv preprint arXiv:1802.05365*, 2018

[8] R. Jeffrey Pennington and C. Manning. Glove: Global Vectors for Word Representation. 2014.

[9] Mandar Joshi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. BERT for Coreference Resolution: Baselines and Analysis *arXiv preprint arXiv:1908.09091*, 2019

[10] Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't Decay the Learning Rate, Increase the Batch Size *arXiv preprint arXiv:1711.00489*, 2018

[11] Dominic Masters, and Carlo Luschi. Revisiting Small Batch Training for Deep Neural Networks *arXiv preprint arXiv:1804.07612*, 2018

[12] Melvin Johnson, Mike Schuster, Quoc V. Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation *arXiv preprint arXiv:1611.04558*

[13] Juntao Yu, Alexandra Uma, and Massimo Poesio. A Cluster Ranking Model for Full Anaphora Resolution *arXiv preprint arXiv:1911.09532*

[14] Michael Strube and Christoph Müller. A Machine Learning Approach to Pronoun Resolution in Spoken Dialogue *Association for Computational Linguistics 10.3115/1075096.1075118*

[15] Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. Context-Aware Neural Machine Translation Learns Anaphora Resolution *arXiv preprint arXiv:1805.10163*

[16] Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. Learning Generic Sentence Representations using CNNs *arXiv preprint arXiv:1611.07897*

[17] Kevin Clark. Neural Coreference Resolution

[18] Ruslan Mitkov. Robust Pronoun Resolution with Limited Knowledge *Association for Computational Linguistics 10.3115/980691.980712*

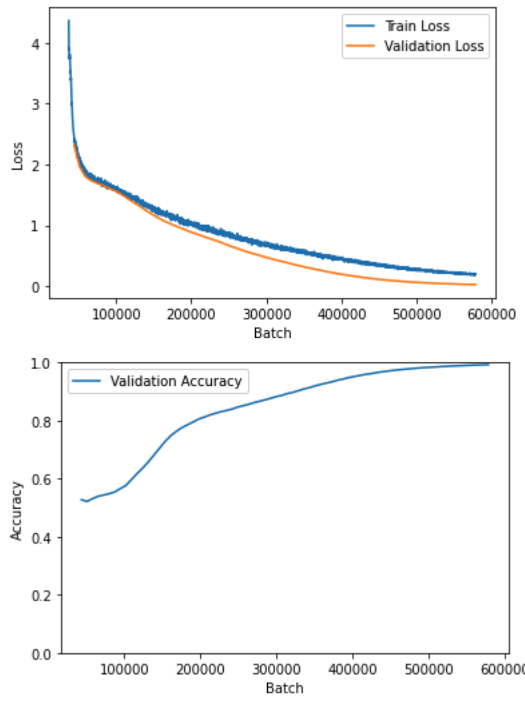
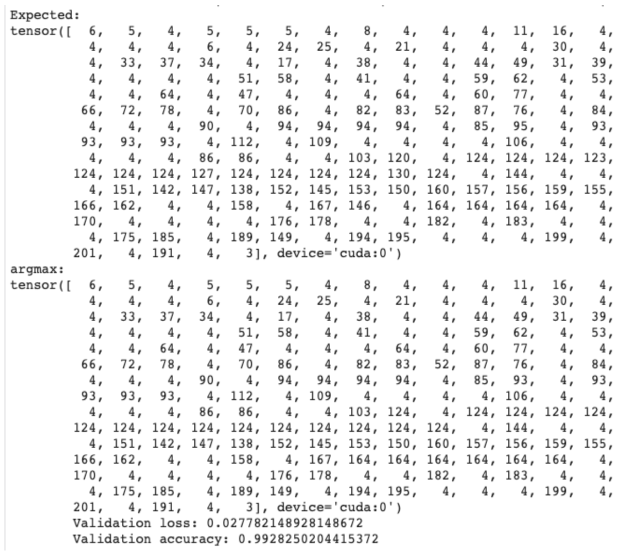


Figure 4: The top graph displays the the training vs validation loss over batch number. The model was trained on 15 epochs, each epoch containing more than 36,000 training examples.



```
Expected:
tensor([  6,   5,   4,   5,   5,   5,   4,   8,   4,   4,   4,  11,  16,   4,
          4,   4,   4,   6,   4,  24,  25,   4,  21,   4,   4,   4,  30,   4,
          4,  33,  37,  34,   4,  17,   4,  38,   4,   4,  44,  49,  31,  39,
          4,   4,   4,   4,  51,  58,   4,  41,   4,   4,  59,  62,   4,  53,
          4,   4,  64,   4,  47,   4,   4,   4,  64,   4,  60,  77,   4,   4,
         66,  72,  78,   4,  70,  86,   4,  82,  83,  52,  87,  76,   4,  84,
          4,   4,   4,  90,   4,  94,  94,  94,  94,   4,  85,  95,   4,  93,
         93,  93,  93,   4, 112,   4, 109,   4,   4,   4,   4, 106,   4,   4,
          4,   4,   4,  86,  86,   4,   4, 103, 120,   4, 124, 124, 124, 123,
        124, 124, 124, 127, 124, 124, 124, 124, 130, 124,   4, 144,   4,   4,
          4, 151, 142, 147, 138, 152, 145, 153, 150, 160, 157, 156, 159, 155,
        166, 162,   4,   4, 158,   4, 167, 146,   4, 164, 164, 164, 164,   4,
        170,   4,   4,   4,   4, 176, 178,   4,   4, 182,   4, 183,   4,   4,
          4, 175, 185,   4, 189, 149,   4, 194, 195,   4,   4,   4, 199,   4,
        201,   4, 191,   4,   3], device='cuda:0')
argmax:
tensor([  6,   5,   4,   5,   5,   5,   4,   8,   4,   4,   4,  11,  16,   4,
          4,   4,   4,   6,   4,  24,  25,   4,  21,   4,   4,   4,  30,   4,
          4,  33,  37,  34,   4,  17,   4,  38,   4,   4,  44,  49,  31,  39,
          4,   4,   4,   4,  51,  58,   4,  41,   4,   4,  59,  62,   4,  53,
          4,   4,  64,   4,  47,   4,   4,   4,  64,   4,  60,  77,   4,   4,
         66,  72,  78,   4,  70,  86,   4,  82,  83,  52,  87,  76,   4,  84,
          4,   4,   4,  90,   4,  94,  94,  94,  94,   4,  85,  93,   4,  93,
         93,  93,  93,   4, 112,   4, 109,   4,   4,   4,   4, 106,   4,   4,
          4,   4,   4,  86,  86,   4,   4, 103, 124,   4, 124, 124, 124, 124,
        124, 124, 124, 124, 124, 124, 124, 124, 124, 124,   4, 144,   4,   4,
          4, 151, 142, 147, 138, 152, 145, 153, 150, 160, 157, 156, 159, 155,
        166, 162,   4,   4, 158,   4, 167, 164, 164, 164, 164, 164, 164,   4,
        170,   4,   4,   4,   4, 176, 178,   4,   4, 182,   4, 183,   4,   4,
          4, 175, 185,   4, 189, 149,   4, 194, 195,   4,   4,   4, 199,   4,
        201,   4, 191,   4,   3], device='cuda:0')
        Validation loss: 0.027782148928148672
        Validation accuracy: 0.9928250204415372
```

Figure 5: The labelled output tensor compared to the expected tensor. The model's accuracy decreases as the tensor indices increase, evidenced in the ninth row.

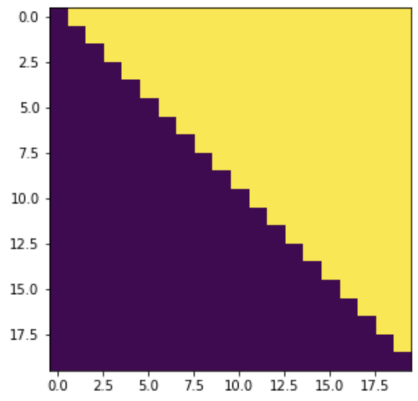


Figure 6: Additive attention mask applied to the decoder tokens. This upper triangular mask allows the decoder to prevent tokens attending to those which are unseen at a given point. Yellow squares are $-\infty$, purple squares are 0.