# Microsoft Research Sentence Completion Challenge Assignment

Advanced Natural Language Engineering

Rupert Menneer

## 1 Introduction

The objective of the Microsoft Research Sentence Completion Challenge (MRSCC) is to predict the most likely missing word from the 1,040 challenge sentences [1]. Each sentence is accompanied by five candidate words, one of which is correct. These candidate words were selected through an n-gram model, meaning that they make good probabilistic fit given the local context words. Furthermore, candidates were hand-curated to ensure the correct choice was unambiguous yet not immediately obvious (e.g. no incorrect grammar). These sentences and candidate words were selected from five Sherlock Holmes novels. Researchers are provided with a training corpus of over 500 19th century novels to build language models from, however the use of external corpora is not prohibited. This challenge presents an important benchmark for language-modelling research due to the significant gap between the author's best model and human performance (49% vs 91%).

An investigation into how language models performed on this challenge was undertaken. Simple unigram and bigram models were used to assess a baseline performance. Three other language models were then investigated to improve on the baseline performance. The models investigated were:

- Continuous Bag Of Words (CBOW) [2]
- BERT [3]
- RoBERTa [4]

The models' performance was evaluated directly against MRSCC accuracy; models scored higher for the more correct candidate words chosen. Using accuracy as a performance metric is not considered best practice in machine learning, however, it was chosen here for two reasons. Firstly, this challenge is a multi-class classification problem with verifiably evenly-distributed classes, therefore accuracy is quite representative of model performance in this scenario. Secondly, accuracy is the performance metric used by the authors and so it was important to keep reported performance consistent with this. For readers' convenience each model and their reported score on the MRSCC challenge is presented in the table below:

| **Model Name** | **MRSCC Accuracy (%)** |
|---|---:|
| Unigram | 26 |
| Bigram | 27.6 |
| CBOW | 32.7 |
| BERT | 71.3 |
| RoBERTa | 78.1 |
| RoBERTa retrained | 82.6 |

**Table 1:** Results of each investigated model on the MRSCC sentences.

Please note that both the BERT and RoBERTa model are pre-trained on masked language tasks. These pre-trained models were acquired through the Hugging Face library. The CBOW and the RoBERTa model were both trained with the 19th century novel corpus. Optimal hyper-parameters were searched for in this training process to achieve best performance. In machine learning it is standard practice to validate models against a held-out validation set. However, since the objective was to maximise performance on the MRSCC, optimal parameters were validated directly against MRSCC accuracy in order to achieve best performance.

## 2 Background

This section provides relevant background knowledge to aid readers in understanding the various language models that were investigated. Summaries of the unigram, bigram, CBOW, BERT and RoBERTa models are included.

## 2.1 Unigram and Bigram Baselines

N-gram models propose a generative approach to modelling natural language by directly calculating the probability distribution of words and word sequences to make predictions [5]. In other words, when trying to predict a missing word in a sentence an n-gram model will predict the most likely word to occur based upon the frequency of this occurring in a training corpus. The simplest n-gram model is the Unigram model which postulates that we can predict the most likely word by simply choosing the most likely word to occur in the corpus.

$$P(w_i) = \frac{w_i}{\sum w_N} \tag{1}$$

In other words the probability a target word occuring is equivalent to the frequency of the word in the data ($w_i$) divided by the frequency of all words in the data ($w_N$). This is a simple approach and does not consider any of the surrounding context. It naively assumes that the probability of a word occurring is independent of the other words seen in a sentence. A Bigram language model attempts to address this by including the previous word in it's probability calculations.

$$P(w_i|w_{i-1}) = \frac{(w_i|w_{i-1})}{\sum(w_N|w_{i-1})} \tag{2}$$

This n-gram technique can be adapted so that it considers 'n' previous or future words, hence the name n-gram model. In order to apply these n-gram techniques to the MRSCC the likelihood of the candidate words occurring was calculated over the training corpus of 19th century novels.

## 2.2 CBOW Model

The continuous bag of words model proposes a discriminative approach to predicting a masked word given its surrounding context. It postulates that each word may be represented by an n-length vector known as a word-embedding. The objective of the CBOW model is to learn word embeddings such that semantically similar words have similar vector representations [2, 6]. In order to achieve this a vocabulary is created where each word is mapped to an initially random vector representation. Then a prediction task is formulated where the aim is to predict a masked word given the sum of the surrounding context. Training examples are generated from a corpus simply by taking a window of consecutive words and masking the central word. These surrounding word vectors are summated to predict the masked word.

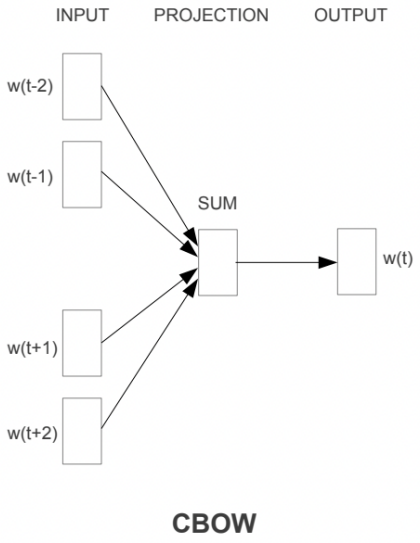


**Figure 1:** (From [2] their Figure 1.) This figures displays the model architecture for the CBOW model.

The architecture of the CBOW model can be described as an input layer of shape (vocab size to embedding size), a summation operation followed by a dense linear layer of shape (embedding size to vocab size). The output from this linear layer is a prediction over the entire vocabulary of words. The Softmax function is applied to this prediction to get a probability distribution over the vocabulary that adds up to 1. CBOW predictions:

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{3}$$

$$P(w_0) = \sigma(W(\sum_{i=-n, i\neq 0}^{n} w_i)) \tag{4}$$

Where the word to predict is $w_0$, the embeddings for context words are $w_i$, the Softmax function is $\sigma$, the linear layer weight matrix is $W$ and there is window size of $n$.

The error between this vocabulary prediction and the true target word is calculated and this is back-propagated through the network which updates the word

embeddings. This can also be viewed as a multi-class classification problem where each class is a word in the vocabulary.

### 2.3 BERT

The Bidirectional Encoder Representations from Transformers (BERT) improved performance on many language-tasks [3]. The model is based upon transformer encoders [7]. These encoders are made of two units: multi-headed attention and a feed-forward neural network. Attention is a system used to create dependencies over the input sequence. It is calculated by first projecting each input in the sequence to Query (Q), Key (K) and Value (V) vector [8]. Attention is then calculated for each input:

$$Attention(Q, K, V) = \sigma(\frac{QK^T}{\sqrt{d_k}})V \tag{5}$$

Where $d_k$ is the dimension of the key vector and $\sigma$ is the Softmax function. Multi-headed self-attention is this process done repeatedly over the same sequence where the Q, K and V are generated from randomly initialised projection matrices. The projection matrices are learned during training. These multi-heads create different dependency perspectives over the input sequence. The output from the attention-heads are concatenated, this means that for for each embedding in the sequence we get an attention output of the same size. This attention output is then fed into the feed forward neural network and onto the next transformer encoder block.

The BERT base architecture is a stack of 12 transformer encoders [3], these give the capabilities to transform input in such a way to learn syntactic and semantic relationships. Through the use of a sub-word tokenisation scheme the model also handles out-of-vocabulary words which it has not encountered before. This also enriches the vector representations of sequences due to learning semantic information stored in shared sub-words [9] (e.g. 'ing' words share semantic properties).

The model is trained on two tasks: a masked language modelling (MLM) task and a next sentence prediction task (NSP) [3]. The MLM task replaces a proportion of input tokens (∼15%) with a mask token, the model must predict these masked tokens. The next sentence prediction task provides the model two sentences and the model has to predict whether the second sentence truly appears after the first sentence, or if it is an imposter. In this training process, natural language features are learned to improve on these two tasks. The model is given 16GB of text data in this process, which is referred to as pre-training. The idea is that the model has a good representation of natural language out-of-the-box, which can be transferred to other tasks. It is common practice to adapt these model on a down-stream tasks, leviating their language features to increase performance. Pre-training on large corpora of general text and then retraining on a domain-specific task is known as transfer-learning and is an effective strategy in natural language processing [10].

Other important features of BERT are the attention-masks, these allow the model to attend to variable length input up to a max length of 512 tokens. This is achieved by padding and truncating sequences to a max length of 512. The attention mask is a vector of 1's for all input other than padding tokens where the value is 0.

### 2.4 RoBERTa

The Robustly optimised Bidirectional Encoder Representations from Transformers approach (RoBERTa) is a revision of BERT [4]. It is shown that BERT is under-trained and under-optimised. RoBERTa addresses this by revising BERT so that is trained for longer, on larger batches and on more text (160GB). They drop the NSP and rely solely on the MLM task. The RoBERTa model implements dynamic masking where training samples have different masks every training epoch. This adapted BERTs static masking procedure where the same tokens are masked over all training epochs. There are other subtle differences such as the token encoding scheme, yet for the most part they are architecturally similar models. However, RoBERTa has a better inner representation of language compared to BERT due to the improved training process. This means the model has better language features which improve performance when adapted to downstream tasks. This results in the RoBERTa model significantly out-performing BERT on language benchmarks [4].

## 3 Methods

This section provides the methodological approach taken to applying each of the described models to the Microsoft Research Sentence Completion Challenge (MRSCC).

The Unigram and Bigram models are omitted in this section as these were considered baseline models.

## 3.1 Implementation overview

Simple data pre-processing using the Gutenberg library to strip out the Gutenberg headers was used. The CBOW model was implemented with the Pytorch Library and optimised with the Ray Tune library [11]. The Transformer models were acquired and retrained with the Hugging Face library [12]. Models were trained using GPUs on Google Colab [13].

## 3.2 CBOW Implementation

The CBOW implementation replicates the architecture introduced in the word2vec paper (one hidden layer) [2]. The model was trained on context-target pairs.

### 3.2.1 Generating context-target pairs

Training examples were generated from the training corpus by simply creating one string for each text and creating a sliding window, such that a target word was chosen and the context was generated from the immediate words around it (+/- the context window size). These examples were converted to their vocabulary IDs before being fed to the model's embedding layer. Therefore the context was in the form $[w_{-i}, ..., w_i]$ for a window size of $i$, and targets were in the form $w_0$. The number of context-target pairs can grow extremely large and approximately equivalent to:

$$(\sum_{i=0}^{N} w_i) - 2 * k \tag{6}$$

Where $k$ is the window size and N is the number of words in the entire tokenised texts. After an initial comparison between the whole training corpus and a subset, it was found there was minimal to no performance gain from using the entire corpus with this method. Therefore the training corpus was restricted to include 50 texts only (16 Conan Doyle novels + 34 randomly selected) to allow for proper hyper-parameter tuning.

### 3.2.2 Model architecture

The CBOW can be described as a neural network with one hidden layer. an input layer of shape (vocab size * embedding size), a summation operation followed by a dense linear layer of shape (embedding size * vocab size). The linear layer outputs have a softmax applied to give a probability distribution over the vocabulary. This is then compared with a one-hot vector representing the target word. The back-propagation algorithm then updates the weights of the network. The word embeddings can simply be extracted from the weights of the first layer of this network. This was implemented through Pytorch and their embeddings functionality .

### 3.2.3 Hyper-parameter optimisation

In order to optimise performance of the CBOW several hyper-parameters were investigated. This was done with the hyper-parameter tuning library Ray Tune library [14] and the Asynchronous Successive Halving Algorithm (ASHA) [15]. Refer to Table 2 for which hyper-parameters were investigated and which were fixed.

| **Hyper-parameter** | **Possible values** |
|---|---|
| Adam Learning rate | 2e-4, 2e-5, 2e-6 |
| Embedding size | 64, 128, 256 |
| Stop word filter | True, False |
| Contextual window size | 2, 3, 4, 5 |
| Max epochs | 20 |
| Batch size | 64 |
| Adam Beta 1 | 0.9 |
| Adam Beta 2 | 0.999 |
| Adam Epsilon 1 | 1e-08 |

**Table 2:** Table displays hyper-parameters investigated for the CBOW model and the possible values they could take.

The max number of epochs was fixed, although ASHA uses aggressive early stopping was used for unfavourable trials. A batch-size of 64 was the largest possible batch-size due to hardware constraints. In order to speed up training time this was fixed at this value, this allowed for more hyper-parameters to be investigated. The number of neural-network layers and their sizes were also unchanged, this was due to wanting to replicate the the CBOW architecture from the original paper.

## 3.3 BERT/RoBERTa implementation

### 3.3.1 Pre-trained models

The BERT and RoBERTa models [3, 4] were implemented with the Hugging Face library, which offers a great library for obtaining and training transformer-based models [12]. Both of these models come pre-trained on large corpora of text. However, it is recommended that they are fine-tuned on downstream tasks to get best performance, in this instance this meant re-training the models with a masked language task using the training resource of 19th century novels. Before re-training these models, their initial performance was assessed on the MRSCC challenge. It was found that even without fine-tuning they significantly outperformed the CBOW and n-gram models. Furthermore, as expected the RoBERTA model outperformed BERT in this task. As these models are computationally expensive to train it was decided that only RoBERTa would be retrained to try and achieve best performance on this challenge. A RoBERTa model was implemented using the Hugging Face Trainer and their Dataset functionality [12].

### 3.3.2 RoBERTa retraining strategy

The Gutenberg training texts were tokenised, this was achieved with the RoBERTa tokeniser from Hugging face. Once each text had been converted into their token IDs they were segmented into sequences with a max length of 128 tokens. A data collator for masked language modelling was then used to replace 15% of tokens with the special <mask> token. This also created a label sequence over the input such that in the place of non-masked words was the value -100 and the value in place of the masked word was the true token ID. This is for a loss function to calculate the difference between the prediction and the true values (-100 values are ignored in the loss). Attention masks were also generated over each of the sequences to handle the few which were of variable length. RoBERTa was then retrained using these masked input IDs, attention masks and the label sequences. Please see Figure 5 in the appendix for an example diagram of this process.

### 3.3.3 Multi-mask or single mask prediction

At inference time there were two strategies investigated, I refer to these as the single-mask and multi-mask approach. The single mask approach predicts a single-mask token in place of the missing word in the sentence, this is used to generate an average probability for the candidate tokens, the highest is taken as the prediction. In contrast, the multi-mask approach handles the problem differently: a sentence for each candidate word is generated, and the sentences are tokenised into their token IDs. One word may be broken down in many tokens (e.g. 'instantaneously' is broken down into 'Ġinstant', 'ane', 'ously', Ġ is a special character signifying the start of a new word). The tokenised sentences are then masked in respect to candidate word token IDs. E.g. if the candidate word is 3 tokens in length then the candidate sentence will have 3 mask tokens in its place. A prediction for each token is then taken over each masked token and averaged to get the final prediction. This approach was far more computationally costly, however it significantly outperformed the single mask approach and was chosen as the correct way to make predictions for the rest of the investigation.

## 3.4 Hyper-parameter optimisation

The RoBERTa base model is large (∼125M parameters) and therefore computationally expensive. It was found that one pass over the entire training corpus took approximately 2-3 hours to run depending on allocated GPU hardware.

| **Hyper-parameter** | **Possible Values** |
|---|---|
| Training Corpus Subset | Refer to table 6 |
| Learning Rate | 1e-5, 1e-4 |
| Weight decay | 0.1, 0.01 |
| Epochs | 3, 5, 10, 15 |
| Batch Size | 8, 32 |
| Sequence Length | 128 |
| Adam Beta 1 | 0.9 |
| Adam Beta 2 | 0.999 |
| Adam Epsilon | 1e-08 |
| Max Gradient Norm | 1 |
| Random seed | 42 |

**Table 3:** Table displays hyper-parameters investigated for the RoBERTa model and the possible values they could take.

Even with smaller proportions of the training data it did not lend itself to automatic hyper-parameter tuning as a maximum of 8 hours of GPU time is enforced on Google

Colab. Colab was used to acquire a GPU that massively increased training time, so it was not an option to perform these trials locally. This meant a manual investigation was launched with varying subsets of the training data, where the results were recorded for each trial. Furthermore, a batch size of 32 and a sequence length of 128 were found to be the largest the model handle before running out of GPU memory. A small number of key hyper-parameters were chosen to be investigated in order to display meaningful difference in the trials. The parameters investigated were training corpus subset, learning rate, batch size and number of epochs. Please reference table 3 for the hyper-parameters investigated and those that were fixed.

# 4 Results

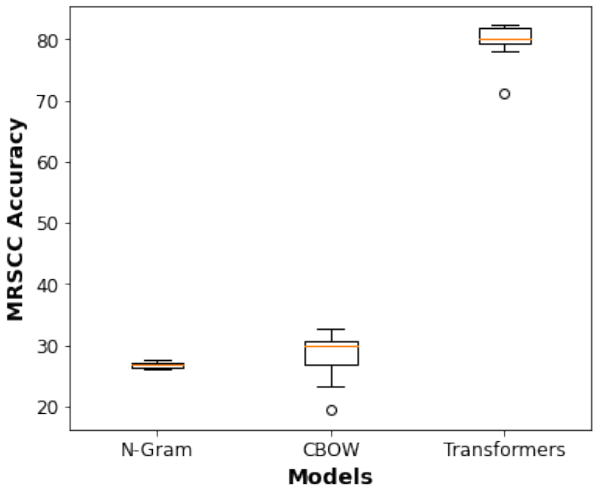


**Figure 2:** This figure compares model performance on the MRSCC challenge sentences. Please note that BERT, RoBERTa and RoBERTa finely-tuned are displayed jointly under 'Transformers' and that both unigram and bigram baselines are displayed jointly under 'N-Gram'.

The following section reports the results that each method achieved on the MRSCC task. This includes findings from hyper-parameter optimisation.

## 4.1 Overview

This investigation discovered that n-gram models performed poorly on this task, this was expected as candidate words for each sentence were selected via the use of a sophisticated n-gram model, this undermines how well an n-gram model can perform on this task. Furthermore, the CBOW model performed only slightly better, although this model was discriminative (as opposed to the generative n-gram) this model still only considers immediately local information to make predictions. Additionally, it had the largest variation in performance and no clear pattern in optimal hyper-parameters. It was not clear to what degree random weight initialisation had effect on the final performance. The transformer-based models outperformed other models by a significant degree. This was to be expected considering how well these models perform on language bench-marking. Their ability to process variable sequence lengths, high contextual understanding and the large volume of pre-training data were undoubtedly reasons for this success. Both BERT and RoBERTa were pre-trained with the BookCorpus meaning they already performed well on novel-based language tasks such as the MRSCC [3, 4]. Fine-tuning the RoBERTa model was able to achieve a modest performance increase. Longer training times and larger batch sizes seemed to produce better results for this model, this concurs with the findings from the RoBERTa paper [4].

## 4.2 Hyper-parameter tuning

For the results of hyper-parameter tuning please reference Table 5 and Table 6 in the appendix for full trial details. Please also view Figure 3 and Figure 4 in the appendix for a graphical representation of these trials.

# 5 Evaluation

The MRSCC is an evenly distributed multi-class classification problem, meaning that a model which randomly picks answers can acheive ∼20% accuracy on this task. The Unigram, Bigram and CBOW models perform slightly better than chance at between 25-32% accuracy. The authors n-gram model that generated candidate words achieves around 31% on this task, it is therefore hard to meaningfully assess these models beyond a generative probabilities of local information. It is also difficult to verify which answers the model picked by chance and which it picked through its inner language representation. The CBOW model is almost equivalent to generative n-gram (32% vs 31% respectively). Although architecturally different, they are modelling sim-

ilar objectives - the probability of the candidate word occurring giving the local immediate context words.

The transformer based models leave more room to verifiably identify weaknesses. The highest scoring model achieved 82.6% accuracy on this task. This is an 8.4% absolute difference between the human judgement score reported in the MRSCC paper. The retrained RoBERTa model was analysed to see where the model was making mistakes. Please reference Table(s) 4 below for model analysis.

| Class | Correct proportion |
|---|---|
| A | 0.83% |
| B | 0.82% |
| C | 0.85% |
| D | 0.82% |
| E | 0.82% |

| Comparison | Incorrect | Correct |
|---|---|---|
| Question lengths | 103.7 | 101.3 |

| Comparison | Prediction | Answer |
|---|---|---|
| Avg candidate word freq | 1901 | 603 |

**Table 4:** All tables are in relation to the Retrained RoBERTa model. The top table displays what proportion the model correctly predicted each class. The middle table compares question length in characters between the correct and incorrect predictions. The bottom table displays the average occurrence in the training corpus between the word the model incorrectly predicted and the correct answers for those sentences.

This investigation highlights one key weakness of the model. For sentences where the model failed to predict the correct candidate word, on average the word chosen was more than 3x more likely to appear in the training corpus. This perhaps suggests that the model is still relying on probabilistic judgements over the entire corpus to make predictions.

# 6 Further Work

This section discusses improvements to be made on the investigated models and suggestions for further work in this area.

The CBOW model performed only slightly better than the n-gram models. This model was first proposed with a sibling model referred to as the skip-gram model [2]. Instead of using the context to predict the target word, the target word is used to predict the context. This could be applied to the MRSCC by taking each candidate word and predicting the probability of the context surrounding it. Research suggests that this would fare better than CBOW as it produces better representations of infrequent words in the vocabulary [2]. The MRSCC authors improve upon the base n-gram model performance by producing smoothed n-gram models for a maximum score of 39%. Similar functionality could be implemented into the CBOW or skip-gram model and this would be a promising avenue to increased performance. Smoothing or negative sampling techniques have been reported to improve performance on these models [6, 2].

The transformer based models performed exceptionally well on this task yet still has a significant gap to reach in order to reach human-level understanding (82.6% vs 91%) of this task. Additionally the large size of these models made it unfeasible to be fully optimised over the full dataset. It seems very plausible that the best score could have been improved by training over the whole corpora for more epochs than was explored here. Model check-pointing would be one solution to this dilemma, yet this does not reduce the overall hours of compute time. Distilled versions of these models (DistillBert and DistillRoBERTa) exist with far fewer weights than the original models lending itself to increase optimisation and improved inference time [16]. Initial testing with these distilled versions saw moderate drop offs in accuracy, and weren't pursued for this reason. However, bridging the gap between these distilled and the base models offers a rich area for future research.

# 7 Conclusion

The MRSCC presents a difficult and insightful challenge for NLE researchers. Candidate words were picked through the use of a probabilistic model meaning researchers must develop language models which go beyond generative probabilities and local context. The transformer models' large pre-training corpora and ability to process variable sequences seemed to provide mechanisms to significantly outperform the other investigated models. However, the size and computational

requirement of these models was identified as a clear drawback. This limits the scope that NLE researchers can investigate a challenge such as the MRSCC down to available hardware resources.

# References


[1] G. Zweig and C. J. C. Burges, "The microsoft research sentence completion challenge," 12 2011. [Online]. Available: http://research.microsoft.com/scc/.

[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space." International Conference on Learning Representations, ICLR, 1 2013. [Online]. Available: https://arxiv.org/abs/1301.3781

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, 10 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv*, 7 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[5] J. Weeds, "Anle university of sussex week 2 slides," *UoS Lectures*, pp. 11–12, 2021.

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, 10 2013. [Online]. Available: http://arxiv.org/abs/1310.4546

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," vol. 2017-December. Neural information processing systems foundation, 6 2017, pp. 5999–6009. [Online]. Available: https://arxiv.org/abs/1706.03762v5

[8] J. Alammar, "The illustrated transformer – jay alammar – visualizing machine learning one concept at a time." 6 2018. [Online]. Available: http://jalammar.github.io/illustrated-transformer/

[9] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 7 2016. [Online]. Available: http://arxiv.org/abs/1607.04606

[10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv*, vol. 21, pp. 1–67, 10 2019. [Online]. Available: http://arxiv.org/abs/1910.10683

[11] Pytorch, "Word embeddings: Encoding lexical semantics — pytorch tutorials 1.8.1+cu102 documentation," 2021. [Online]. Available: https://pytorch.org/tutorials/beginner/nlp/word_embeddings_tutorial.html

[12] Hugging-Face, "Hugging face library." [Online]. Available: https://github.com/huggingface

[13] Google-colab, "Google colab - cloud computing." [Online]. Available: https://colab.research.google.com

[14] Ray-Tune, "Automatic hyper-parameter optimisation tool." [Online]. Available: https://docs.ray.io

[15] L. Li, K. Jamieson, A. Rostamizadeh, E. Gonina, M. Hardt, B. Recht, and A. Talwalkar, "A system for massively parallel hyperparameter tuning," 10 2018. [Online]. Available: http://arxiv.org/abs/1810.05934

[16] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv*, 10 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[17] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He, "A comprehensive survey on transfer learning," pp. 43–76, 11 2019. [Online]. Available: http://arxiv.org/abs/1911.02685

# 8 Appendix

| Trial | Learning rate | Embedding dim | Stop filter | Window size | Epochs | MRSCC Accuracy |
|---|---|---|---|---|---|---|
| 1 | 2e-05 | 64 | True | 2 | 5 | 32.21 |
| 2 | 2e-06 | 64 | True | 3 | 3 | 19.32 |
| 3 | 2e-04 | 64 | False | 2 | 5 | 30.28 |
| 4 | 2e-05 | 64 | True | 3 | 3 | 29.51 |
| 5 | 2e-04 | 128 | True | 4 | 5 | 29.71 |
| 6 | 2e-04 | 64 | False | 4 | 5 | 30.67 |
| 7 | 2e-05 | 256 | True | 4 | 5 | **32.59** |
| 8 | 2e-04 | 256 | True | 10 | 3 | 25.86 |
| 9 | 2e-04 | 128 | True | 3 | 3 | 30.00 |
| 10 | 2e-04 | 64 | True | 5 | 3 | 23.17 |

**Table 5:** Results from CBOW hyper-parameter tuning.

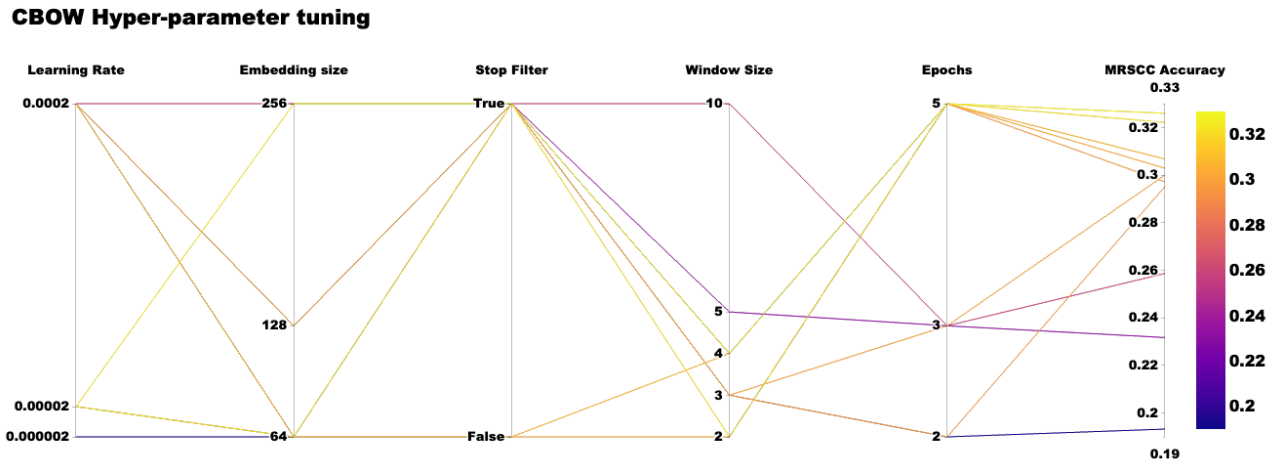


**Figure 3:** This figure shows a graphical representation of the hyper-parameter tuning for the CBOW model. Each line represents a model with different parameters. The final scale represents the accuracy that this model achieved on the MRSCC challenge sentences.

| Trial | Training Corpus | Batch size | Epochs | Accuracy |
|---|---|---|---|---|
| 1 | Conan Doyle only (16 books) | 8 | 5 | 79.80 |
| 2 | Conan Doyle only (16 books) | 32 | 5 | 80.02 |
| 3 | Conan Doyle + random (50 total) | 32 | 5 | 80.09 |
| 4 | Conan Doyle + random (50 total) | 32 | 15 | **82.59** |
| 5 | Conan Doyle + random (100 total) | 32 | 10 | 81.92 |
| 6 | All texts (6 hour run time) | 32 | 3 | 81.82 |

**Table 6:** Results from RoBERTa hyper-parameter tuning.

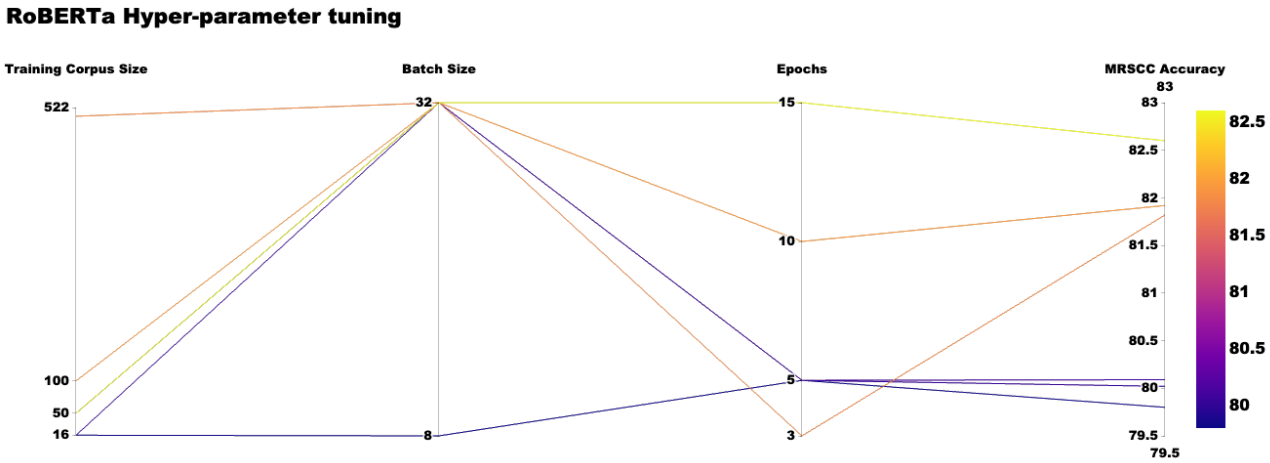


**Figure 4:** This figure shows a graphical representation of the hyper-parameter tuning for the RoBERTa model. Each line represents a model with different parameters. The final scale represents the accuracy that this model achieved on the MRSCC challenge sentences.

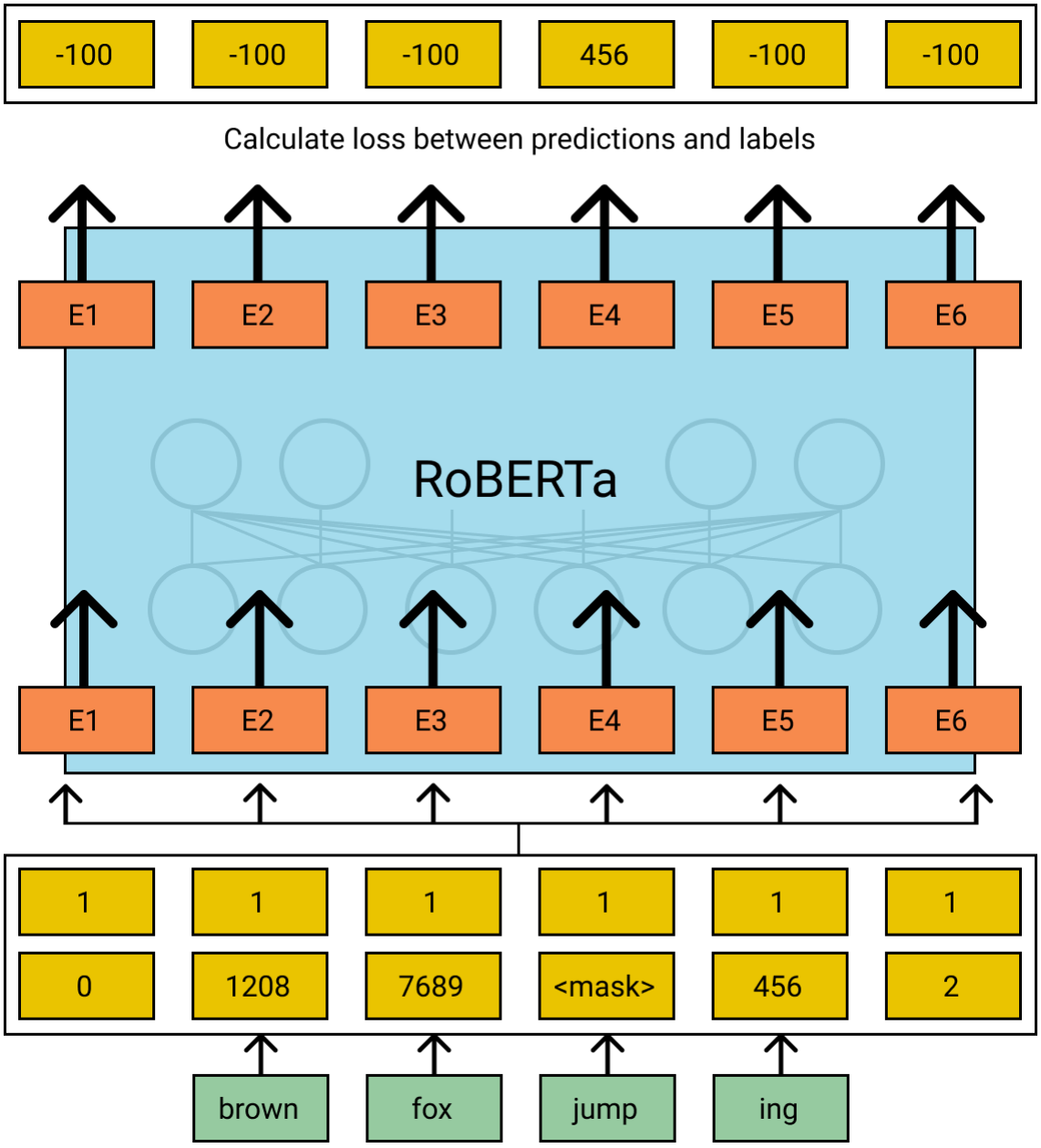


**Figure 5:** This figure shows an example input sequence 'brown fox jumping' passed as tokens into the model. This is transformed into a masked input ID sequence and an attention mask. These input IDs and attention masks are passed into the RoBERTa model. The output from the RoBERTa model is compared with the labels of the original input sequence. Note that non-masked inputs are replaced with -100 as they are ignored by the loss function; masked inputs are replaced with their true input IDs.