RANK OVER CLASS: THE UNTAPPED POTENTIAL OF RANKING IN NATURAL LANGUAGE PROCESSING

Amir Atapour-Abarghouei
School of Computing
Newcastle University, UK
amir.atapour-abarghouei@newcastle.ac.uk

Stephen Bonner
School of Computing
Newcastle University, UK
stephen.bonner3@newcastle.ac.uk

Andrew Stephen McGough
School of Computing
Newcastle University, UK
stephen.mcgough@newcastle.ac.uk

ABSTRACT

Text classification has long been a staple in natural language processing with applications spanning across sentiment analysis, online content tagging, recommender systems and spam detection. However, text classification, by nature, suffers from a variety of issues stemming from dataset imbalance, text ambiguity, subjectivity and the lack of linguistic context in the data. In this paper, we explore the use of text ranking, commonly used in information retrieval, to carry out challenging classification-based tasks. We propose a novel end-to-end ranking approach consisting of a Transformer network responsible for producing representations for a pair of text sequences, which are in turn passed into a context aggregating network outputting ranking scores used to determine an ordering to the sequences based on some notion of relevance. We perform numerous experiments on publicly-available datasets and investigate the possibility of applying our ranking approach to certain problems often addressed using classification. In an experiment on a heavily-skewed sentiment analysis dataset, converting ranking results to classification labels yields an approximately 22% improvement over state-of-the-art text classification, demonstrating the efficacy of text ranking over text classification in certain scenarios.

1 Introduction

Recent advances in machine learning over the past decade have led to significant strides in various active areas of research, such as image recognition [1, 2], scene understanding [3, 4, 5, 6] and robotic navigation [7]. Many such applications have already been integrated into, and have become an essential part of, our daily lives. However, as natural languages are the primary method of human communication, learning-based Natural Language Processing (NLP) is receiving an ever-increasing level of attention within both academia and industry. Amongst the various applications of NLP, text classification [8, 9] has arguably blazed the trail due to the simplicity of its definition and its numerous use cases. From online content tagging [10] to sentiment analysis [11], text classification has always been at the forefront of natural language processing.

With the emergence of deep learning, various approaches have addressed text classification via feed-forward networks using bag-of-word inputs [12], recurrent neural networks that consider structural elements of the text [13] and convolutional neural networks capable of detecting position-invariant patterns in the text [14]. Transformers [15], however, have arguably been the greatest advancement in NLP with significant improvements enabled by large-scale pre-trained language models, taking advantage of deeper architectures and larger corpora of text for better representation learning.

However, despite the successes of text classification, there still remain significant challenges. For instance, classification of text data can be highly subjective due to the presence of textual ambiguity and potential unknown classes. Examples of this are widely seen in the numerous publicly-available movie review datasets [16, 17, 18], commonly used as benchmarks for text classification. For instance, imagine the following sentence from a movie review:

> **Example 1:** Movie Review - *while plagued with a plodding mess of a narrative, the sincere performance of the character salvages the clichéd dialogue and provides some escapism from the distorted perspective of the protagonist.*

Any observer would have trouble accurately labelling this review as positive or negative. Similarly, the performance of a machine learning model would solely depend on the presence of similar words, patterns and structures of the text in other less ambiguous and more concrete data points in the dataset. This level of subjectivity and ambiguity essentially makes a fair and accurate classification of the sentiment of such passages impossible using learning-based models.

Another example of the challenges of text classification lies in the large number of closely-related classes for a given problem. For instance, the now-discontinued Google Directory service [19, 20] included around 2 billion classes in a deep hierarchy, most of which were extremely close to each other, making it difficult for a learning-based classification approach to correctly classify unseen data.

Further significant issues with text classification stem from dataset imbalances. If opinions on a specific topic are extracted from social media to be used as training data, most opinions would lie on the extreme ends of the spectrum as individuals with extreme beliefs are more likely to voice their opinions publicly, thus creating a skew in the dataset towards more extreme opinions. In another scenario, if subjective passages are labelled by multiple annotators with an averaging method determining the final label, most of the passages will inevitably fall near the centre of the labelling interval due to the subjectivity of the annotation process. Such data imbalances can impede accurate classification and necessitate significant algorithmic intervention.

There are numerous other similar shortcomings in text classification, depending on the scope of the problem, the nature of the classes and the features available in the dataset. Here, we argue that many such issues can be resolved by reformulating certain text classification problems as *text ranking*. A ranking problem is defined as a derivation of ordering over a list of items that maximises the utility of the entire list [21]. Ranking is a significant component of many information retrieval systems with applications including web search, recommender systems, document summarisation and question answering [22]. Ranking, by nature, is different from classification and regression in that a classifier or a regressor attempts to assign a specific class label or value to an individual data point, while the objective of a ranking approach is to optimally sort a list, such that, for some notion of *relevance*, the items within the list with higher relevance scores appear earlier in the list.

This is precisely why we believe ranking is a more appropriate formulation of many problems currently solved with classifiers. Take the movie review from *Example 1.* While it might be difficult for a human or a neural network to accurately classify this as a positive or a negative review or associate it with an absolute sentiment score, it is significantly easier to rank this review with respect to other reviews as *more positive* or *more negative*, especially if they are placed in some known shared *context*, for example all the reviews given for the same movie. Arguably, ranking is more aligned with the final objectives of many text classification applications, as assigning an arbitrary absolute score or class to an individual piece of text reveals less about it than placing said text in some context along with similar passages. In this paper, we explore the capabilities of text ranking and demonstrate the efficacy of ranking over classification using experiments conducted over publicly-available datasets. In short, the primary contributions of this work are as follows:

- With end-to-end training based on representations produced by a Transformer, a context aggregating dense network and a ranking loss, passages can be accurately ranked based on some ranking label or score (Section 3.1).
- The potentials of ranking are demonstrated on various datasets [18, 23] with the learning process *pivoting* around the different contexts the passages can be placed in, *e.g.* the writer of the passage or the topic of the passage, and the efficacy of contextual ranking is explored (Sections 4.1 and 4.2).
- When ranking results are artificially converted to class labels, our approach is capable of outperforming state-of-the-art text classification models on heavily-skewed text classification datasets [24] (Section 4.3).

To enable reproducibility, a PyTorch implementation of our ranking approach is made publicly available[1].

[1] https://github.com/atapour/rank-over-class

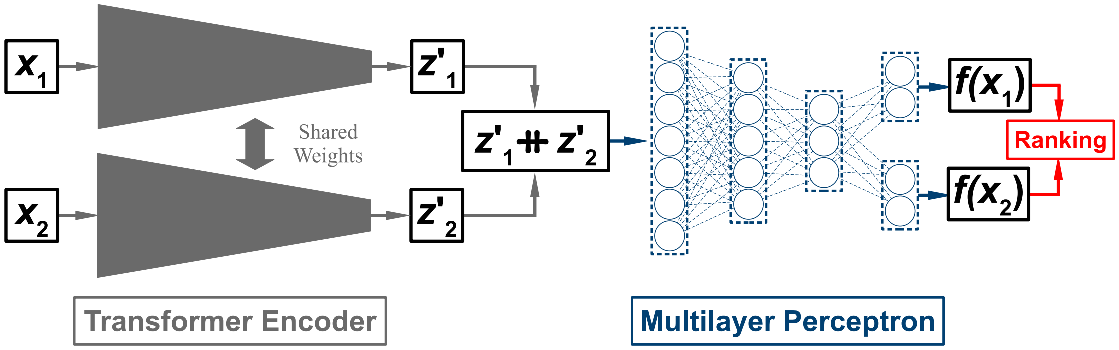


Figure 1: Training procedure of the overall model, approximating the ranking function, f. Input passages, x_1 and x_2, are first passed through a Transformer. A multilayer perceptron takes the concatenated outputs ($z'_1 + z'_2$) and produces the ranking scores, $f(x_1)$ and $f(x_2)$, used to rank the inputs.

2 Related Work

Having first appeared in the literature in the 1940s [25], ranking gained prominence as the foundation of modern search engines towards the end of the millennium [26]. Given a query q and a collection P of passages p that match the query, the goal is to rank the passages in P according to some notion of relevance to q so that the *best* results appear earlier. Note that while we focus on passage ranking in this work, the same concepts can equally be applied to many different forms of data.

Though traditionally solved via boolean, vector space and probabilistic models [27, 28], the ranking problem is now commonly addressed using learning-based approaches [22], taking advantage of labelled data and some parametrised function to map feature vectors extracted from items in a list to real values used as ranking scores. This function is subsequently used to sort the items. Based on the loss functions used to optimise the ranking process, the approaches can be point-wise, pair-wise or list-wise.

Point-wise approaches [29, 30] utilise a classifier or a regressor trained to predict the relevance score of a passage with respect to a given query with the ranking achieved by sorting the passages based on said score. When the number of relevant passages varies for different queries, the loss function is dominated by queries with larger numbers of passages, creating an imbalance in training. Pair-wise techniques [31, 32], on the other hand, consider a pair of items in their loss function. The final objective in such an approach is to minimise the number of inversions in the ranked list, where the items are in the wrong order relative to the ground truth. List-wise approaches [33, 34, 35] attempt to solve for the optimal ranking for the entire list all at once. This is often done either via a loss function designed based on the unique properties of the items that are to be ranked [36] or by directly optimising certain information retrieval metrics [37].

Here, we propose a pair-wise ranking approach, not only suited for information retrieval but also capable of providing better performance for many common classification problems in natural language processing. The details of the proposed approach are explained in Section 3.

3 Proposed Approach

Our approach is designed to perform pair-wise ranking over a list of passages based on some ranking score, which is chosen based on the features available in the dataset and the parameters of the problem. Consisting of two sub-networks trained end to end, the model receives two passages as its input and outputs two corresponding scalar values which can be used to determine whether the passages have been input in the correct order or not. Further details of the approach are discussed in the following section.

3.1 Overall Ranking Model

In text classification, either it is assumed that the entirety of the training dataset falls within the same context or the context is completely ignored. Context, in this setting, refers to a unifying element which the passages can be grouped by and provides a coherent background for understanding what those passages represent. In information retrieval, the query is what often provides this context, which is missing in most classification tasks. To get a clearer picture of what is meant by context, let's refer to the following examples:

Classification Model	(a) Stack Exchange			(b) Fine Foods		
	Accuracy	F_1 Score	AUC	Accuracy	F_1 Score	AUC
ALBERT [41]	0.272	0.278	0.618	0.722	0.617	0.810
RoBERTa [40]	0.281	0.291	0.635	0.705	0.590	0.809
GPT2 [39]	0.285	0.298	0.622	0.765	0.738	0.813
BERT [38]	0.292	0.309	0.648	0.768	0.721	0.828

Table 1: Classification for (a) quality assessment of Stack Exchange posts and (b) sentiment analysis of Fine Food Reviews.

> **Example 2:** a human observer is given three random passages and asked to assess their quality. One passage is an excerpt from a technical report, one a poem and the other from a scientific textbook.

> **Example 3:** a human observer is given three random passages and asked to assess their quality. All three passages are answers from different individuals to the same question.

In the case of *Example 2*, as the passages are from widely different sources, the lack of a shared context makes comparing their quality virtually impossible. However, in *Example* 3, as the three passages are all different answers to the same question (shared context), the problem can be solved in a more meaningful and objective manner.

In this vein, we first group the data points (passages) in our training dataset based on some shared context, thereby pivoting the focus of the learning process around it and thus providing a stronger background for the model's representation learning capabilities. From now on, we will refer to this as the *contextual pivot point*. Subsequently, all possible *combinations* of passages within the individual groups generated around our contextual pivot point are extracted and used as training data for our pair-wise ranking model.

The overall pipeline of our approach is seen in Figure 1. The passages are passed through the layers of a headless Transformer network to get a latent vector containing the representation of each passage. In our experiments, BERT [38], GPT2 [39], RoBERTa [40] and ALBERT v2 [41] are used to obtain the feature vectors but any other Transformer model can similarly be used. As seen in Figure 1, the resulting feature vectors of the two passages are concatenated and used as the input to a four-layered multilayer perceptron (context-aggregating network). The multilayer perceptron aggregates the features representing the context and the content of the two passages, assesses the relationship between them and regresses to two values (ranking scores), subsequently used to rank the passages. Note that both the input passage pairs and the output score pairs are correspondingly-ordered and any change in their ordering can affect the performance of the model. Trained end to end, the entire model uses a ranking loss function (explained in Section 3.2) and accurate pair-wise ranking of the input passages is enabled by comparing the values produced by the model.

3.2 Loss Function

While most ranking approaches traditionally utilise loss functions such as sigmoid cross-entropy for binary relevance labels, pair-wise logistic loss or softmax cross-entropy [21], we make use of a margin-based ranking loss, which has been effectively used for representation learning in embedding models [42] by separating the positive samples from the negative samples within the dataset by a given margin. Using this loss function, we can take advantage of its representation learning capabilities to extract more robust features from the passages and better learn their compositional relationship during the ranking process. Formally, for any passage pair p_i and p_j, the ranking label, E, is determined as follows:

$$E(p_i, p_j) = \begin{cases} 1, & p_i \text{ is ranked higher than } p_j \\ \text{-1}, & p_j \text{ is ranked higher than } p_i \end{cases}, \tag{1}$$

where E is the ground truth ranking label. Consequently, for the set of all passage pairs $\Psi = \{(p_i, p_j); E(p_i, p_j)\}$, a pair is fed into our model, which approximates the desired ranking function f. The loss function is thus defined as:

$$\mathcal{L} = \sum_{(p_i, p_j) \in \Psi} \max(0, -E(p_i, p_j) \times (f(p_i) - f(p_j)) + \gamma), \tag{2}$$

where $f(p_i)$ and $f(p_j)$ are the ranking scores produced by the overall model and γ is the margin enforced between $f(p_i)$ and $f(p_j)$. In our experiments, $\gamma = 2$ empirically yields the most favourable results. This loss measures ranking violations of passage pairs, allowing the network to learn discriminative features to enforce a clearer distinction between the passages and produce a more accurate ranking.

3.3 Implementation Details

For all experiments, the smallest pre-trained versions of the Transformers provided by the HuggingFace library [43] are used. Text sequences are truncated if they exceed 128 tokens, except for experiments in Section 4.4, where sequences of 512 tokens are used. The context aggregating multilayer perceptron uses Linear-BatchNorm-PReLu modules with a dropout of 0.2 for each layer during training. All implementation is done in PyTorch [44], with AdamW [45] providing the optimization ($\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$). The learning rate is $\alpha = 4e-5$ for BERT and GPT2 and $\alpha = 4e-6$ for ALBERT and RoBERTa. All experiments are carried out using two NVIDIA Titan RTX GPUs in parallel with a combined memory of 48 GB on an Arch Linux system with a 3.30GHz 10-core Intel CPU and 64 GB of memory. Note that for large NLP models, such as those used in this work, this hardware has limited capabilities, which is why smaller Transformer models and datasets are used in the approach. *For further details on the choice of hyperparameters, please refer to the Appendix.*

4 Experimental Results

To rigorously evaluate our approach, we perform extensive experiments applying our ranking approach to four publicly available datasets, Stack Exchange [23], Fine Food Reviews [18], a dataset of tweets about self-driving cars [24] and the MS MARCO passage ranking dataset [46]. We also conduct ablation studies and demonstrate the importance of the components of the proposed approach.

4.1 Stack Exchange

A large Question and Answer Forum made up of numerous communities focusing on different subjects, Stack Exchange offers a space where users can post questions about specific topics for which they can receive answers from other users. Any user can annotate whether an answer is useful or not by voting for it favourably (up-vote) or unfavourably (down-vote). The original asker can also mark one answer as the best. As these votes are expected to be based on quality, subjective though they might be, they provide an opportunity for a learning-based approach to assess the quality of each answer for a potential recommendation system.

A naïve way to assess post quality (only the *answers* are considered) is to simply use a classification model. The ground truth quality scores can be generated based on the number of votes an answer has received normalised by the number of votes the original question has received. This normalisation removes the possibility of bad answers to more popular questions with a higher vote count overwhelming better answers to less popular questions. The quality scores are then categorised into 5 classes via a simple histogram.

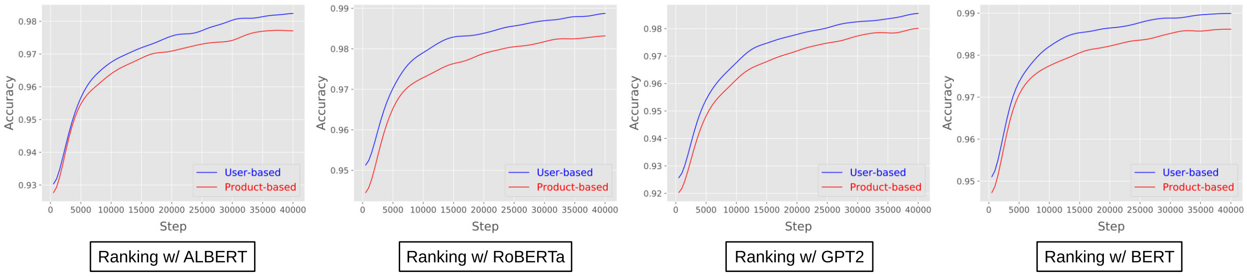
The dataset is extracted from the communities of *Ask Ubuntu*, *Cryptography*, *Data Science*, *Network Engineering*, *Unix & Linux* and *Webmasters*. 250,000 posts are randomly selected for the training dataset and 50,000 for testing. State-of-the-art text classification models [38, 39, 40, 41] are trained for 3 epochs (at which point all models converge) to classify the posts based on their quality scores. As seen in Table 1 (a), all models fail to assess the quality of the posts beyond randomly guessing, evidenced by the low accuracy, F_1 Score and Area Under the ROC Curve (AUC). This is explained by the lack of context providing a coherent backdrop for the passages as they are associated with different users, questions and communities. As demonstrated by *Example 2* in Section 3.1, even a human would have trouble assessing the text quality of unrelated passages without any shared context. This is why a contextual pivot point is necessary.

The Stack Exchange post quality annotations (user votes) are within the context of each question, *i.e.* the users vote on the quality of the posts based on how well the post addresses that specific question. Therefore, measuring the quality of the posts within each individual question would be very easy as the desired task directly aligns with the ground truth data. As a result, we experiment with unique users as the contextual pivot point. There are 17,085 unique users who have answered questions within the dataset selected from the aforementioned Stack Exchange communities with an average of 20.1 answers per user. In our experiments, we intend to assess the quality of passages posted by individual users using our ranking method. While evaluating the quality of posts for each individual user is possible via text classification, it would entail training a separate classification model for each individual user, which is intractable. Our ranking approach, however, is trained end to end over the entire dataset once and can provide an accurate measure of the quality for the posts across all questions, all users and all communities.

With users as the contextual pivot point, the passages are grouped based on unique users and all combinations of passages are extracted as training data (≈8,000,000 pairs). Similar to the classification setup, ground truth ranking labels (Eqn. 1) are based on the number of votes an answer has received normalised by the number of votes the original question has received. To enable evaluation with the appropriate ranking metrics, 100 random users with at least 100 answers that have unique quality scores are selected as the test set.

	Approach	Ranking Metrics @k			Pair Labels
		MRR	NDCG	MAP	Accuracy
(a) Stack Exchange	w/ ALBERT	0.656	0.772	0.662	0.854
	w/ RoBERTa	0.671	0.780	0.678	0.882
	w/ GPT2	0.760	0.806	0.764	0.895
	w/ BERT	**0.762**	**0.814**	**0.768**	**0.910**
(b) Review (User)	w/ ALBERT	0.970	0.982	0.976	0.985
	w/ RoBERTa	0.972	0.984	0.979	0.990
	w/ GPT2	0.975	**0.986**	0.982	**0.991**
	w/ BERT	**0.977**	**0.986**	**0.984**	0.990
(c) Review (Product)	w/ ALBERT	0.972	0.984	0.976	0.981
	w/ RoBERTa	0.970	0.982	0.972	0.985
	w/ GPT2	**0.976**	**0.987**	**0.982**	**0.990**
	w/ BERT	0.974	0.985	0.981	**0.990**

Table 2: Results for (a) Stack Exchange [23], (b) Fine Food Reviews [18] with users as the contextual pivot point and (c) Fine Food Reviews with products as the contextual pivot point.



Ranking w/ ALBERT

Ranking w/ RoBERTa

Ranking w/ GPT2

Ranking w/ BERT

Figure 2: Test accuracy of pair rankings on the Fine Foods reviews dataset (40,000 steps). When the ranking is pivoted around individual users, the models consistently reach convergence faster, due to the overpowering effect of user writing styles.

Table 2 (a) presents the results of our experiment. Better results are obtained with BERT and GPT2. Common ranking metrics, including Normalized Discounted Cumulative Gain (NDCG), Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) are used to evaluate the results with $k = 10$. We have also measured the accuracy of the ranking approach directly applied to passage pairs from the test set (with success referring to an accurate ranking of pairs based on their ranking label - Eqn. 1). As seen in Table 2 (a), despite the fact that the ground truth labels are essentially crowd-sourced and not objective, the results are very promising and highly accurate ranking is achieved.

An interesting application of ranking the passages with the contextual pivot point being the users is the possibility of tracking the change in the skill level of users over time. To test this, we extract a test set of 50 random users with at least 20 answers that have unique ranking scores. Considering that the model receives no information about the time, if the resulting ranking matches the order at which passages were posted, the approach is implicitly capable of tracking the skill level of users, which can be of value in downstream recommendation and forum moderation systems, with opportunities for recommending questions to users and ordering posts based on the skill level of the user, among others. A numerical analysis of this use case is seen in Table 3, which presents the results after evaluating the ranking of a total of 1,683 posts from these 50 random users. Since for this experiment, the accuracy of the comparison of the pairs is what matters, we present the accuracy metric over the 1,633 comparisons of temporally-ordered passage pairs. The promising results in Table 3 are indicative of the potential for better user behaviour and profile tracking, leading to more accurate recommendation and security systems.

4.2 Fine Food Reviews

The Amazon Fine Food Reviews dataset [18] contains around 500,000 reviews of approximately 75,000 products from roughly 250,000 users. The objective is to predict the sentiment of the review based on scores (from 1 to 5) provided in the dataset. As opposed to the Stack Exchange data, the passages in this dataset are significantly easier to classify

Ranking Model	Pair Labels	
	Accuracy	F_1 Score
Ranking w/ ALBERT	0.905	0.908
Ranking w/ RoBERTa	0.910	0.917
Ranking w/ GPT2	0.932	0.936
Ranking w/ BERT	0.945	0.950

Table 3: Accuracy of 1,633 comparisons of temporally-ordered passages from 50 users with at least 20 answers that have unique ranking scores from [23].

as they all substantially belong to the same context (movie reviews). Experimental results presented in Table 1 (b) demonstrate that classification methods are reasonably capable of classifying the reviews albeit with underwhelming results. 20% of the data is randomly selected as the test set and the text sequences are truncated to 128 tokens. The dataset is unbalanced with higher sentiment scores being more prevalent in the dataset. However, if the task of sentiment analysis is reformulated as ranking, significantly better performance can be expected.

The contextual pivot point can be either the *user* (ranking the sentiment of the reviews from individual users for all the products they have reviewed) or the *product* (ranking the sentiment of the reviews of individual products from all users who have reviewed them). We perform experiments based on both users and products as contextual pivot points. Separate training and testing sets are created, with the product-based test set made up of 200 random products with reviews that have unique scores (note that only 5 unique scores exist) and the user-based test set made up of 200 random users with reviews that have unique scores. As our ranking model always ranks a pair of passages, even if they are similar, only pairs with different ranking scores are passed into the model during training and in the test set, passages with unique scores are used to evaluate the model.

Table 2 (b) shows the results of the user-based review ranking experiments and Table 2 (c) the results of the product-based experiments. As there are only 5 items per each user or product (5 sentiment classes), the ranking metrics are calculated with $k = 2$. *Accuracy* alludes to the correctness of pair rankings over all possible pairs in the test set. The metric values are higher than those of Section 4.1, as only five items exist in the list, but as seen in Table 2, the results are extremely promising with the models achieving near perfect ranking in both product and user based experiments. While highly accurate results are achieved across the board, an interesting observation is that it is easier for the model to learn the context and thus perform better ranking when the pairs are pivoted around users as opposed to products. Figure 2 demonstrates how all models consistently reach convergence faster when the contextual pivot point is the user. This is due to the powerful influence of the writing style of users with different reviews. While the reviews of all products share a clear context as evidenced by Table 2 (c), user-based ranking is more easily learned (Figure 2).

4.3 Twitter Sentiment Analysis: Self-Driving Cars

This dataset [24] focuses on people's opinions on autonomous driving and consists of 6,943 tweets relevant to self-driving technologies labelled from 1 to 5, with 1 indicating the most negative sentiment and 5 the most positive. This dataset is heavily skewed with over 61% of the data points annotated as *neutral* (4,245 tweets with label 3) and fewer than 2% of the data points annotated as *very negative* (110 tweets with label 1). This creates significant challenges for any classification approach and requires extreme measures to combat the imbalance in the dataset.

Here, we evaluate the ability of our model to deal with such a challenging dataset. While all the tweets follow the same context (self-driving vehicles), the user that posted each tweet is ignored in this dataset and thus no contextual pivot point is available other than the shared subject of the tweets (self-driving vehicles). The dataset is split into an unbalanced training set and a balanced test set with the test data containing 100 tweets from each class (500 tweets in total). Note that this leaves only 10 tweets from class 1 for training, further exacerbating the data imbalance problem.

As our approach is a ranking one, it is only capable of producing an ordered list and hence cannot predict absolute class labels. To enable a comparison with classification methods, we convert the resulting ranked list to a set of class labels, which can be trivial as the class labels represent sentiment scores from 5 to 1 (most positive to most negative) and our ranking approach is trained to rank based on sentiment scores (most positive to most negative). This is accomplished by simply sorting the balanced test set of 500 tweets from most positive to most negative using our ranking approach, dividing the ranked list into 5 segments and assigning labels (5 to 1) to each segment from top to bottom (*refer to the supplementary* ***video*** *for a clearer description of this process*). The issue with this process is that the converted results are always balanced, making the F_1 metric meaningless and as no thresholding is done, AUC cannot be calculated

Classification Model	Evaluation Metrics		
	Accuracy	F_1 Score	AUC
[40]	0.668	0.598	0.642
[41]	0.672	0.602	0.626
[39]	0.684	0.608	0.638
[38]	0.688	0.620	0.634
[47]	0.692	0.626	0.642
Converted Ranking w/ ALBERT	0.872	–	–
Converted Ranking w/ RoBERTa	0.902	–	–
Converted Ranking w/ GPT2	0.898	–	–
Converted Ranking w/ BERT	0.916	–	–

Table 4: Accuracy results of state-of-the-art classification models compared against the results of our ranking approach converted to class labels on [24].

either. However, since the test set follows a completely balanced distribution, accuracy provides an excellent metric for evaluating the performance. So while we report accuracy, F_1 and AUC for classification methods, our converted ranking results are only evaluated using accuracy (Table 4).

Note that this conversion is not necessary for a sentiment analysis task as the sentiment of a text is often subjectively measured against others, and is only done here to evaluate our approach in comparison to text classification methods. All models are trained for 100,000 steps. Table 4 shows how effective our ranking approach is by achieving very promising results despite the significant imbalance in the training dataset. Based on the F_1 and AUC scores of the state-of-the-art text classification methods [40, 41, 39, 38, 47], it is clear that the skew in the dataset has severely affected the learning capabilities of the classifiers while our ranking approach remains robust. In fact, the converted results of our best ranking approach improve on the most effective text classifier [47] by about 22%.

4.4 MS MARCO

While the main focus of this paper is not information retrieval, ranking is indelibly linked to information retrieval so much so that the metrics used to evaluate our ranking results are predominantly from the information retrieval literature. In this vein, we apply our approach to the publicly available benchmark dataset of MS MARCO [46]. We perform the passage ranking task within the benchmark using our pipeline. The queries and the passages are concatenated into sequences of no more 512 tokens. Due to hardware restrictions, less than 1% of the entire available dataset is used for training and the smallest possible Transformer versions are used. Table 5 demonstrates the results of our approach compared to contemporary ranking approaches on the validation set of the MS MARCO dataset. Despite using smaller Transformers and a fraction of the available training data, our approach produces promising results and remains competitive with information retrieval techniques.

4.5 Ablation Studies

To evaluate the importance of every component of our approach, we re-train our model as different components are removed or replaced and pair ranking accuracy is measured as the primary metric. 30% of the Stack Exchange dataset is selected as the dataset for our ablation experiments. As an important part of our pipeline is the Transformer generating the feature vector representing the input sequences, the choice of the Transformer is a critical one. We perform all our experiments with four commonly-used state-of-the-art Transformers. Table 6 demonstrates that BERT produces superior results with GPT2 remaining competitive despite its larger size. This is also supported by other experiments with results presented in Tables 2, 3, 4 and 5.

Another important component of the approach is the context aggregating multilayer perceptron that receives the sequence representations of the two input passages and generates the ranking scores. To evaluate its overall influence, the model is re-trained with a single linear layer replacing the entire MLP that maps the sequence representation into a single scalar value that is passed into the loss function as the ranking score. As seen in Table 6, while the approach still learns to rank the passages reasonably well without the MLP, the pair ranking accuracy significantly drops, emphasising the importance of this component of the approach.

Ranking Approach	Dev
	MRR@10
[48]	0.252
[49]	0.254
[50]	0.262
[51]	0.277
[52]	0.311
[53]	0.318
Our Approach w/ ALBERT	0.264
Our Approach w/ RoBERTa	0.267
Our Approach w/ GPT2	0.273
Our Approach w/ BERT	0.275

Table 5: Comparison of contemporary ranking methods and the proposed approach applied to MS MARCO. Note that unlike many comparators, the proposed approach uses the smallest most basic version of the Transformers.

Approach	Pair Labels
	Accuracy
Full Approach w/ ALBERT	0.861
Full Approach w/ RoBERTa	0.882
Full Approach w/ GPT2	0.899
Separate Transformers (BERT)	0.890
Approach w/o MLP (BERT)	0.815
Full Approach w/ BERT	**0.902**

Table 6: Pair-wise ranking accuracy results with varying components of the proposed ranking approach applied to a portion of our Stack Exchange data [23].

Within our pipeline, we opt for the use of a single Transformer to produce the embedding of both passages in the pair. One could envisage using two separate detached Transformers, each learning the representation of one of the input passages. While this will almost double the number of parameters and can potentially introduce training instabilities, one could imagine that the increased number of parameters will enhance the learning process. However, as seen in Table 6, the pair ranking accuracy is reduced slightly when separate Transformers are used for the passages. This is primarily due to the reduction in the number of training samples for each Transformer and the possible overfitting of each network. Additionally, by using the same network for all passages, the model will get a better sense of the entire dataset and can produce more robust representations.

5 Limitations and Future Work

While many text classification applications can be replaced by ranking with superior results, as ranking does not generally suffer from issues such as extreme data imbalances, text subjectivity and lack of context, this does not apply to all classification problems. There are certain situations where text classification remains the only option. For instance, a scenario wherein passages are meant to be categorised based on their topic cannot be addressed with ranking and requires a classification solution. Moreover, text ranking cannot deal in absolutes and is only capable of relatively comparing items. For example, while a ranking approach can provide an answer to whether a movie review is more or less positive than another, it cannot definitively say whether both the reviews are positive or negative. This can potentially be addressed in the future with a double-headed model that simultaneously ranks and classifies. As the two heads can correct each other during training, better representation learning and thus more accurate results can be expected, as well as the ability to provide an absolute relevance and sentiment value for each passage. Additionally, by improving the loss function to cope with lists rather than pairs, a better understanding of the context of the dataset can be obtained, leading to better results and fewer post-processing requirements.

6 Conclusion

We have investigated the applications of ranking in the world of natural language processing using a novel pair-wise ranking approach and different publicly-available datasets. Our text ranking approach makes use of state-of-the-art Transformers to generate a learned representation of a pair of text sequences. These representation vectors are subsequently used as the input to a context-aggregating multilayer perceptron, which combines the features representing the context and the content of the passages, assesses the relationship between the two passages and regresses to two values denoting the ranking scores subsequently used to rank the passages. The entirety of the model is trained end to end using a margin-based ranking loss. Experiments are carried out on four datasets, Stack Exchange, Fine Food Reviews, a dataset of tweets about self-driving cars and the MS MARCO passage ranking dataset. Passages are ranked according to different contexts, *e.g.* users posting the passages and the questions to which the passages are meant to respond. Experimental analyses demonstrate the effectiveness of text ranking for potential recommendation, forum moderation and security applications. We also make attempts to compare the results of our ranking approach directly with state-of-the-art classification techniques. A comparison of our ranking results converted to class labels with classification results shows an approximately 22% improvement, pointing to the efficacy of text ranking over text classification.
Please refer to the video (https://youtu.be/5GLZ9zH_hao) submitted as part of the supplementary material for a clear description of the approach/results.

7 Acknowledgment

This work was in part supported by the EPSRC EMPHASIS (EP/P01187X/1) and CRITiCaL (EP/M020576/1) projects.

8 Ethics Statement

As many day-to-day tasks in the modern world are automated using various machine learning models, the efficacy of such models becomes ever more important and influential in our lives. Many sentiment analysis, content tagging and fraud detection tasks are already being, or will soon be, carried out using NLP classification models. As such, the shortcomings of such classification models can greatly influence the quality of life and even the rights of individuals. By taking a first step towards offering an alternative to many classification tasks and removing the challenges associated with them, we believe the contribution of this work (exploring ranking as an alternative to classification) is ethical and can reduce the potential negative affects of many natural language processing systems in the future. Additionally, being able to track and predict the post quality (or any other attribute other than the quality) of users posting online can enable recommendation systems to suggest better content to users or security experts to detect malicious intent, which are also ethical applications of our work.

References

[1] Amir Atapour-Abarghouei, Stephen Bonner, and Andrew Stephen McGough. A King's ransom for encryption: Ransomware classification using augmented one-shot learning and bayesian approximation. In *IEEE Int. Conf. Big Data*, pages 1–6, 2019.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[3] Amir Atapour-Abarghouei and Toby P Breckon. Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 2800–2810, 2018.

[4] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[5] Amir Atapour-Abarghouei and Toby P Breckon. Veritatem Dies Aperit - temporally consistent depth prediction enabled by a multi-task geometric and semantic scene understanding approach. In *IEEE Conf. Computer Vision and Pattern Recognition*, pages 3373–3384, 2019.

[6] Amir Atapour-Abarghouei, Samet Akcay, Gregoire Payen de La Garanderie, and Toby P Breckon. Generative adversarial framework for depth filling via Wasserstein metric, cosine transform and domain transfer. *Pattern Recognition*, 91:232–244, 2019.

[7] Bruna G Maciel-Pearson, Samet Akçay, Amir Atapour-Abarghouei, Christopher Holder, and Toby P Breckon. Multi-task regression-based learning for autonomous unmanned aerial vehicle flight control within unstructured outdoor environments. *IEEE Robotics and Automation Letters*, 4(4):4116–4123, 2019.

[8] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019.

[9] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*, 2020.

[10] Joni Salminen, Vignesh Yoganathan, Juan Corporan, Bernard J Jansen, and Soon-Gyo Jung. Machine learning approach to auto-tagging online content for content marketing efficiency: A comparative analysis between methods and content type. *Journal of Business Research*, 101:203–217, 2019.

[11] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[12] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Int. Joint Conf. Natural Language Processing*, pages 1681–1691, 2015.

[13] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI Conf. Artificial Intelligence*, 2016.

[14] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. In *Int. Joint Conf. Artificial Intelligence*, pages 2915–2921, 2017.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[16] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150. Association for Computational Linguistics, 2011.

[17] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 115–124. Association for Computational Linguistics, 2005.

[18] Julian John McAuley and Jure Leskovec. From amateurs to connoisseurs: Modeling user expertise through online reviews. In *Int. Conf. World Wide Web*, pages 897–908, 2013.

[19] Jiahui Liu and Larry Birnbaum. Measuring semantic similarity between named entities by searching the web directory. In *Int. Conf. Web Intelligence*, pages 461–465. IEEE, 2007.

[20] Giordano Adami, Paolo Avesani, and Diego Sona. Clustering documents in a web directory. In *Int. Workshop on Web Information and Data Management*, pages 66–73, 2003.

[21] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. TF-Ranking: Scalable TensorFlow library for Learning-to-Rank. In *Int. Conf. Knowledge Discovery & Data Mining*, pages 2970–2978, 2019.

[22] Hang Li. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113, 2011.

[23] Inc. Stack Exchange. *Stack Exchange Data Dump*, 2019. `https://archive.org/details/stackexchange`.

[24] Twitter. *Twitter Sentiment Analysis: Self-Driving Cars*, 2017. `https://www.kaggle.com/c/twitter-sentiment-analysis-self-driving-cars`.

[25] Wassily W Leontief. The structure of american economy, 1919-1929. Technical report, Harvard University Press, 1941.

[26] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[27] K Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: Development and comparative experiments: Part 2. *Information Processing & Management*, 36(6):809–840, 2000.

[28] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*, volume 463. ACM Press New York, 1999.

[29] Norbert Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Trans. Information Systems*, 7(3):183–204, 1989.

[30] Fredric C Gey. Inferring probability of relevance using the method of logistic regression. In *Int. Conf. Research and Development in Information Retrieval*, pages 222–231, 1994.

[31] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Int. Conf. Machine Learning*, pages 89–96, 2005.

[32] Christopher JC Burges. From RankNet to LambdaRank to LambdaMART: An overview. *Learning*, 11(23-581):81, 2010.

[33] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Int. Conf. Machine Learning*, pages 129–136, 2007.

[34] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Int. Conf. Machine Learning*, pages 1192–1199, 2008.

[35] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The LambdaLoss framework for ranking metric optimization. In *ACM Int. Conf. Information and Knowledge Management*, pages 1313–1322, 2018.

[36] Yanyan Lan, Yadong Zhu, Jiafeng Guo, Shuzi Niu, and Xueqi Cheng. Position-aware ListMLE: A sequential learning process for ranking. In *Conf. Uncertainty in Artificial Intelligence*, pages 449–458, 2014.

[37] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. SoftRank: Optimizing non-smooth rank metrics. In *Int. Conf. Web Search and Data Mining*, pages 77–86, 2008.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019.

[39] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[40] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[41] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. ALBERT: A lite BERT for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[42] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *Int. Conf. Machine Learning*, 2016.

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[44] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8026–8037, 2019.

[45] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[46] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. MS MARCO: A Human Generated MAchine Reading COmprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

[47] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. XLnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5754–5764, 2019.

[48] Bhaskar Mitra and Nick Craswell. An updated duet model for passage re-ranking. *arXiv preprint arXiv:1903.07666*, 2019.

[49] Bhaskar Mitra, Corby Rosset, David Hawking, Nick Craswell, Fernando Diaz, and Emine Yilmaz. Incorporating query term independence assumption for efficient retrieval and ranking using deep neural networks. *arXiv preprint arXiv:1907.03693*, 2019.

[50] Corby Rosset, Bhaskar Mitra, Chenyan Xiong, Nick Craswell, Xia Song, and Saurabh Tiwary. An axiomatic approach to regularizing neural ranking models. In *Int. Conf. Research and Development in Information Retrieval*, pages 981–984, 2019.

[51] Rodrigo Nogueira, Jimmy Lin, and AI Epistemic. From doc2query to docTTTTTquery. *Online preprint*, 2019.

[52] Yifan Qiao, Chenyan Xiong, Zhenghao Liu, and Zhiyuan Liu. Understanding the behaviors of BERT in ranking. *arXiv preprint arXiv:1904.07531*, 2019.

[53] Sebastian Hofstätter, Navid Rekabsaz, Carsten Eickhoff, and Allan Hanbury. On the effect of low-frequency terms on neural-IR models. In *Int. Conf. Research and Development in Information Retrieval*, pages 1137–1140, 2019.

A Appendix

This brief appendix to the paper will outline technical details that had to be excluded from the original manuscript for the sake of brevity. The details will aid in a better understanding of the approach and reproducibility. Consequently, in Section A.1, we provide a detailed discussion on the *technical details that aid in the re-implementation* of the approach including the *software packages* used for implementation, details of *data selection and processing* and the *number of runs* for the experiments. Section A.2 will subsequently outline the details of the experiments leading to the *choice of the hyperparameters* used in our experiments.

A.1 Technical Implementation Details

This section contains technical details that will aid in the implementation of the approach. We kindly propose that the readers read this after the original manuscript. All implementation is done in Python 3.8.2 with PyTorch 1.4.1 [44] and Transformers 2.9.0. [43]. Note that the pre-trained Transformer weights from the Transformers library version 2.9.0. [43] are required and updated weights might lead to slightly different numerical results, even though they will support the same conclusions drawn in the paper (*i.e.* efficacy of ranking in natural language processing). The source code[2] has been tested using versions 2.9.0, 2.9.1 and 3.0.2 of the Transformers library [43].

All four datasets used in the paper, Stack Exchange 2019 [23], Fine Food Reviews [18], a dataset of tweets about self-driving cars [24] and the MS MARCO passage ranking dataset [46], are publicly available and can be downloaded freely. All the data for our experiments are read and processed using the Python Data Analysis Library (pandas 1.0.3).

As mentioned in the original paper, in certain experiments, smaller portions of these dataset are randomly selected and used due to their large size. In all of these instances, the Python Random Module is first used to shuffle the indices of the data points and the first n data points (as outlined in the descriptions of the experiments in the original manuscript) are subsequently selected for use. For the sake of consistency of to aid reproducibility, all random seeds for all libraries (Python Random, NumPy, PyTorch [44], etc.) in all primary experiments reported in the paper are set to 1.

In any experimental setup in deep learning, one would expect several runs of all algorithms (*e.g.* model training and experiments) with different seeds to obtain an average value and appropriate error metrics (*e.g.* variance) for cleaner and more accurate results without random initialisation conditions tainting the experimental results. However, this was not possible for our experiments, due to the large size of the models, large size of the datasets and intensive computational requirements. All ranking models trained for the experiments in this paper took between 6 to 18 days (each) to train to convergence.

This makes training every model several times with different seeds to remove the effects of random initialisation intractable, even for a modest number of runs. As a result, the results are reported after every model has only been trained to convergence once. It is important to note, however, that the Transformer networks are all trained from a fixed pre-trained state and the only portion of our overall models that actually starts from random initialisation is the four-layer context aggregating fully-connected network. This greatly reduces the effect of random initialisation on our approach as the size of the context aggregating multilayer perceptron pales in comparison to the size of the Transformer networks.

We can thus safely conclude that random initialisation does not have a significant effect on the results of our approach. This is empirically supported by the fact that the distances between the results of our experiments are large enough in all cases to remove any suspicion of initialisation conditions having any meaningful effect on the conclusions. The only case where the results of the experiments are close to each other is when different Transformers are used for the same task (especially GPT2 and BERT). However, even in the case of these results, we can see that they follow the same pattern across various different experiments with different datasets and even different tasks (as seen in Tables 2, 3, 4, 5 and 6 of the original manuscript), pointing to the fact that conclusions of the paper are valid across various datasets and various experimental conditions.

A.2 Choice of Hyperparameters

A very positive aspect of our approach is that there are very few hyperparameters associated with it. The only hyperparameters that might have an effect on the overall results of the approach in the experiments are the margin value in the loss function (γ) and the parameters used in the AdamW optimiser [45].

The margin in the loss function (γ) is the value by which the output ranking scores are forced to be distant from each other. This essentially ensures that the representations of the different inputs into the overall model (text sequences)

[2] https://github.com/atapour/rank-over-class

Margin Value	Pair Labels
	Accuracy
$\gamma = 0$	0.521 ± 0.082
$\gamma = 0.1$	0.688 ± 0.026
$\gamma = 1$	0.621 ± 0.025
$\gamma = 2$	**0.706 ± 0.031**
$\gamma = 5$	0.691 ± 0.023
$\gamma = 10$	0.686 ± 0.045

Table 7: Pair-wise ranking accuracy results with the margin value γ selected from values of {0, 0.1, 1, 2, 5, 10}.

learned by the model are sufficiently distant from each other in the hyperspace of all possible representations. If this margin value is set to zero, the input text sequences will be represented by the model as closely to each other as possible, but any value above zero should force a level of distance between different samples. We experiment with values {0, 0.1, 1, 2, 5, 10} to discover any existing patterns in the behaviour of the model with different margin values. As expected, zero is the wrong choice (Table 7) as we would like the learned representations of different inputs not to be similar to each other. We found that while the other choices in our set did not have a significant influence over the results, $\gamma = 2$ produced the best results.

To select the best margin value, γ, 10% of our Stack Exchange dataset is randomly selected as the training dataset and 2% as the test set. Every experiment is repeated three times with the seed value changing over {1, 2, 3} and the average and the standard deviation of the results is reported. The conditions of the experiment are the same as those described in the original paper. BERT is selected as the Transformer and the overall model is trained for 50,000 iterations during each run. Pair-wise ranking accuracy is measured as the primary metric, as a higher pair-wise accuracy will lead to better sorting performance and thus improved ranking and relevance metrics such as MRR, NDCG and MAP (used in the original paper).

The results of this experiment are seen in Table 7. As seen in Table 7, when $\gamma = 0$, the model is essentially guessing the ordering of the input, which is to be expected as the sequence representations are pushed towards each other, preventing any meaningful learning. As for the rest of the possible margin values, the results are all roughly the same with $\gamma = 2$ producing the best results. This experiment is repeated with multiple seeds and the very low standard deviation for the all the models with different margins further supports the claim made in Section A.1 of this document about the approach not being heavily dependant on initialisation conditions.

The only other hyperparameters used in our approach relate to the optimisation process. In the AdamW optimiser, there are four parameters:

- α: also known as the learning rate, determines how much the weights are updated at every iteration during training. This can be greatly influential as smaller values can delay or hinder convergence and larger values can make the optimiser leap over the optimum, thus completely preventing any learning.
- ϵ: is a very small value to prevent division by zero.
- β_1: is the exponential decay rate for the first moment estimates.
- β_2: is the exponential decay rate for the second-moment estimates. For tasks with sparse gradients (such as NLP problems), this value should be close to one

Previous work [45] has strongly demonstrated that the default values of $\beta_1 = 0.9$, $\beta_2 = 0.999$ produce the best results across a multitude of problem. Consequently, we use the same default values for our optimiser. As the only role of ϵ is preventing division by zero, its value will have no significant effect over the outcome of the optimisation. This makes the learning rate α the most important hyperparameter to tune.

To select the best learning rate, α, 10% of our Stack Exchange dataset is randomly selected as the training dataset and 2% as the test set. Every experiment is repeated once only with the seed value of 1. The conditions of the experiment are the same as those described in the original paper. The experiment is run using all four Transformers as the learning rate is heavily dependant on the architecture of the network. The overall model is trained for 50,000 iterations and pair-wise ranking accuracy is measured as the primary metric. We experiment with the gradually increasing values of {1e-6, 4e-6, 8e-6, 1e-5, 4e-5, 8e-5, 1e-4, 4e-4, 8e-4} to discover the best possible value of the learning rate within a reasonable interval for BERT [38], GPT2 [39], RoBERTa [40] and ALBERT [41]. The results of the experiment are seen in Table 8.

Learning Rate		Pair Labels
		Accuracy
ALBERT:	$\alpha = 1e{-}4$	0.508
ALBERT:	$\alpha = 4e{-}4$	0.512
ALBERT:	$\alpha = 8e{-}4$	0.531
ALBERT:	$\alpha = 1e{-}5$	0.548
ALBERT:	$\alpha = 4e{-}5$	0.502
ALBERT:	$\alpha = 8e{-}5$	0.494
ALBERT:	$\alpha = 1e{-}6$	0.526
ALBERT:	$\alpha = 4e{-}6$	**0.692**
ALBERT:	$\alpha = 8e{-}6$	0.686
RoBERTa:	$\alpha = 1e{-}4$	0.522
RoBERTa:	$\alpha = 4e{-}4$	0.490
RoBERTa:	$\alpha = 8e{-}4$	0.508
RoBERTa:	$\alpha = 1e{-}5$	0.537
RoBERTa:	$\alpha = 4e{-}5$	0.544
RoBERTa:	$\alpha = 8e{-}5$	0.530
RoBERTa:	$\alpha = 1e{-}6$	0.621
RoBERTa:	$\alpha = 4e{-}6$	**0.690**
RoBERTa:	$\alpha = 8e{-}6$	0.676
GPT2:	$\alpha = 1e{-}4$	0.551
GPT2:	$\alpha = 4e{-}4$	0.548
GPT2:	$\alpha = 8e{-}4$	0.688
GPT2:	$\alpha = 1e{-}5$	0.641
GPT2:	$\alpha = 4e{-}5$	**0.728**
GPT2:	$\alpha = 8e{-}5$	0.712
GPT2:	$\alpha = 1e{-}6$	0.688
GPT2:	$\alpha = 4e{-}6$	0.693
GPT2:	$\alpha = 8e{-}6$	0.662
BERT:	$\alpha = 1e{-}4$	0.486
BERT:	$\alpha = 4e{-}4$	0.532
BERT:	$\alpha = 8e{-}4$	0.527
BERT:	$\alpha = 1e{-}5$	0.708
BERT:	$\alpha = 4e{-}5$	**0.736**
BERT:	$\alpha = 8e{-}5$	0.698
BERT:	$\alpha = 1e{-}6$	0.719
BERT:	$\alpha = 4e{-}6$	0.686
BERT:	$\alpha = 8e{-}6$	0.680

Table 8: Pair-wise ranking accuracy results with the learning rate α.

It would be desirable to select the greatest possible learning rate without breaking the optimisation process completely as a larger learning rate will lead to faster convergence. A seen in Table 8, the best learning rate out of the tested options for ALBERT and RoBERTa is $4e - 6$ and for GPT2 and BERT $4e - 5$. It is important to note that slightly smaller learning rates may eventually lead to similar or even more accurate results but at a significantly slower rate.