Unsupervised Extreme Multi Label Classification of Stack Overflow Posts

Peter Devine
Kelly Blincoe
pdev438@aucklanduni.ac.nz
k.blincoe@auckland.ac.nz
The University of Auckland
Auckland, New Zealand

ABSTRACT

Knowing the topics of a software forum post, such as those on StackOverflow, allows for greater analysis and understanding of the large amounts of data that come from these communities. One approach to this problem is using extreme multi label classification (XMLC) to predict the topic (or "tag") of a post from a potentially very large candidate label set. While previous work has trained these models on data which has explicit text-to-tag information, we assess the classification ability of embedding models which have not been trained using such structured data (and are thus "unsupervised") to assess the potential applicability to other forums or domains in which tag data is not available.

We evaluate 14 unsupervised pre-trained models on 0.1% of all StackOverflow posts against all 61,662 possible StackOverflow tags. We find that an MPNet model trained partially on unlabelled StackExchange data (i.e. without tag data) achieves the highest score overall for this task, with a recall score of 0.161 R@1. These results inform which models are most appropriate for use in XMLC of StackOverflow posts when supervised training is not feasible. This offers insight into these models' applicability in similar but not identical domains, such as software product forums. These results suggest that training embedding models using in-domain title-body or question-answer pairs can create an effective zero-shot topic classifier for situations where no topic data is available.

ACM Reference Format:
Peter Devine and Kelly Blincoe. 2022. Unsupervised Extreme Multi Label Classification of Stack Overflow Posts. In *The 1st Intl. Workshop on Natural Language-based Software Engineering (NLBSE'22), May 21, 2022, Pittsburgh, PA, USA*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3528588.3528652

1 INTRODUCTION

StackOverflow [1] is an online forum containing questions and answers regarding programming. It has been widely used throughout the software engineering literature to study, amongst other things, how developers write code examples [30], how developers interact [42], how code gets reused [1], how to use StackOverflow posts to support code re-documentation [39], and how to use code from StackOverflow to direct automatic program repair [26]. Each question asked on StackOverflow has a title and body, which describe what the user is asking about, and a set of between one and five "tags", which describes the theme or topic of the question. These tags can be useful in analysing posts on a large scale. For example Abdalkareem et al. [1] filtered StackOverflow posts based on the "Android" tag to investigate code reuse from the platform.

Automated approaches have been proposed for automatically applying tags to posts [25, 36]. Many standard machine learning classifiers exist within the literature which are trained to classify posts into a small subset of the most popular tags available [18, 21, 43]. However, if classification is not constrained to just the most popular tags, then predicting tags of a post becomes an extreme multi label classification (XMLC) problem, in which a potential label set of thousands of tags are considered when classifying [45]. Recent work has trained text embedding models on text-tag pairs to classify text using tags that were unseen at training time [23]. However, this approach still requires a large amount of text-to-tag training pairs and may not be able to be applied to new platforms or domains.

App store reviews [8, 28], software subreddit Reddit posts [2], and product user forums [38] have all been shown to contain information that could be relevant to the software development cycle, and thus could be used to improve the product overall. However, these data sources do not contain explicit tags of all the topics discussed within the posts. Being able to classify tags or topics to posts on these other platforms would enable more effective filtering and analysis of this valuable information. A model that can accurately classify tags to StackOverflow posts without being trained on StackOverflow tag data could open up possibilities for other platforms.

We extend previous work by using publicly available embedding models which have not been explicitly trained to embed tags to questions to do this classification. Our approach embeds each post and each possible tag name using a pre-trained model, and then compares the similarity of each embedding to create a ranking across the label set for each post. We evaluate across a wide variety of models to determine which are best able to classify correct tags to a given post. We also provide an evaluation focused on "rare" tags that are seldom used within our dataset to demonstrate how these models perform in situations where training data would not be available, and thus supervised classifiers could not be applied.

We report which models achieve the highest performance for fully unsupervised XMLC of StackOverflow posts. Our findings indicate that training an embedding model on pairs of text from the

[1] https://stackoverflow.com/

NLBSE'22, May 21, 2022, Pittsburgh, PA, USA

same data domain as the intended XMLC task creates a superior XMLC model. These findings have implications for topic classification and topic search in data sources outside of StackOverflow.

This work is guided by the following research question:

> **RQ -** Which unsupervised model is best able to perform XMLC on StackOverflow data?

The replication package for these experiments has also been made available online [2].

2 BACKGROUND

StackOverflow has been described within the literature as *"one of the most visible venues for expert knowledge sharing around software development"* [29]. This has lead researchers to study the topics and content of what this large community of programmers are saying [4]. Due to the fact that the tags associated with any StackOverflow post may not be exhaustive, machine learning systems have been employed to extract generic topics contained within these discussions. Examples include using Latent Dirichlet allocation (LDA) to extract topics from posts which have already been filtered by using the `machine_learning` tag [3]. Thus, much work has been done on automatically classifying all the tags applicable to a post.

Initial approaches for classifying tags of StackOverflow posts, such as those by Kuo [25] and by Schuster et al. [36], included using Bayesian word co-occurrence models, which find the probability that each word from a post appears individually in the post from a given tag. These probabilities are modified by the tags global frequency, and are then used to predict the most probable tags given a piece of text.

Other early classification methods involved classifying posts into a small tag subset (i.e. hundreds of the most common tags). This includes training classical machine learning models (e.g. Bayesian co-occurrence, k-NN, SVM, NNS) on bag-of-words (BoW) or term frequency inverse document frequency (TF-IDF) features to classify into a relatively small tag subset, such as in work by Hong and Fang [21], González et al. [18], and Wang et al. [43].

Neural networks have also been used to classify posts, with the TagCNN, TagRNN, TagHAN, and TagRCNN models presented by Zhou et al. [46], which are convolutional neural networks (CNNs) or recurrent neural networks (RNNs) which generate embedding representations of posts that are passed to a softmax layer which classifies over tens of thousands of tags. Another example of this include Post2Vec by Xu et al. [45], which are a series of CNNs which are similarly trained as above (with a sigmoid layer instead of a softmax classification layer), but also include code snippet information within the generated representation.

Such embedding approaches have been further improved upon by training neural networks to generate embeddings of both text and tag. Nie et al. present a model which trains embeddings of posts to be closer in embedding space to embeddings of tags associated with a post than those not associated [31]. This is done by preprocessing the posts and tags separately, generating vectors for both, then calculating the dot product between these vectors. This dot product result is then passed to a sigmoid cross entropy loss function, with the labels supplied to the loss function being 1 if the tag in question was given to the post, and 0 otherwise. In contrast to previous approaches, these embeddings allow for fast comparison between a piece of text and many candidate tags, which in turn enables comparison of a large amount of posts and classes at once.

The above approaches all rely on training on a set of tags and then predicting tags at inference time only out of those included at training time. Jain and Roy propose a zero-shot setting for StackExchange forum classification [23], in which skip-gram embeddings are first trained using a selection of posts to tags. Tags are then split into "seen" and "unseen", with the former being used for training a standard text classifier into one of the seen tag classes. An embedding similarity matrix created between the seen and unseen tags, and posts that are classified as being one of a set of seen tags are also classified as being one of a set of similar unseen tags. This approach allows for new tags to be considered at inference time, but still requires much data for creating the initial skipgram embeddings. Moreover, since they focus on three smaller StackExchange subfora, there are a maximum of 1,895 classes considered at any one time for classification, which is much less than the potential topics available on larger StackExchange forums such as StackOverflow. With a smaller class set, it becomes more difficult to predict the ability of models to deal with "one-off" unique tags or topics, which would appear in the real world.

A model for generalized XMLC was created by Gupta et al. [20]. This model is efficiently trained on 31 million labels from four data sources (legal data, Amazon review data, Wikipedia data, proprietary Bing Ad search data) to create a model that can classify text on previously unseen labels. Due to the fact that this model has not been tested on StackOverflow data, it is unclear how well it would perform in the StackOverflow XMLC task.

General embedding models are a potential candidate for use in the XMLC problem. Models such as SBERT[34] and USE[7] have been trained on a wide variety of paired text datasets such that they produce embeddings that demonstrate good performance when used on a range of down-stream tasks, such as in the Semantic Text Similarity Benchmark[6]. Due to their relative generality, these models may be well placed to perform zero-shot tasks. It remains an open question as to whether these models would provide a good embedding base from which XMLC could be performed on the StackOverflow tag prediction task.

To the authors' knowledge, there has been no prior work within the literature which seeks to evaluate the StackOverflow XMLC effectiveness of embedding models which have not been trained on the StackOverflow post to tag classification problem. In this paper, we aim to fill this gap by examining whether it is possible to perform XMLC on software engineering artifacts without a supervised label-set data signal.

3 METHOD

3.1 Our data

The data used in this study was downloaded from a StackOverflow mirror available on archive.org [3]. This data was unzipped, and the posts data (Posts.xml) was used as our dataset. We selected only the

[2] https://doi.org/10.5281/zenodo.5880185

[3] https://archive.org/download/stackexchange/stackoverflow.com-Posts.7z

question posts (i.e. no comments or answers) for our evaluation, as has been done in the previous literature. We focused on analysing posts only from the StackOverflow forum as it is the biggest on StackExchange and the most studied throughout software engineering literature.

From the 21,641,802 posts downloaded, we found 61,662 unique tags. Due to computational limitations, a subset of the 21,641,802 posts was selected such that all posts could be loaded into memory at once on the computer being used. The entire dataset was iterated over, one post at a time, with each post having a 0.1% chance of being added to a dataset for further classification. The 0.1% figure was chosen as the highest order of magnitude that could be loaded easily into memory. This resulted in a classification dataset of 217,174 individual posts containing 27,092 unique tags.

In our results, we further break down our dataset into an "unpopular" subset, in which we select only posts which contain at least one unique tag within the dataset (i.e. only selecting posts which contains a tag which is only used once within our dataset). This resulted in a subset of 9,740 posts that contain tags that would be difficult for a standard supervised classifier to classify due to a paucity of training data. There is a "long tail" of unpopular posts within the dataset we studied, with 21,765 out of 27,092 unique tags being used in less than 10 posts, which represents 55,683 (25%) total posts. This shows the importance of being able to classify rare or unseen topics, and so this evaluation is also reported to contextualise the evaluated model's performance in a scenario unsuited to standard supervised classifiers.

3.2 Pre-processing

Tag text was cleaned by replacing all hyphens in text with spaces (e.g. "`amazon-s3`" becomes "`amazon s3`") to make tags appear more like natural language, which most evaluated models have been exclusively trained on.

Only text from the question body was used in our evaluation. Because the post data as available on archive.org is supplied in HTML format, text was extracted from the post body using the BeautifulSoup HTML parser. During this process, any `<code>` and `<blockquote>` tags were removed from the body before extracting all text within the remaining HTML.

3.3 Evaluation

An illustration of our zero shot classification evaluation method can be found in Fig 1.

All posts were converted from text into an embedding using an embedding model. All possible tags (61,662) were also converted from text into an embedding. These two sets of embeddings were then compared using a similarity function to create a similarity matrix between all posts and all tags. Tags were then ranked based on similarity for each post, with the more similar tags being ranked higher. The top 10 rankings for each post were then compared to the true labels given to each post. Evaluation metrics were then generated.

Multiple embedding models were tested, and are explained in more detail in section 3.4. The similarity function used within this paper was cosine similarity due to the fact that some models tested (such as **MPNet (All)** and **CLIP**) were trained to explicitly model

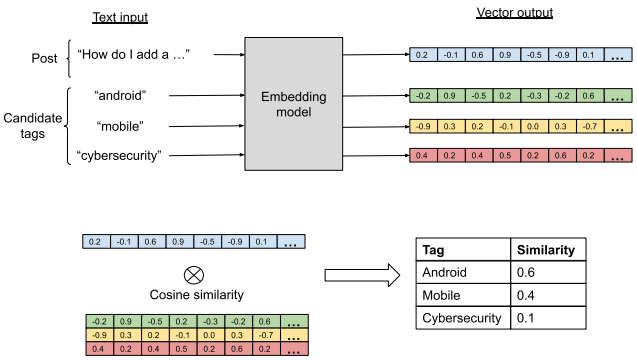


Figure 1: Example of how the post text and tags are compared to generate a ranking of tag recommendations based on similarity score.

semantic similarity in text through cosine distance between text. Early experiments indicated that using cosine, Euclidean, or Manhattan distance measures did not greatly affect the final rankings of tags. The evaluation metrics used in this work are the same as those used by Xu et al. [45] and Jain and Roy [23]. We report the recall, precision, and F1 of the recommendations generated at different recommendation sizes. This gives the accuracy (and hence usefulness) of our XMLC method when outputting n recommendations. We choose to report evaluation metrics for between 1-10 recommendations. Notation is standard, with precision over 3 recommendations being denoted as P@3, for example.

3.4 Models (and their datasets)

Various models were used to create embeddings for comparing post text to tags. These are all derived from deep Transformer [40] based models which have been pre-trained on a large set of data. The pre-training tasks of the models evaluated are broadly in two categories: language modelling and similarity embedding. Language modelling trains a model to re-write a known passage of text given that some of it is hidden to the model, and this can be performed on any format of free text. Similarity embedding trains a model to embed two pieces of text into a close embedding space if they are labelled as "matches" and in a distant embedding space if not. A model can be trained using this task as long as similar and dissimilar text pairs are available, which could be questions and answers, post titles and bodies, abstracts and articles, or some other similarity signal. Outside of these two tasks, the Universal Sentence Encoder (USE) [7] is trained in a multi-task setting, with the tasks including a SkipThought like task [24], a conversational input-response task, and various classification tasks. This multi-task pre-training is done to create a general embedding of a piece of text for use in any of a multitude of different situations. Once pre-trained, models all take text as an input and output a vector or set of vectors which represent some of the semantic content of the text, and these vectors are used as an embedding which can then be compared.

	Source	Source ID	Cite	Training dataset	Training task	Base model
MPNet (All)	sbert.net	all-mpnet-base-v2	[34]	"All" dataset	Similarity task with text pairs	MPNet[37]
MiniLM (All)		all-MiniLM-L6-v2				MiniLM[44]
MPNet (QA)		multi-qa-mpnet-base-dot-v1		"QA" dataset		MPNet[37]
MiniLM (QA)		multi-qa-MiniLM-L6-dot-v1				MiniLM[44]
LaBSE		LaBSE	[16]	CommonCrawl, Wikipedia, and multilingual text pairs mined online		None
SPECTER		allenai-specter	[9]	Pairs of scientific papers that are close or distant on the citation graph		None
CLIP (Multilingual)		clip-ViT-B-32-multilingual-v1	[32]	Image and caption pairs	Similarity task with text and image pairs	None
USE (Large)	TFHub	google/universal-sentence-encoder-large/5	[7]	Wikipedia, web news, web question-answer pages, discussion forums, and SNLI[5]	Assorted tasks to create general embedding	None
USE (Base)		google/universal-sentence-encoder/4				
BERT (Base uncased)	Huggingface	bert-base-uncased	[12]	Wikipedia, Book Corpus[47], CommonCrawl	Masked language modelling	None
BERT (Base cased)		bert-base-cased				
DistilBERT (Base cased)		distilbert-base-uncased	[35]			None
CodeBERT		microsoft/codebert-base	[17]	CodeSearchNet challenge dataset[22]		None
Graph CodeBERT		microsoft/graphcodebert-base	[19]	CodeSearchNet challenge dataset[22]		CodeBERT[17]

Table 1: Description of all models used in our evaluation. Base model details the original model which was used upon which the resulting model was trained using the training task.

A selection of models available on sbert.net [4] [34], Huggingface [5], and TF Hub [6] were evaluated.

The criteria for selection of our models was based on a mixture of choosing models with high performance on the sbert.net semantic search and sentence embedding leaderboard [7], speed, and training diversity. This selection was intentionally diverse to aid in finding the highest performing model on our task.

Two of the models evaluated, **MPNet (QA)** and **MPNet (All)**, were based on MPNet [37] and were both trained using siamese networks (as was done by S-BERT [34]) but with different datasets. These two models were the highes performing on the aforementioned sbert.net leaderboard. One was trained using a "QA" dataset, which contains text pairs from a variety of sources, including WikiAnswers [14], Amazon product pages [41], SQuAD2.0 [33], and StackExchange. The StackExchange data makes up 47,017,540 text pairs out of 214,988,242 total training pairs (22%) of the whole dataset, which in turns consists of 25,316,456 (12%) StackExchange post title-body pairs, 21,396,559 (10%) title-answer pairs, and 304,525 (<1%) title-title duplicate post pairs [8]. The second MPNet-based model was trained using an "all" dataset, which contains all of the data in the "QA" dataset, plus other text pair data from different sources, including Semantic Scholar (title and abstract pairs) [27] and Wikipedia (English article and simple English article [10]). This dataset is much larger (1,170,060,424 training pairs), meaning that StackExchange data is a lower share of total training data in this dataset [9].

Two similar models, **MiniLM (QA)** and **MiniLM (All)**, were also evaluated, and are based on the 6 layer variant of the MiniLM model [44], again with variants being trained on either the "QA" or "all" datasets. These model variants perform slightly worse the the aforementioned sbert.net leaderboard, but are much smaller models, consequently embedding text faster. These models are included as a faster alternative model in our evaluation.

[4]https://www.sbert.net/docs/pretrained_models.html
[5]https://huggingface.co/models
[6]https://tfhub.dev/
[7]Provided by sbert.net at https://www.sbert.net/docs/pretrained_models.html#model-overview

[8]https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1 - Archived on 24 November 2021 on the WayBack Machine (https://archive.org/web/)
[9]https://huggingface.co/sentence-transformers/all-mpnet-base-v2 - Archived on 12 January 2022 on the WayBack Machine (https://archive.org/web/)

Both the base **USE (Base)** [10] and large **USE (Large)** versions [11] of USE [7] were also evaluated. The training data for the USE model is only given as being sourced from "Wikipedia, web news, web question-answer pages and discussion forums", and is augmented with the SNLI dataset [5]. Therefore, we do not know specifically what these models were trained on. This model has been used within the Software Engineering literature before with promising results [11, 13], and thus was included in our evaluation.

For other embedding models, we also investigated the performance of the sbert.net version of **SPECTER** [9], the multilingual version of **CLIP** [32], and the sbert.net version of **LaBSE** [16]. These models have been trained on scientific citations, image and text pairs, and multilingual text pairs respectively, representing a distinct set of training data compared to the above models.

Outside of embedding models, we also evaluated the averaged outputs of several standard language models (i.e. models that were not trained to create good text embeddings specifically). This included 2 **BERT** [12] variants, **distilBERT** [35], **CodeBERT** [17], and **GraphCodeBERT** [19]. The output of these models was averaged across all token positions to create a standard length embedding for each piece of text.

A basic baseline was also included within the evaluation to contextualise results. This baseline is the 10 most frequent tags from our dataset recommended for every post. This was chosen as a simple, naive baseline against which to test, effectively serving as a majority baseline for the recommendations.

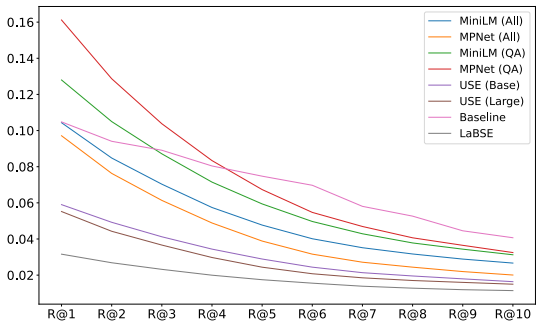
Full details of the models can be found in Table 1.

4 RESULTS

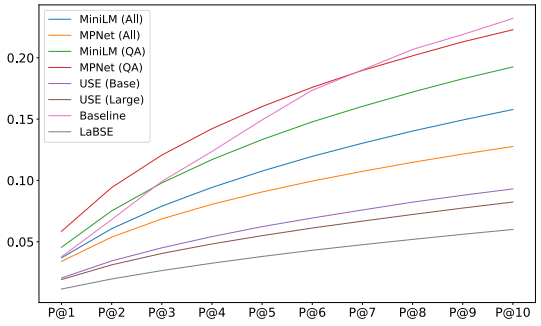
The recall of each model is listed in Table 2, while plots of recall, precision, and F1 score at different recommendation sizes can be seen in Fig 2 for all models with a non-negligibly small performance. We can see that the MPNet and MiniLM based models outperform all other models significantly. Specifically, the models trained on the "QA" datasets are best performing, with the **MPNet model (QA)** best out of these. This model outperforms the baseline for P and R @1-4, and has a higher maximum F1 score. The **MPNet (QA)** model has a maximum recall value at R@1 of 0.161, meaning that roughly 16% of all top recommended tags were in the list of tags for the given StackOverflow post. For F1 score, the **MPNet (QA)** model performs best at F1@3, with a score of 0.112 (compared to the baseline value of 0.094), and the baseline performs best at F1@5, with a score of 0.097 (compared to the **MPNet (QA)** score of 0.095). This is best illustrated in Fig 2, where **MPNet (QA)** (red line) has a higher precision, recall, and F1 for small recommendation sizes, while the naive baseline (pink line) exceeds it at larger recommendation sizes. The non-embedding trained models, as well as **CLIP**, **LaBSE** and **SPECTER**, all performed poorly on this task, with recall never rising above 0.053 for any of these models.

The graphs of precision, recall, and F1 score for the highest scoring models on the "unpopular" subset are plotted in Fig 3. Again, both QA-dataset trained models outperformed all other models, but also outperformed the baseline on R@1-10.

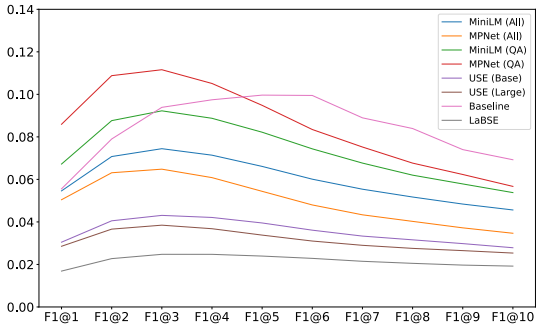
[10] https://tfhub.dev/google/universal-sentence-encoder/4
[11] https://tfhub.dev/google/universal-sentence-encoder-large/5



(a) Recall



(b) Precision



(c) F1

Figure 2: Evaluation metrics @1-10 on the whole evaluation dataset. MPNet (QA) performs best of all models across all metrics, and outperforms the naive baseline for small recommendation sizes.

RQ - Which unsupervised model is best able to perform XMLC on StackOverflow data?
A - We find both models that were trained mainly on StackOverflow text pairs, **MPNet (QA)** and **MiniLM (QA)**, perform better at XMLC on StackOverflow posts than all other models evaluated. Overall accuracy is poor over the large label set evaluated.

	R@1	R@2	R@3	R@4	R@5	R@6	R@7	R@8	R@9	R@10
Baseline	0.105	0.094	0.089	0.080	**0.075**	**0.070**	**0.058**	**0.053**	**0.045**	**0.041**
MPNet (QA)	**0.161**	**0.129**	**0.104**	**0.083**	0.067	0.055	0.047	0.041	0.036	0.032
MiniLM (QA)	0.128	0.105	0.087	0.071	0.059	0.050	0.043	0.038	0.034	0.031
MiniLM (All)	0.104	0.085	0.070	0.057	0.048	0.040	0.035	0.032	0.029	0.027
MPNet (All)	0.097	0.076	0.061	0.049	0.039	0.032	0.027	0.024	0.022	0.020
USE	0.059	0.049	0.041	0.034	0.029	0.024	0.021	0.020	0.018	0.016
USE (Large)	0.055	0.044	0.037	0.030	0.024	0.021	0.019	0.017	0.016	0.015
LaBSE	0.032	0.027	0.023	0.020	0.017	0.016	0.014	0.013	0.012	0.011
SPECTER	0.014	0.011	0.009	0.008	0.007	0.006	0.005	0.005	0.004	0.004
CLIP (Multilingual)	0.002	0.002	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
Bert (Base uncased)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
DistilBERT (Base cased)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
GraphCodeBERT	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Bert (Base cased)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
CodeBERT	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 2: Recall@1-10 for each model evaluated. The QA dataset-trained MPNet model performs better than all other models and the baseline for recall @1-4. The baseline performs better than all models evaluated for recall @5-10. Bolded values are the best performing scores across all models and baseline for a given metric.

5 DISCUSSION

The results suggest that the **MPNet (QA)** and **MiniLM (QA)** models are best able to classify StackOverflow posts over a large potential label set out of all models evaluated. These models modestly outperform a naive baseline over the whole dataset for a small number of recommendations, but perform better and outperforms the baseline more convincingly on less frequently used classes.

We can also note that the models that were not explicitly trained to embed text (**BERT**, **DistilBERT**, **CodeBERT**, **GraphCodeBERT**) performed poorly on the XMLC task. This is not surprising as they have been trained to model language at a token level rather than create a holistic embedding of a piece of text.

One finding of the results is that all models trained on StackExchange data (title-body pairs, question-answer pairs, duplicate question pairs) outperform all other embedding models. Indeed, the models trained solely on the "QA" dataset, which included the StackExchange data, outperformed models trained on that plus other data. This suggests that models trained on in-domain paired data (in our case, StackExchange post data) are effective at XMLC, even when they have not been explicitly trained on using tags for an XMLC task. There are many sources of data online which do not have topic tags in the way that StackExchange has them, but do contain possible text pairs (title-body, question-answer, duplicate questions) on which a model could be trained. Examples of this include the app reviews and forum posts. Future work could explore the possibility of creating such an embedding model using text pairs from specific data sources for their use in XMLC or topic searching on that data source.

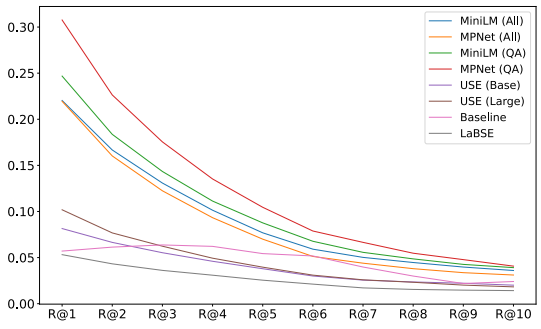
The gross performance values of our evaluation are relatively low (not exceeding 0.2 F1 score in any evaluation), and so these results need to be considered in relation to the setting of this task. For a classification task in which supervised training data is available, the zero shot methods evaluated in this paper could not compete with most trained classifiers. However, the fact that 16% of all top recommendations are valid for our top performing model is noteworthy, given the fact that this is a lower bound. This lower bound is due to the fact that nearly all tags of a post are relevant, but not all tags that weren't applied to a post are irrelevant. Therefore, while this XMLC method should not be applied in settings where topic data is already available (e.g. tags for StackOverflow posts), it may be useful in data domains in which this topic data is not available.

While classification is the main focus of this work, this could also illuminate which embedding models create a good general semantic representation of a post, which could then be used in similarity detection, clustering, or outlier detection. Future work could evaluate the effectiveness of the models evaluated in this work, or models evaluated on text-pairs from other software engineering artifact sources, on these down-stream tasks.

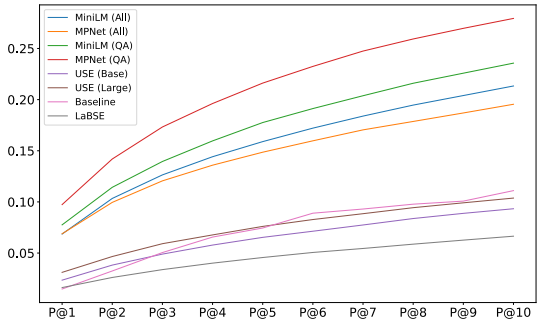
Another consideration of this work is that we provide a much more difficult XMLC task compared to previous work. We consider all 61,662 unique tags used on StackOverflow at the time of writing on a random sample of all StackOverflow posts. This is in contrast to Xu et al. [45], which identified 29,357 "rare" tags which were used less than 50 times throughout all StackOverflow posts, and removed all posts which only contained "rare" tags. The use of all available tags classifying on a random sample of posts is a realistic setting, especially due to the fact that as new technology becomes available, new tags will be used on the StackOverflow site. Thus, our results reflect how a model would perform in this zero-shot setting as well as in settings where training data would otherwise be available.

6 THREATS TO VALIDITY

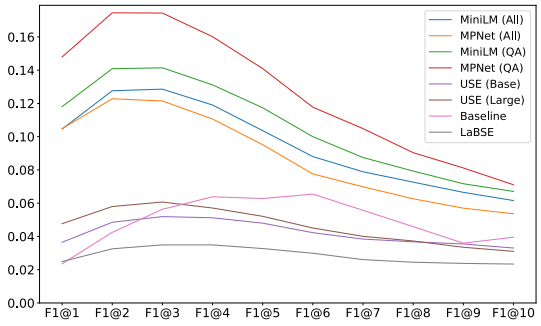
A threat to the validity of these results is that not all models from the literature were tested. Models such as GMXML [20] and SE_Bert [15] were considered, but were not readily available online as a pre-trained model. As wide a variety of seemingly applicable



(a) Recall



(b) Precision



(c) F1

Figure 3: Evaluation metrics @1-10 on the rare tag evaluation data subset. MPNet (QA) performs better than all other models and the naive baseline across all metrics.

and available models as possible was used in our evaluation, but it remains for future work to add new models to this evaluation as they become available.

Another threat to validity is the fact that we only included the post body when performing classification, ignoring the code and title associated with each post. Code was excluded as most models evaluated were not trained on source code, and so would not necessarily be able to make a meaningful embedding using it. Titles were excluded as it was not clear how best to combine this data with the body text. However, work by Xu et al. notes that including code and title information as features for a supervised classifier improves the performance of that classifier overall [45]. It remains for future work to explore whether incorporating title and code data to post body text changed the classification performance of the models tested.

Finally, another threat to validity of these results is the possibility of false negatives. Due to the fact that we consciously included as many candidate classes as possible (more than 61,000 possible tags) to capture the diversity of topics within software forums, we evaluated posts on many synonymous tags. For example, recommended tags of `python` and `machine learning` on a post with only `python 3.x` and `neural networks` real tags would both be classified as incorrect recommendations. However, Python is a superset of Python 3.x and neural networks are a form of machine learning, making both recommendations accurate to the post. Conversely, we expect relatively few false positives, since each tag is labelled directly by the author of a post on StackOverflow. Therefore, we expect the recall values listed in this work to be a lower bound of their true usefulness. It remains for future work to determine the false negative rate of this evaluation.

7 CONCLUSION

In this work, we have evaluated the ability of pre-trained deep text embedding models to perform XMLC of tags for StackOverflow posts.

We find that out of all models evaluated, the **MPNet (QA)** and **MiniLM (QA)** models performed best, with R@1 scores over the entire evaluation dataset of 0.161 and 0.128 respectively. These models were both trained on a variety of data, including text pairs derived from StackOverflow posts, but crucially not containing any post tag data.

Our results give insight into which embedding models are most appropriate for use in classifying tech forum posts when training is unfeasible. Particularly, our results suggest that training an embedding model using in-domain data (such as title-body pairs or question-answer pairs) creates a better XMLC model than general text embedding models. This finding could be applied to software engineering natural language artifacts which do not have tag data available, but do have titles or answers, such as app reviews and forum posts. Future work could determine whether such an embedding model would able to effectively classify topics of these artifacts without training explicitly on tag data at a higher performance than baseline methods.

REFERENCES

[1] Rabe Abdalkareem, Emad Shihab, and Juergen Rilling. 2017. On code reuse from stackoverflow: An exploratory study on android apps. *Information and Software Technology* 88 (2017), 148–158.

[2] Javed Ali Khan, Lin Liu, Lijie Wen, and Raian Ali. 2020. Conceptualising, extracting and analysing requirements arguments in users' forums: The CrowdRE-Arg framework. *Journal of Software: Evolution and Process* 32, 12 (2020), e2309.

[3] Abdul Ali Bangash, Hareem Sahar, Shaiful Chowdhury, Alexander William Wong, Abram Hindle, and Karim Ali. 2019. What do developers know about machine learning: a study of ML discussions on StackOverflow. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 260–264.

[4] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. 2014. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering* 19, 3 (2014), 619–654.

[5] Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*. Association for Computational Linguistics (ACL), 632–642.

[6] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055* (2017).

[7] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).

[8] Ning Chen, Jialiu Lin, Steven CH Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. AR-miner: mining informative reviews for developers from mobile app marketplace. In *Proceedings of the 36th international conference on software engineering*. 767–778.

[9] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S Weld. 2020. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2270–2282.

[10] William Coster and David Kauchak. 2011. Simple English Wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 665–669.

[11] Souvick Das, Novarun Deb, Agostino Cortesi, and Nabendu Chaki. 2021. Sentence Embedding Models for Similarity Detection of Software Requirements. *SN Computer Science* 2, 2 (2021), 1–11.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[13] Ayesha Enayet and Gita Sukthankar. 2020. A Transfer Learning Approach for Dialogue Act Classification of GitHub Issue Comments. *arXiv preprint arXiv:2011.04867* (2020).

[14] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1156–1165.

[15] Eliane Maria De Bortoli Fávero and Dalcimar Casanova. 2021. BERT_SE: A Pre-trained Language Representation Model for Software Engineering. *arXiv preprint arXiv:2112.00699* (2021).

[16] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding. *arXiv preprint arXiv:2007.01852* (2020).

[17] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. CodeBERT: A Pre-Trained Model for Programming and Natural Languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 1536–1547.

[18] José R Cedeno González, Juan J Flores Romero, Mario Graff Guerrero, and Felix Calderón. 2015. Multi-class multi-tag classifier system for stackoverflow questions. In *2015 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*. IEEE, 1–6.

[19] Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, LIU Shujie, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, et al. 2020. GraphCodeBERT: Pre-training Code Representations with Data Flow. In *International Conference on Learning Representations*.

[20] Nilesh Gupta, Sakina Bohra, Yashoteja Prabhu, Saurabh Purohit, and Manik Varma. 2021. Generalized Zero-Shot Extreme Multi-label Learning. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 527–535.

[21] James Hong and Michael Fang. 2013. Keyword extraction and semantic tag prediction. *unpublished(http://cs229. stanford. edu/proj2013/FangHong-Keyword% 20Extraction% 20and% 20Semantic% 20Tag% 20Prediction. pd f)* (2013).

[22] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. CodeSearchNet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436* (2019).

[23] Yash Jain and Anurag Roy. 2021. Distributed representation of tags for Active Zero Shot learning. In *8th ACM IKDD CODS and 26th COMAD*. 228–232.

[24] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. 3294–3302.

[25] Darren Kuo. 2011. On word prediction methods. *Technical report, Technical report, EECS Department* (2011).

[26] Xuliang Liu and Hao Zhong. 2018. Mining stackoverflow for program repair. In *2018 IEEE 25th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 118–129.

[27] Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel S Weld. 2020. S2ORC: The Semantic Scholar Open Research Corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 4969–4983.

[28] Walid Maalej, Zijad Kurtanović, Hadeer Nabil, and Christoph Stanik. 2016. On the automatic classification of app reviews. *Requirements Engineering* 21, 3 (2016), 311–331.

[29] Lena Mamykina, Bella Manoim, Manas Mittal, George Hripcsak, and Björn Hartmann. 2011. Design lessons from the fastest q&a site in the west. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 2857–2866.

[30] Seyed Mehdi Nasehi, Jonathan Sillito, Frank Maurer, and Chris Burns. 2012. What makes a good code example?: A study of programming Q&A in StackOverflow. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*. IEEE, 25–34.

[31] Liqiang Nie, Yongqi Li, Fuli Feng, Xuemeng Song, Meng Wang, and Yinglong Wang. 2020. Large-scale question tagging via joint question-topic embedding learning. *ACM Transactions on Information Systems (TOIS)* 38, 2 (2020), 1–23.

[32] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020* (2021).

[33] Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 784–789.

[34] Nils Reimers, Iryna Gurevych, Nils Reimers, Iryna Gurevych, Nandan Thakur, Nils Reimers, Johannes Daxenberger, Iryna Gurevych, Nils Reimers, Iryna Gurevych, et al. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

[35] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[36] Sebastian Schuster, Wanying Zhu, and Yiying Cheng. 2017. Predicting tags for stackoverflow questions. In *Proceedings of the LWDA 2017 Workshops: KDML, FGWM, IR, and FGDB*.

[37] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297* (2020).

[38] James Tizard, Hechen Wang, Lydia Yohannes, and Kelly Blincoe. 2019. Can a conversation paint a picture? mining requirements in software forums. In *2019 IEEE 27th International Requirements Engineering Conference (RE)*. IEEE, 17–27.

[39] Carmine Vassallo, Sebastiano Panichella, Massimiliano Di Penta, and Gerardo Canfora. 2014. Codes: Mining source code descriptions from developers discussions. In *Proceedings of the 22nd International Conference on Program Comprehension*. 106–109.

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

[41] Mengting Wan and Julian McAuley. 2016. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE, 489–498.

[42] Shaowei Wang, David Lo, and Lingxiao Jiang. 2013. An empirical study on developer interactions in stackoverflow. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. 1019–1024.

[43] Shaowei Wang, David Lo, Bogdan Vasilescu, and Alexander Serebrenik. 2018. EnTagRec++: An enhanced tag recommendation system for software information sites. *Empirical Software Engineering* 23, 2 (2018), 800–832.

[44] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *arXiv preprint arXiv:2002.10957* (2020).

[45] Bowen Xu, Thong Hoang, Abhishek Sharma, Chengran Yang, Xin Xia, and David Lo. 2021. Post2vec: Learning distributed representations of Stack Overflow posts. *IEEE Transactions on Software Engineering* (2021).

[46] Pingyi Zhou, Jin Liu, Xiao Liu, Zijiang Yang, and John Grundy. 2019. Is deep learning better than traditional approaches in tag recommendation for software information sites? *Information and software technology* 109 (2019), 1–13.

[47] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*. 19–27.