Assessing Text Representation Methods on Tag Prediction Task for StackOverflow

Erjon Skenderi
Tampere University
erjon.skenderi@tuni.fi

Salla-Maaria Laaksonen
University of Helsinki
salla.laaksonen@helsinki.fi

Jukka Huhtamaki
Tampere University
jukka.huhtamaki@tuni.fi

Kostas Stefanidis
Tampere University
konstantinos.stefanidis@tuni.fi

Abstract

A large part of knowledge evolves outside of the operations of an organization. Question and answer online social platforms provide an important source of information to explore the underlying communities. StackOverflow (SO) is one of the most popular question and answer platforms for developers, with more than 23 million questions asked. Organizing and categorizing data is crucial to manage knowledge in such large quantities. Questions posted on SO are assigned a set of tags and textual content of each question may contain coding syntax. In this paper, we evaluate the performance of multiple text representation methods in the task of predicting tags for SO questions and empirically prove the impact of code syntax in text representations. The SO dataset was sampled and questions without code syntax were identified. Two classical text representation methods consisting of BoW and TF-IDF were selected along four other methods based on pre-trained models including Fasttext, USE, Sentence-BERT and Sentence-RoBERTa. Multi-label k'th Nearest Neighbors classifier was used to learn and predict tags based on the similarities between feature-vector representations of the input data. Our results indicate a consistent superiority of the representations generated from Sentence-RoBERTa. Overall, the classifier achieved a 17% or higher improvement on F1 score when predicting tags for questions without any code syntax in content.

Keywords: Text representation, Tag prediction, Q&A forums, StackOverflow, Knowledge-intensive work

1. Introduction

Modern organizations are increasingly becoming knowledge-intensive, which means their success is intertwined with their capacity to solve complex problems through creative and innovative solutions, information processing, and functionalist expertise (Alvesson, 2001; Feldman and March, 1981). Therefore, gathering, storing and consuming information is an essential process for knowledge-intensive organizations (Feldman and March, 1981), such as companies working in the field of information technology. Furthermore, as organizations and organizing are becoming more fluid and flexible (Schreyögg and Sydow, 2010), a large part of information and knowledge essential to them might exist outside their operational boundaries—for example, on various digital repositories of online collaboration platforms. As potential extra-organizational sites for knowledge-based value creation such platforms are of strategic importance (Laihonen and Huhtamäki, 2020).

One prominent example of such knowledge-creation sites are crowd-sourced Question and Answer (Q&A) online platforms, which present an opportunity to investigate and analyse the organisation of information as well as communities emerging around them. Such platforms may consist of millions of questions, answers and members. Managing large quantity of information, however, introduces various challenges. The most well-known Q&A professional online forum, StackExchange (SE) is a collection of more than 178 Q&A communities. Platforms of all SE communities follow a similar structure despite the fact that each has its own topic. Community members are incentivized to perform various activities such as question answering, marking duplicate questions etc., using on a reputation reward system.

One of the first and most popular community introduced on SE is StackOverflow (SO), which centers around questions related to programming. SO is designated for developers and covers multiple technology-related topics. As such, users asking questions on this platform are inclined to include coding syntax, known as code-snippets, in their questions' content. More than 23 million questions have been asked on SO since it was created. Organizing and categorizing the data is crucial to make an online platform more usable for its purpose (Alaimo and Kallinikos, 2020). As other SE communities, SO platform categorizes questions using keyword labels known as tags. Each question is assigned a set of tags that better describe the question's scope to facilitate indexing and browsing for other community members who can provide an answer.

In this work, we assess the quality of multiple text representation techniques in the task of predicting question tags. Generally, this task is known as Multi-Label Classicication (MLC) and it consists in training a classifier to learn and predict multiple labels to be assigned to a data entity. Recent approaches addressing the MLC problem achieve state-of-the-art performance in terms of accuracy (Jiang et al., 2021; J. Zhang et al., n.d.). In this study, we employ the Multi-Label k'th Nearest Neighbors (ML-kNN) (M. L. Zhang and Zhou, 2007) technique to predict tags for SO questions. ML-kNN classifier learns and predicts tags based on the similarities between the feature-vector representations of the input data. This feature of the algorithm makes it an appropriate method to assess the quality of text representation methods, despite the fact that it is outperformed by other models (Skenderi et al., 2021).

Text representation methods enable the conversion of textual information into a machine readable format which can then be used to analyse and process the extracted information by the use of algorithms. For this work, we employed two classical text representation techniques consisting of the Bag of Words and TF-IDF. In addition, we used Fasttext as a word2vec (Mikolov et al., 2013) inspired text embedding model and three other transformer-based (Vaswani et al., 2017) models consisting of: Universal Sentence Encoder (Cer et al., 2018), Sentence-BERT and Sentence-RoBERTa (Liu et al., 2019; Reimers and Gurevych, 2019).

We sampled the SO questions dataset to include 47,698 questions. The attributes of each question include title, question's body and a list of up to 5 tags. In our experiment, we extracted the information contained within the title and body of a question and used it to generate representations from the selected text representation techniques. Questions without a code-snippet in their body are identified and used separately. In our experiment, we employ a set of metrics to measure the performance of ML-kNN using using as input features obtained from each selected text representation technique.

Such experimentation is necessary in order to define best solutions to build algorithmic systems to organize data in knowledge-creation systems. Further, while text representation methods might have shown to work in one context, they might not perform as good in a other context or with different data. Thus, the contributions of this work are to 1) evaluate the performance of multiple text representation methods in the context of StackOverflow for the tag-prediction task 2) measure the impact that code snippets have over text representations, if present in a question's textual content.

The rest of this paper is organized as following. In Section 2, we describe multiple text representation methods and define the ones that are evaluated in this work. In section 3, we provide an overview of the StackOverflow dataset and specific details for the data pre-processing stage that we employed prior to evaluation. The multi-label classification task and model that we selected for our experiment are described in section 4. In section 5 we provide details on the experiment setup, evaluation metrics and disseminate the evaluation results. The experiment results and corresponding implications are discussed in section 6. Finally, in section 7 we provide the conclusions and suggest various methodological improvements to consider in future work.

2. Text Representation Methods

Text representation refers to a set of methods which enable the conversion of textual information into a machine readable format which can then be used to analyse and process textual data further with algorithms. Specifically, machine learning algorithms require machine readable input that is usually comprised of numerical values. Text representation methods can be roughly divided to classical baseline methods and more novel word embedding techniques that use pre-trained language models. In this section we explain the differences between these approaches and provide the definitions of the representation methods that were evaluated in this work.

Our selection includes the Bag of Words and Term Frequency-Inverse Document Frequency as baselines, Fasttext (Bojanowski et al., 2017), Universal Language Encoder (Cer et al., 2018), Sentence-BERT (Reimers and Gurevych, 2019) and an approach based Sentence-BERT using a refined RoBERTa model (HuggingFace, 2021; Liu et al., 2019).

2.1. Classical baselines

Bag of Words (BoW) is one of the simplest text representation techniques and could be considered a baseline. This technique consists in counting the word occurrences of a text and providing a vector of counts as the text representation. BoW ignores any potential information included in the grammar or in the order of words comprising the text. The list of counts is provided as a vector of natural numbers where each position in the vector corresponds to a distinct word. The length of a BoW representation vector equals the number of distinct words in the corpus. BoW representations do not consider the order of the words nor their frequency over all corpus.

Term Frequency - Inverse Document Frequency (TF-IDF) is a popular text representation technique which provides weighted scores for each distinct word comprising a text. This technique adjusts the frequency of each specific term in the represented document by using the total number of documents in the corpus in which the term appears. The TF-IDF score for a single word w in a text $t \in T$ is defined as:

$$Score(w,t) = tf(w,t) * log(\frac{|T|}{df(w,T)}). \quad (1)$$

Similar to BoW representations, the length of TF-IDF representations equals the number of distinct words appearing on the corpus of texts.

The subset of words appearing in a single text is usually much smaller than the size of the entire corpus. Hence, the representation vectors generated from BoW and TF-IDF techniques have a relatively small number of non-zero values and are known as sparse vector representations. BoW and TF-IDF are used on multiple works as relatively strong baselines or as textual input vectorizers for various machine learning architectures (Prabhu et al., 2018; Singh et al., 2020; Skenderi et al., 2021; You et al., 2018).

2.2. Pretrained Text Embeddings

Text representation techniques, such as BoW and TF-IDF, utilise word frequencies only and therefore do not capture the semantic meaning of the represented text, i.e. they do not account for word positions in sentences or relations between words. The representation techniques that enclose the syntactic and semantic meaning of words in a text are known as word embedding techniques. A representation generated by such techniques usually consists of a vector of real numbers $r \in R$ and it is known as a dense representation.

Specific language models are trained on usually large corpus of text data to learn patterns emerging from the co-occurrence of words (Bengio et al., 2003; Mikolov et al., 2013; Mikolov et al., 2017). Such pre-trained models can be used to generate text representations across domains, avoiding the need to train from scratch. Generally, the language models are implemented using neural-networks based architecture.

Word2vec embedding model was introduced by Mikolov et al. (2013). Their language model is well-known for it's novel architecture which consists of a shallow neural network with a single hidden layer. The authors proposed two learning approaches for Word2vec: the Continuous Bag Of Words (CBOW) in which given a set of context words, the model learns to predict a target word and Skip-gram in which the model predicts the set of context words, when the target word is provided as input. One disadvantage of the Word2vec model is its inability to generate embeddings for words which are not included in the vocabulary used during the training stage.

Fasttext model addressed the problem of providing embeddings for out-of-vocabulary words (Bojanowski et al., 2017). In their work, authors represented each word as a bag of character n-grams, including the word itself. During the inference stage, the Fasttext model sums up the corresponding n-gram representation vectors of a word and provides the result as an output. When generating the representations in the sentence or paragraph level, individual text representation vectors are normalized using the L2-norm and their average is output as an embedding. The Fasttext representation of a text consists of a vector of 300 real numbers. To overcome issues with expensive training resources needed, a list of Fasttext models, pre-trained on data from multiple languages, were presented by Mikolov et al. (2017). Due to its architecture, Fasttext provides static text representations, disregarding the text context.

Context-aware text representations can be obtained from transformer-based language models. The transformer architecture was introduced by Vaswani et al. (2017) as an approach to the machine translation problem. Nevertheless, their proposed architecture has become popular in many NLP applications since it is more efficient than previous models used for language modeling (Naseem et al., 2020). Transformers consist of an encoder-decoder structure using attention mechanisms which allow the model to focus more on certain components of text. In their work, the authors provide an exhaustive description of the transformer's architecture and implications.

Universal Sentence Encoder (USE) model is a transformer-based text representation method that was introduced by Cer et al. (2018). The USE approach consists of a sentence embedding model that learns and generates text embeddings by using the encoder component of the transformer. Authors use the attention mechanism of the transformer's encoder to calculate context-aware representations for individual words of a sentence. These representations are summed for each word position which enables the model to output a sentence representation vector with 512 dimensions. The USE representation of a text consists of a vector with 512 real numbers.

The transformer's encoder component was also used by Devlin et al. (2019) to introduce the Bidirectional Encoder Representation (BERT) model. In contrast to the previous models which processed text from the left to the right, BERT considers the context in both directions, left and right simultaneously. Originally, authors released two pre-trained model versions with varying number of parameters, namely *bert-base* and *bert-large*. Although superior on multiple Natural Language Procesing (NLP) tasks, the pre-trained models under-performed in the text embedding task (Reimers and Gurevych, 2019).

Sentence-BERT approach was presented from Reimers and Gurevych (2019) as a solution to BERT models shortcomings on providing good quality text embeddings . Sentence-BERT employs a siamese network architecture and optimizes the output embeddings so they can be compared with ordinary similarity metrics to facilitate various text classification tasks. Although authors suggest fine-tuning the proposed models when performing other tasks, in their approach they show superior results on multiple embedding evaluation benchmarks on which the models were not previously trained. The representations generated from the Sentence-BERT model for a single text consist of a vector of 1024 real numbers.

In their work, Liu et al. (2019) found that the original proposed BERT model was significantly under-trained and proposed a set of configurations for BERT's pre-training process. The authors named their technique as Robustly optimized BERT approach (RoBERTa). Their modifications improved the model's performance on multiple NLP tasks.

Sentence-RoBERTa was another model trained and evaluated from Reimers and Gurevych (2019) who found that using the original RoBERTa model instead of BERT for their approach didn't provide significant improvements in performance. A superior RoBERTa model was released as part of a project organized by HuggingFace (HuggingFace, 2021). They trained various models, including RoBERTa, with 1 billion training pairs of textual data and achieved state-of-the-art performance on multiple benchmark tasks. Similar to Sentence-BERT's, the text representation obtained from Sentence-RoBERTa consists of a vector of 1024 real numbers.

3. StackOverflow Dataset

StackOverflow (SO) is one of the first and most popular community introduced on StackExchange. As of March 2022, this Q&A site consists of a combined number of +56 Million questions and answers posted. SO is designated for programmers and it is voluntarily moderated by users based on an incentive system. Each question is manually assigned by the author at least one and up to five tags. A tag consists of keywords that represents a specific topic of the respective question. Tags are used to organize the questions and enable other expert users to browse and provide answers for them. Community members can propose and approve new tags according to a privilege reward system based on user's reputation.

Recently, the SO community is being studied from various angles such as Expert Finding (Dargahi Nobari et al., 2020), Duplicate Question Identification (Silva et al., 2018)

and Tag Suggestion (Beyer and Pinzger, 2015). In addition, multiple works have also been presented for the Tag Prediction task. For example, (Saini and Tripathi, 2018) tested various classifiers to predict tags, given the TF-IDF extracted features. Singh et al. (2020) proposed a tag association system which predicts the tags assigned to a question by employing code analysis and a tag sampling strategy. In another work, Moreno and Camargo (2020) presented a tag recommendation model intended as a tool to suggest tags for newly created questions.

Data dumps of the SO community are released quarterly by InternetArchive[1]. We imported the data dump released on March 2022 and loaded the SO community dataset in a non-relational database. SO is a particularly dynamic community with new technologies emerging constantly. Hence, to ensure more homogeneous data, we sampled our dataset to include questions posted from the beginning of 2022. Initially, the raw data consisted of 47,698 questions. We used the *title*, *body* and *tags* attributes of each question to evaluate our methods. The body attribute consisted of raw Hypertext Markup Language (HTML) and the tags were formatted as a list of keywords. We converted the body of each question to raw text and applied a series of pre-processing actions that consisted on removing urls, newlines, tabs and replacing the consecutive multiple white-spaces with a single one. When users ask a question on SO, they could provide code-snippets to illustrate the issue that they are facing. Since one of the goals of this work is to evaluate the impact of code-snippets on a question's textual content, we split

[1]https://archive.org/download/stackexchange

the dataset into two groups corresponding to questions with and without a code-snippet in their body.

Each question could be assigned up to 5 tags and from our data, we observed that most of them were given 2 or 3 tags. In Figure 1, we provide the distribution of the number of tags allocated to every question.

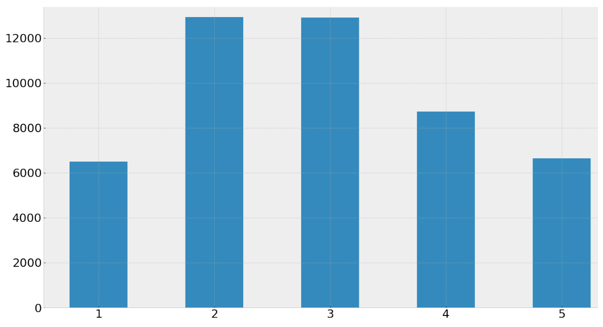

Figure 1. Distribution of the Number of Tags Assigned to Each Question.

There were 10,135 distinct tags in the dataset. The overall tag frequency distribution is illustrated in Figure 2.

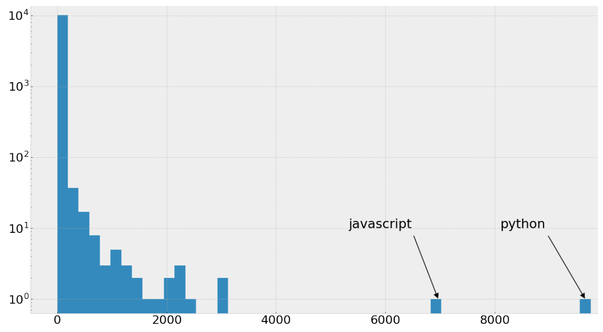

Figure 2. Overall Tag Frequency Distribution.

We observed a right skewed distribution with a long tail toward high frequencies to the right. The most popular tag is *python* which was assigned to 9,754 questions followed by *javascript*, assigned to 6,976 questions. The majority of tags only appear once or a few times. In our analysis, we used 5 fold cross validation to train a supervised learning algorithm. Given our use case, we filtered out all of the tags that appeared less than 5 times in our dataset to maximize the algorithm's learning potential. This filtering step reduced the number of distinct tags to 2024 and had a minimal effect over the number of questions in our data by reducing it by less than 0.5%. An overview of the filtered dataset, is provided in Table 1.

Table 1. Dataset Overview.

	# of Questions		# of Tags
	With Code	**Without Code**	
Raw	42,213	5,485	10,135
Filtered	41,965	5,341	2,024

4. Methodology

In this section, we provide an overview of the multi-label classification (MLC) problem, describe the classification model that we employed for this work and provide details on how it enabled us to evaluate the performance of text representation methods while predicting the tags of questions.

4.1. Multi-label Classification

MLC is the task of predicting zero or more labels to be assigned to a target data entity and has been applied in multiple domains. In this work (Loza Mencía and Fürnkranz, 2008), authors proposed EUR-Lex which is a database comprised of legal documents, each labeled with a number of labels. Multiple works in the context of MLC dataset (Chalkidis et al., 2019; Prabhu et al., 2018; You et al., 2018) have used EUR-Lex as a benchmark dataset. The authors of this work (Papanikolaou et al., 2017) evaluated an ensemble of methods to predict labels of 12 million bio-medical papers. Fiallos and Jimenes (2019) proposed a technique which consisted on training a multi-label classifier on data obtained from Reddit social network and test the model's performance at predicting the topics of interest on another dataset comprised of Twitter posts.

The solutions for the MLC problem usually employ at least one of these techniques: problem transformation and algorithm adaptation. The problem transformation approach consists in the transformation of MLC task into multiple single-label classification tasks. Usually, the predictions obtained from the single-label classifiers are merged together. Binary relevance was among the first solutions that was based on the problem transformation principle (Boutell et al., 2004). Other works such as (Hüllermeier et al., 2008) provided alternative implementations of the single-label classifiers while following the same problem transformation logic.

Other solutions for the MLC problem focus on modifying a single-label classification model to support MLC. This technique is known as algorithm adaptation. Among the first proposed approaches following this method was introduced by Schapire and Singer (1999), who modified the original algorithm of AdaBoost (Freund and Schapire, 1997) to perform MLC.

4.2. ML-kNN method

The Multi-Label k-Nearest Neighbors (ML-kNN) approach proposed from M. L. Zhang and Zhou (2007), follows the algorithm adaptation principle . This method builds on top of the k-Nearest Neighbors algorithm to provide a solution for the MLC task. On the training stage, k clusters are identified based on the euclidean distances between the feature vector representations of each data point in the training set. For our specific tag-prediction application, the tags assigned to a question q are represented through a vector defined as $Tags(q) = (1, 0, ..., 1)$, where each binary value corresponds to a distinct tag. For each tag, ML-kNN counts the number of neighbours which have selected it. The counts are used to calculate the prior and the likelihood probabilities for every single label.

During the inference stage, the ML-kNN model is provided a set of features. First, the algorithm tries to identify top k entities with the most similar set of features from the training data. The probabilities learned during the training stage are used to produce the tag-predictions by employing the maximum a posteriori (MAP) estimation. MAP assigns a probability value to each tag available, based on the prior probabilities measured during the training phase and the tag distributions of the identified k nearest neighbours.

ML-kNN's algorithm predicts the tags based on the similarity between the feature-vector representations of the training data. This feature of the algorithm makes it an appropriate method to use in our experiment. By implementing ML-kNN for the tag-prediction task we evaluated the quality and performance of the text representation methods that we selected.

The similarity scores of the representation vectors are calculated by measuring the euclidean distance. This metric performs well when comparing the text representation features of underlying texts that have similar lengths. In our case, the textual content of questions is not identical. We addressed this issue by performing L2 normalization over each representation vector. This technique scales the values comprising a vector such that the sum of their squares equals 1. The euclidean distance computed from the normalized vectors is proportional to the cosine distance which performs well in various text mining tasks (M.K and K, 2016).

5. Experimental Evaluation

In this section, we describe the evaluation of the ML-kNN model using the selection of text representation methods. In Table 2 we provide details on the properties of each method. We used three evaluation metrics to provide insights on the performance ML-kNN combined with each representation method. A description of the experiment setup is provided to facilitate reproducing of this work. Next, we provide details on the evaluation process and disseminate the results.

Table 2. The properties of the selected text representation methods.

Representation	Sparsity	Dimensions	Context Aware
Bag of Words	Sparse	41,579	No
TF-IDF	Sparse	41,579	No
Fasttext	Dense	300	No
USE	Dense	512	Yes
Sent-BERT	Dense	1024	Yes
RoBERTa	Dense	1024	Yes

5.1. Evaluation Metrics

The ML-kNN model provides the predicted tags of a question as a multi-hot numbered vector comprised of 1's (select) and 0's (do not select) where each position corresponds to a distinct tag. The order of the predicted tags was not of concern to our evaluation. Ideally, the tag predictions for a question in our dataset should include all of the actually assigned tags and avoid selecting the non-assigned ones. Evaluating the performance of our classifier by using the simple accuracy metric would not be feasible given the class imbalances. We selected the Precision, Recall and F1 scores to evaluate the performance of the multi-label predictions, based on related work (M. L. Zhang and Zhou, 2014). In Table 3, we provide the binary confusion matrix to illustrate the definitions of the selected metrics where *positive* and *negative* terms correspond to a tag being *selected* and *not selected*, respectively.

Table 3. Binary Confusion Matrix.

		Ground Truth	
		Positive(1)	Negative(0)
Predicted	Postive(1)	TP	FP
	Negative(0)	FN	TN

Given two vectors, one representing the actual tags and the other containing the predicted ones, the totals for true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN) are calculated.

The precision score measures the proportion of positive predicted tags that are actually positive over all tags predicted as positive and it is defined as:

$$Precision = \frac{TP}{TP + FP}. \tag{2}$$

With Recall we compute the proportion of positive predicted tags that are actually positive over all tags that are indeed positive. The Recall score is defined as:

$$Recall = \frac{TP}{TP + FN}. \tag{3}$$

Our selected model can achieve a high precision by only predicting positive tags on which it has the highest confidence i.e. with highest probability.

The F1 score is used as a balancing measure between the precision and recall scores, and is defined as follows:

$$F1\_score = 2 * \frac{Precision * Recall}{Precision + Recall}. \tag{4}$$

There are two main methods to aggregate precision, recall and F1 scores: macro and micro-averaging (M. L. Zhang and Zhou, 2014). With the macro-averaging, the scores are calculated independently for each tag and the average overall tags is computed. In micro-averaging technique the true positives, false positives and false negatives for each tag are summed up and the resulting values are used to calculate the precision, recall and F1 scores. We choose the micro-averaging method since it provides better overview for datasets with imbalanced labels, such as ours (M. L. Zhang and Zhou, 2014).

5.2. Experiment Setup

The experiment was run on a single computing instance[2] to keep the evaluation reproducible and congruous across the multiple combinations. We choose the *scikit-multilearn* python library's implementation of the ML-kNN model (Szymański and Kajdanowicz, 2017). We used the *tensorhub*[3] library to generate the text representations for USE and *SentenceTransformers*[4] to generate representations from Sentence-BERT and Sentence-RoBERTa pre-trained models.

ML-kNN model is initialized with two parameters: number of neighbors *k* and smoothing parameter *s*. We performed a grid-search parameter optimization with 3 folds of cross validation and found k=3 and s=1.0 to be the configurations yielding the best performances in our dataset. Given the imbalanced distribution of tags in our dataset, we performed a 5 fold cross validation evaluation of ML-kNN for each of our selected text representation methods. The results were computed separately for questions with and without a code snippet in their textual content.

[2]CPU: Intel Xeon Processor (Skylake, IBRS) 10 cores ,RAM: 42.1GB

[3]https://www.tensorflow.org/hub

[4]https://www.sbert.net/

5.3. Evaluation Results

To assess the performance of each combination of ML-kNN model with the selected text representation methods, we provide the evaluation results in Table 4. The evaluation results are assessed for questions with and without code-snippet(s) in their textual content. Each score is computed by performing micro-average aggregation of the corresponding scores obtained from evaluating the model on 5 cross-validation folds.

Table 4. Evaluation results of the text representation methods for questions with and without a code snippet in their textual content.

	Precision	**Recall**	**F1 score**
Questions with code snippets			
BoW	0.25	0.54	0.34
TF-IDF	0.28	0.59	0.38
Fasttext	0.19	0.47	0.27
USE	0.39	0.65	0.49
Sentence-BERT	0.17	0.51	0.25
Sentence-RoBERTa	**0.48**	**0.70**	**0.57**
Questions without code snippets			
BoW	0.39	0.76	0.51
TF-IDF	0.40	0.78	0.53
Fasttext	0.32	0.73	0.44
USE	0.50	0.80	0.61
Sentence-BERT	0.27	0.75	0.39
Sentence-RoBERTa	**0.58**	**0.82**	**0.68**

6. Discussion

Based on the computed F1 scores during the evaluation, the overall performance of ML-kNN classifier was similar to other approaches using datasets with similar properties (Singh et al., 2020; Skenderi et al., 2021).

From our results, we observe that the recall score is consistently higher than precision among all of the evaluated combinations. Hence, ML-kNN predicts most of the correct tags, regardless of which representation method was used. The classification model achieved highest scores when using the Sentence-RoBERTa representations. Presumably, the excessive pre-training of the RoBERTa model on data from multiple domains (HuggingFace, 2021) improved the quality of its representations. The runner-up text representation method according to our results is the Universal Sentence Encoder (USE) model. In terms of precision and recall scores, the performance of USE representations was up to 0.08 points off the Sentence-RoBERTa representation's performance. Overall, both BoW and TF-IDF classical text representation methods consistently outperformed the Fasttext and Sentence-BERT generated representations.

All of our selected text representation methods perform better when ML-kNN predicts tags of questions without code-snippets in the textual content. On questions without code-snippets the respective evaluation scores increase by more than 17%. Even though the syntax of most high-level programming and scripting languages may mimic specific natural language words and structure, our results indicate that they introduce an additional noise to text data.

7. Conclusions and Future Work

Managing valuable information generated on social platforms is important for knowledge workers and knowledge-intensive organizations. Text representation methods enable the conversion of textual information into a machine readable format which can then be used to analyse and process text data further by using algorithms. This work presented a quantitative evaluation study of text representation methods in the multi-label classification task. We employed the Question and Answer StackOverflow (SO) community data to run our experimental evaluations. SO community maintains a list of available tags that are used to label and categorize each question. The potential presence of code-snippets in a questionis one specific property of the StackOverflow questions. Our filtered dataset comprised of 41,965 questions with code-snippets and 5,341 questions without a code-snippets in their body.

The Multi-label k'th Nearest Neighbors (ML-kNN) method was used to learn and predict tags for each question. This specific multi-label classifier considers the euclidean distance of the feature representations of the input data when learning and inferring labels. We selected the Bag of Words, Term Frequency-Inverse Document Frequency, Fasttext (Bojanowski et al., 2017), Universal Language Encoder (Cer et al., 2018), Sentence-BERT (Reimers and Gurevych, 2019) and Sentence-RoBERTa model (HuggingFace, 2021; Liu et al., 2019) as text representation techniques. The ML-kNN classifier was evaluated in the tag prediction task for each representation type by employing a 5-fold cross validation scheme. We used precision, recall and F1-scores to assess the predictive performance of the classifier for each run. Our findings show a superiority of the representations generated from Sentence-RoBERTa in terms of predictive performance. The classifier achieved +17% improvement of the predictive performance when predicting the tags of questions without code-snippets, for all of the representation techniques.

In future work, we will consider running the experiment with the full StackOverflow dataset. In addition, models such as Code-BERT (Feng et al., 2020) and BERTOverflow (Tabassum et al., 2020), that were specifically trained on code textual data, may be used to generate the text representations. Using such models may convert the noise introduced from code-snippets into additional information that may improve the quality of text representations in various tasks. Fine-tuning the pre-trained models to learn textual representations specifically for SO dataset is another aspect which may enable the ML-kNN classifier to achieve a higher performance.

References

Alaimo, C., & Kallinikos, J. (2020). Managing by Data: Algorithmic Categories and Organizing: *https://doi.org/10.1177/0170840620934062*, *42*(9), 1385–1407. https://doi.org/10.1177/0170840620934062

Alvesson, M. (2001). Knowledge work: Ambiguity, image and identity. *Human relations*, *54*(7), 863–886.

Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, *3*(6), 1137–1155. https://doi.org/10.1162/153244303322533223

Beyer, S., & Pinzger, M. (2015). Synonym Suggestion for Tags on Stack Overflow. *IEEE International Conference on Program Comprehension*, *2015-Augus*, 94–103. https://doi.org/10.1109/ICPC.2015.18

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, *5*, 135–146. https://doi.org/10.1162/tacl{_}a{_}00051

Boutell, M. R., Luo, J., Shen, X., & Brown, C. M. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757–1771. https://doi.org/10.1016/J.PATCOG.2004.03.009

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St John, R., Constant, N., Guajardo-Céspedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., & Kurzweil Google Research Mountain View, R. (2018). Universal Sentence Encoder. *AAAI*, 16026–16028. https://doi.org/10.48550/arxiv.1803.11175

Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N., & Androutsopoulos, I. (2019). Extreme Multi-Label Legal Text Classification: A case study in EU Legislation. http://arxiv.org/abs/1905.10892

Dargahi Nobari, A., Neshati, M., & Sotudeh Gharebagh, S. (2020). Quality-aware skill translation models for expert finding on StackOverflow. *Information Systems*, *87*, 101413. https://doi.org/10.1016/J.IS.2019.07.003

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*, 4171–4186. http://arxiv.org/abs/1810.04805

Feldman, M. S., & March, J. G. (1981). Information in organizations as signal and symbol. *Administrative science quarterly*, 171–186.

Feng, Z., Guo, D., Tang, D., Duan, N., Feng, X., Gong, M., Shou, L., Qin, B., Liu, T., Jiang, D., & Zhou, M. (2020). CodeBERT: A pre-trained model for programming and natural languages. *Findings of the Association for Computational Linguistics Findings of ACL: EMNLP 2020*, 1536–1547. https://doi.org/10.18653/V1/2020.FINDINGS-EMNLP.139

Fiallos, A., & Jimenes, K. (2019). Using reddit data for multi-label text classification of twitter users interests. *2019 6th International Conference on eDemocracy and eGovernment, ICEDEG 2019*, 324–327. https://doi.org/10.1109/ICEDEG.2019.8734365

Freund, Y., & Schapire, R. E. (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. https://doi.org/10.1006/JCSS.1997.1504

HuggingFace. (2021). *Train a Sentence Embedding Model with 1B Training Pairs* (tech. rep.). https://huggingface.co/blog/1b-sentence-embeddings

Hüllermeier, E., Fürnkranz, J., Cheng, W., & Brinker, K. (2008). Label ranking by learning pairwise preferences. *Artificial Intelligence*, *172*(16-17), 1897–1916. https://doi.org/10.1016/J.ARTINT.2008.08.002

Jiang, T., Wang, D., Sun, L., Yang, H., Zhao, Z., & Zhuang, F. (2021). LightXML: Transformer with Dynamic Negative Sampling for High-Performance Extreme Multi-label Text Classification. *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, *9B*, 7987–7994. https://doi.org/10.48550/arxiv.2101.03305

Laihonen, H., & Huhtamäki, J. (2020). Organisational hybridity and fluidity: deriving new strategies for dynamic knowledge management. *Knowledge Management Research and Practice*. https://doi.org/10.1080/14778238.2020.1794993

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., & Allen, P. G. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. https://doi.org/10.48550/arxiv.1907.11692

Loza Mencía, E., & Fürnkranz, J. (2008). Efficient pairwise multilabel classification for large-scale problems in the legal domain. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5212*(PART 2), 50–65. https://doi.org/10.1007/978-3-540-87481-2{_}4

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2017). Advances in Pre-Training Distributed Word Representations. *LREC 2018 - 11th International Conference on Language Resources and Evaluation*, 52–55. http://arxiv.org/abs/1712.09405

M.K, V., & K, K. (2016). A Survey on Similarity Measures in Text Mining. *Machine Learning and Applications: An International Journal*, *3*(1), 19–28. https://doi.org/10.5121/MLAIJ.2016.3103

Moreno, F. A., & Camargo, J. E. (2020). A Content-Based Multi Label Classification Model to Suggest Tags for Posts in Stack

Overflow. *Communications in Computer and Information Science*, *1154 CCIS*, 176–187. https://doi.org/10.1007/978-3-030-46785-2{_}15

Naseem, U., Razzak, I., Khan, S. K., & Prasad, M. (2020). A Comprehensive Survey on Word Representation Models: From Classical to State-Of-The-Art Word Representation Language Models. *ACM Transactions on Asian and Low-Resource Language Information Processing*, *20*(5). https://doi.org/10.48550/arxiv.2010.15036

Papanikolaou, Y., Tsoumakas, G., Laliotis, M., Markantonatos, N., & Vlahavas, I. (2017). Large-scale online semantic indexing of biomedical articles via an ensemble of multi-label classification models. *Journal of Biomedical Semantics*, *8*(1). https://doi.org/10.1186/S13326-017-0150-0

Prabhu, Y., Kag, A., Harsola, S., Agrawal, R., & Varma, M. (2018). Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*, 993–1002. https://doi.org/10.1145/3178876.3185998

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, 3982–3992. https://doi.org/10.48550/arxiv.1908.10084

Saini, T., & Tripathi, S. (2018). Predicting tags for stack overflow questions using different classifiers. *Proceedings of the 4th IEEE International Conference on Recent Advances in Information Technology, RAIT 2018*, 1–5. https://doi.org/10.1109/RAIT.2018.8389059

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336. https://doi.org/10.1023/A:1007614523901

Schreyögg, G., & Sydow, J. (2010). Crossroads—organizing for fluidity? dilemmas of new organizational forms. *Organization science*, *21*(6), 1251–1262.

Silva, R. F., Paixão, K., & De Almeida Maia, M. (2018). Duplicate question detection in stack overflow: A reproducibility study. *25th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2018 - Proceedings*, *2018-March*, 572–581. https://doi.org/10.1109/SANER.2018.8330262

Singh, P., Chopra, R., Sharma, O., & Singla, R. (2020). Stackoverflow tag prediction using tag associations and code analysis. *https://doi.org/10.1080/09720529.2020.1721857*, *23*(1), 35–43. https://doi.org/10.1080/09720529.2020.1721857

Skenderi, E., Huhtamäki, J., & Stefanidis, K. (2021). Multi-Keyword Classification: A Case Study in Finnish Social Sciences Data Archive. *Inf.*, *12*(12), 491. https://doi.org/10.3390/info12120491

Szymański, P., & Kajdanowicz, T. (2017). A scikit-based Python environment for performing multi-label classification. *Journal of Machine Learning Research*, *1*, 1–15. https://doi.org/10.48550/arxiv.1702.01460

Tabassum, J., Maddela, M., Xu, W., & Ritter, A. (2020). Code and Named Entity Recognition in StackOverflow, 4913–4926. https://doi.org/10.18653/V1/2020.ACL-MAIN.443

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*, 5999–6009. https://arxiv.org/abs/1706.03762v5

You, R., Zhang, Z., Wang, Z., Dai, S., Mamitsuka, H., & Zhu, S. (2018). AttentionXML: Label Tree-based Attention-Aware Deep Model for High-Performance Extreme Multi-Label Text Classification. *arXiv*. http://arxiv.org/abs/1811.01727

Zhang, J., Chang, W., . . ., H. Y. .-. A. i. N., & 2021, u. (n.d.). Fast multi-resolution transformer fine-tuning for extreme multi-label text classification. *proceedings.neurips.cc*. https://proceedings.neurips.cc/paper/2021/hash/3bbca1d243b01b47c2bf42b29a8b265c-Abstract.html

Zhang, M. L., & Zhou, Z. H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, *40*(7), 2038–2048. https://doi.org/10.1016/j.patcog.2006.12.019

Zhang, M. L., & Zhou, Z. H. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837. https://doi.org/10.1109/TKDE.2013.39