

2020

Comparing tagging suggestion models on discrete corpora

Bojan Bozic

Technological University Dublin, bojan.bozic@tudublin.ie

Andre Rios

Technological University Dublin, andre.rios@tudublin.ie

Sarah Jane Delany

Technological University Dublin, sarahjane.delany@tudublin.ie

Follow this and additional works at: <https://arrow.tudublin.ie/ittsciart>

 Part of the [Databases and Information Systems Commons](#)

Recommended Citation

Bozic, Bojan; Rios, Andre; and Delany, Sarah Jane, "Comparing tagging suggestion models on discrete corpora" (2020). *Articles*. 88.

<https://arrow.tudublin.ie/ittsciart/88>

This Article is brought to you for free and open access by the School of Science and Computing at ARROW@TU Dublin. It has been accepted for inclusion in Articles by an authorized administrator of ARROW@TU Dublin. For more information, please contact arrow.admin@tudublin.ie, aisling.coyne@tudublin.ie.



This work is licensed under a [Creative Commons Attribution-NonCommercial-Share Alike 4.0 License](#)

Comparing tagging suggestion models on discrete corpora

Bojan Bozic, Andre Rios and Sarah Jane Delany
*Department of Computer Science, Technological University Dublin,
Dublin, Ireland*

Tagging
suggestion
models

201

Received 3 August 2019
Revised 6 August 2019
Accepted 6 August 2019

Abstract

Purpose – This paper aims to investigate the methods for the prediction of tags on a textual corpus that describes diverse data sets based on short messages; as an example, the authors demonstrate the usage of methods based on hotel staff inputs in a ticketing system as well as the publicly available StackOverflow corpus. The aim is to improve the tagging process and find the most suitable method for suggesting tags for a new text entry.

Design/methodology/approach – The paper consists of two parts: exploration of existing sample data, which includes statistical analysis and visualisation of the data to provide an overview, and evaluation of tag prediction approaches. The authors have included different approaches from different research fields to cover a broad spectrum of possible solutions. As a result, the authors have tested a machine learning model for multi-label classification (using gradient boosting), a statistical approach (using frequency heuristics) and three similarity-based classification approaches (nearest centroid, k-nearest neighbours (k-NN) and naive Bayes). The experiment that compares the approaches uses recall to measure the quality of results. Finally, the authors provide a recommendation of the modelling approach that produces the best accuracy in terms of tag prediction on the sample data.

Findings – The authors have calculated the performance of each method against the test data set by measuring recall. The authors show recall for each method with different features (except for frequency heuristics, which does not provide the option to add additional features) for the dmbook pro and StackOverflow data sets. k-NN clearly provides the best recall. As k-NN turned out to provide the best results, the authors have performed further experiments with values of k from 1–10. This helped us to observe the impact of the number of neighbours used on the performance and to identify the best value for k .

Originality/value – The value and originality of the paper are given by extensive experiments with several methods from different domains. The authors have used probabilistic methods, such as naive Bayes, statistical methods, such as frequency heuristics, and similarity approaches, such as k-NN. Furthermore, the authors have produced results on an industrial-scale data set that has been provided by a company and used directly in their project, as well as a community-based data set with a large amount of data and dimensionality. The study results can be used to select a model based on diverse corpora for a specific use case, taking into account advantages and disadvantages when applying the model to your data.

Keywords Tag prediction, Natural language processing, Multi-label classification, k-nearest neighbor, Naive Bayes

Paper type Research paper

This paper is an extended version of the following conference paper: Bojan Božić, André Rios and Sarah Jane Delany (2018), Validation of Tagging Suggestion Models for a Hotel Ticketing Corpus. In Proceedings of the 20th International Conference on Information Integration and Web-based Applications and Services (iiWAS2018), Maria Indrawan-Santiago, Eric Pardede, Ivan Luiz Salvadori, Matthias Steinbauer, Ismail Khalil and Gabriele Anderst-Kotsis (Eds.). ACM, New York, NY, USA, 15–23. DOI: <https://doi.org/10.1145/3282373.3282386>. Furthermore, the authors send their thanks to DmBook Pro [9] who provided a manually tagged data set, which our research was based on.



1. Introduction

In this paper, we are analysing two separate data sets, one that represents short text entries from hotel staff in a ticketing system [1] [2] and the other one is a data set from the well-known StackOverflow [3] platform. Firstly, we will be looking into the dmbook pro data set, which has 2,017 entries that have been tagged by a representative user manually and include a the name of the book (which represents the category of a ticket and equals a hotel book for separate types of entries, e.g. rooms, maintenance, logs, etc.), the type of the book (is a short version of the book name), the actual content of the ticket, an overall ID of the ticket, the respective tags that the representative user regards as correct and complete and finally, the user ID of the user he created the ticket. Secondly, we are analysing our models with the StackOverflow data set as well. This data set has 263,540 entries which are tagged and include the title of the entry or question, a unique question identifier for every question, the user ID of the user who created the question, a score from other users representing the impact or quality of the question, the time at which the question has been posted (measured in epoch time), the respective tags that represent correct tags for our training set and could be e.g. “python”, “R”, “jnlp”, “compatibility”, etc. (in most cases, a number of tags between two and six have been used), number of views for the posted question, number of answers to the question, a unique identifier for each answer, an identifier for every user who answered, score for each of the answers and finally, the time when the answer has been posted. The main concern is how to predict as many correct tags as possible based on the several feature variables, with text content being the main one. Our approach was to apply server models, compare the results and find the best fit for our problem.

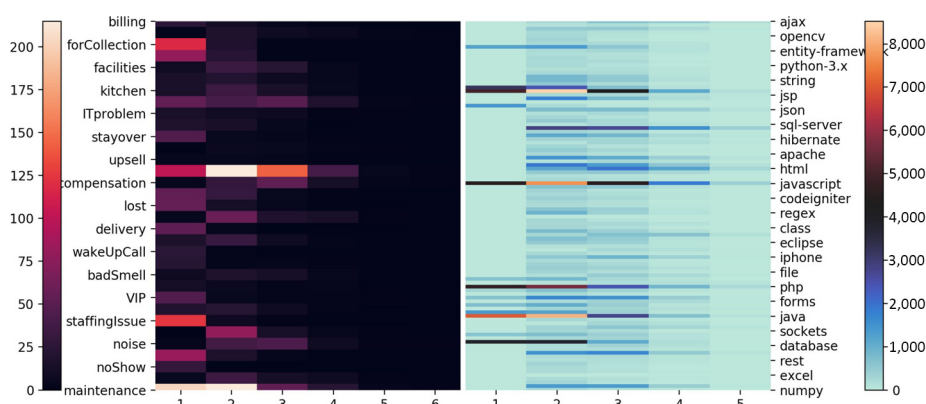
However, our work is limited by the amount of data, especially the dmbook pro data set which only has 2,017 observations; therefore, we have tried to make up for it by including the StackOverflow data set, which, having 263,540 observations, is significantly bigger. The size of the sample has a direct impact on the difficulty of prediction, and hence, our goal is to increase the amount of available training data even more to improve our results. Considering the underlying data, however, one should keep in mind that especially tags with lower frequency are harder to predict.

Our main objective is to improve recall in special and the whole process of tagging short text entries in specific for all types of semi- or fully automated tagging systems. The baseline of our research are users who have to tag their text entries or tickets manually with pre-defined hash tags, which leads to low consistency as some users may require more tags for a specific entry than others, because there might be users who are more prone to tagging, while others might not be willing to put in the effort and do not use tags at all. Our suggestion for a future system is, hence, either a fully automatic tagging with a feature for manual correction or a system that suggests most likely tags and the user has to select the right ones before proceeding, which would be a semi-automated approach.

In this paper, we will be first exploring our two data sets by the means of statistical analysis of independent variables, as well as visualisation with graphing techniques to gain understanding of the underlying data sets. In the second part of the paper, we will evaluate several approaches and methods of predicting tags to solve the earlier introduced problem. The main contribution is a recommendation of one specific model and configuration (selection of parameters and variables) that provides the best results for predicting tags on the evaluated data.

In Figure 1, we can see how tags are distributed in the dmbook pro and StackOverflow data sets through heatmaps. On the left and right of the figure, tags that have been used are shown (e.g. ITproblem, TvNotWorking, VIP, etc. for dmbook pro, and ajax, opencv, entity framework, python-3.x, etc. for StackOverflow). The bottom of the figure shows how many

Figure 1.
Overview of tags and
frequencies for both
data sets in
a heatmap



tags have been used for the respective ticket or question (tickets are ranging from one to six and questions from one to five). Every single field in the map shows how many times a tag has been used in combination with other tags (from one alone to six with five others). For example, the tag “maintenance” has been used 498 times in total, but even 204 times just standalone, while it has been used 215 times together with exactly one other tag, etc. In comparison, the tag “electrical” is always used in combination with others. On the right side, we can easily see the most frequently used tags in StackOverflow, such as “java”, “javascript” and “string”.

The structure of the paper includes the following sections: In Section 2, we summarise the state of the art in the field of tag prediction by referencing and exploring related work. Section 3 explains our approach in detail, especially focusing on the research method and the measures we used to define the performance. Exploration of data is presented in Section 4, and the functionality of investigated methods and approaches in Section 5. The last two sections, Section 7 and Section 8, show results as well as our conclusions and future work.

2. Related work

In this work, we define tags are short phrases or words written as hashtags and concatenated in camel case, which annotate a short free text entry (in our data sets, these entries are either tickets or questions). These tags are used to represent the free text and can be seen as a form of classification or even summarisation of the content. While exploring the state of the art, we have split relevant work in subsections depending on the method of tag detection and prediction used. Therefore, we are providing an overview of statistical, similarity, probabilistic, semantic models and recommender systems.

Since the dawn of Web 2.0, tagging has been gaining traction in terms of popularity and usefulness, especially in social media and general social platforms where images and short-text posts are shared, linked and reacted upon (Smith, 2007). However, tagging can be used in many different tasks of information management. From a software perspective, the biggest challenge is how to identify and assign tags in an automatic way so that textual corpora can be archived/stored, retrieved and searched for most efficiently and by adding semantic enrichment to keep the context.

2.1 *Statistic models*

The simplest approaches used for text analysis and tagging involve traditional statistical methods. An overview of statistical text analysis can be found in [Miner et al. \(2012\)](#). Most popular have been linear models, which are very good at finding the context of a text based on keywords ([Kutner et al., 2005](#)). Lastly, [Briscoe and Carroll \(2002\)](#) describe a robust and accurate statistical model for annotating text corpora, which can be applied as an automatic tagging system.

2.2 *Similarity models*

Similarity models span from collaborative tagging where every user is able to tag textual content ([Golder and Huberman, 2005](#)) to polysemy where words can have different meanings, and therefore specificity is an important factor ([Noruzi, 2006](#)). When tagging text corpora collaboratively, the concept that is used is called a folksonomy ([Trant, 2009](#)). The similarity approaches used in this paper and represent this area of research are multi-label classification ([Tsoumakas and Katakis, 2007](#)), nearest centroid (as described in [Cazzanti et al., 2008](#)) and k -nearest neighbours ([Pascal and Mineau, 2001](#)).

2.3 *Probabilistic models*

Most probabilistic models are used in machine learning techniques or generate models that can be used to classify data sets ([Krestel and Fankhauser, 2009](#); [Allahyari and Kochut, 2016](#)). Therefore, they are a logical choice for detection and prediction of tags in free text. An interesting topic model that uses a probabilistic approach on DBpedia [4] knowledge is [Allahyari and Kochut \(2016\)](#). Already tagged data can be processed further by the use of tag clustering ([Begelman et al., 2006](#); [Shepitsen et al., 2008](#)), graph partitioning algorithms ([Song et al., 2011](#)) or supervised learning algorithms ([Stanley and Byrne, 2013](#)). The probabilistic model used in our work and representing this category of models is naïve Bayes ([Rish, 2001](#)).

2.4 *Semantic models*

Capturing the context of free text has been a major obstacle in tag prediction; however, the Semantic Web community has produced good results in using ontologies for this purpose ([Wang et al., 2004](#)). Such an example is the work published by [Djuana et al. \(2011\)](#) where user tagging is used to learn a domain ontology and WordNet builds the basis on which this ontology relies upon. Similarly, [Lin et al. \(2009\)](#) use low support association rule mining to build ontological structures. Alternatively, collaborative ontologies can be used in a user-driven way for recommendation of tags ([Prokofyev et al., 2012](#)) or a topic ontology can be constructed by using an existing set of categories and enriching it by the use of external resources for knowledge, e.g. Wikipedia [5] or WordNet [6].

2.5 *Recommender systems*

Even though recommender systems are not part of our research scope, we are listing some relevant work in the area that could also be used for tag prediction and similar tasks in relation to tags. One such approach to improve the process of assigning user-created tags to text input with the help of a recommender system in the area of social web is [Fabiano et al. \(2017\)](#) and [Campana and Delmastro, 2017](#)). Also natural language processing (NLP) has been used in combination with recommender systems to extract tags, identify keywords and use them for tags ([Kazi and Ng., 2014](#)). This is also a regularly used approach for capturing the context, such as metadata or ratings, which can be associated with tag prediction and used to improve the process. As recommender systems use structured tagging and therefore

provide tags with metadata or context to improve results, there is research in this area as well (Case study, 2006). Finally, extracting tags from information found in the profiles of users can help to solve the problem of how to kick off the tagging process, which is called overcoming the problem of a “cold start”, mentioned in the work of Preisach *et al.* (2010).

3. Approach and research method

Our approach consists of two phases. Phase 1 includes the identification of methods that are appropriate for predicting tags and represent an area of research (such as statistics, similarity and probability) and comparing them to each other based on recall as metric of performance. The second phase includes evaluation of the models and selection of the best fit as the recommended model. Therefore, we have named the two phases exploration and evaluation. Before commencing our work, the data has undergone preprocessing and cleansing to make it ready for usage as an input to the models. In detail, Bag of Words has been used to represent the text as well as tokenising to remove white spaces and punctuation and extract only words of interest. Furthermore, stopwords have been removed with the same purpose, and all remaining words were lemmatised to make sure that no redundant information was involved.

3.1 Exploration of sample data

The first step was to statistically describe and explore the data set and identify any signal present as well as derive facts from the underlying data. In the dmbook pro data set, we have analysed the features textual content, books and tags by creating statistics and plotting graphs for first visual data exploration, and in further steps, aggregation of variables of interest and basic descriptive statistical analysis. The features of interest from the StackOverflow data set were the titles of questions, scores for questions, times when the questions were posted, number of views, number of answers and tags as dependent variable.

3.2 Evaluation of tag prediction approaches

Step 2 of our approach and research method was the evaluation of tag prediction approaches in which identification of several algorithms for tag prediction, based on research papers, has been performed, and these algorithms became candidates for the best fit approach. The list of candidates contains five models and algorithms that could potentially reach the best results in relation to the input variables and desired output. The selected models and algorithms are a statistical analysis model, which uses frequency heuristics to find the right tag dynamically, a multi label-classification model that is a supervised machine learning model and uses gradient boosting, a model that uses probability to determine the most probable tags using the naïve Bayes algorithm, and two classification models that use similarity to find the closest tags, which are the nearest centroid algorithm and the k -nearest neighbours algorithm. A more detailed overview of the models and algorithms used can be found in Section 5. Finally, we have used recall to measure the performance of all selected models and algorithms and decide which is the best fit for our task.

3.2.1 Method. We have used ten-fold cross validation to evaluate candidate methods and algorithms. k -fold cross-validation (where k stands for the number of times the data set is split into test set and training set) is a *de facto* standard in supervised machine learning. In our case, the data set has been split into ten parts, and we have used each of them as the test set when the rest have been used to train the model. The performance metric (recall) is the average of those ten runs and is calculated by counting how many of the predicted tags are also in the set of actual tags as defined by the user.

3.2.2 *Test approach.* To evaluate the models and algorithms, we were using ten-fold cross validation to split data into training and test sets and took the average recall results of the ten runs in the result table. Therefore, the approaches that require training have been trained on the training set and tested with the test set by comparing the identified tags to the user-defined ones and calculating recall according to the definition below. It was made sure that different data sets were being used for training and test to avoid learning data from same examples as were used for testing.

3.2.3 *Recall.* This is a definition of the performance measure used which is recall. Recall, as defined in the performed experiments, is the ratio of correctly found and suggested tags over the amount of actually correct tags as defined manually in the test set. As a formula, this is $\frac{ST}{OT}$ where *ST* is the set of correctly suggested tags and *OT* is the set of originally defined correct tags in the test set. For example, a text entry that was tagged with [“plumbing”, “maintenance”, “noise”] could score a set of suggested tags of [“plumbing”, “cleanliness”, “maintenance”] by the system, in which case the recall would be $\frac{2}{3}$, as of the three tags from the original data set, two have been guessed correctly, the position of guessed tags is hereby not relevant. The reason recall was chosen is that it fits the problem of finding as many relevant tags as possible, whereby it is not a problem if the system suggests a few tags that might not be relevant as long as the number of correct tags is maximal. This is a good fit, as we expect a software solution that leaves the final decision to the user, and therefore, the user will be able to eliminate all irrelevant tags manually.

4. Data exploration

Firstly, we were looking into how the two selected data sets were structured and which characteristics they had. After that, we have extracted features and defined parameters for our tag prediction experiments. Lastly, we have tested all selected models with both data sets.

4.1 The dmbook pro data set

A snippet of dmbook pro data is shown in Table 1. This data set is structured as follows:

- *book_name:* This is the name of the so-called hotel book that also describes the category of the text entry. A hotel book is the place where physical tickets are created by hotel staff to log specific events at reception, regarding room service, maintenance, general logs, etc. Smaller hotels would usually maintain only one book. As there is no standardised naming of books, this variable might be different

	book_name	book_type	content	id	tags	user_id
0	DM Log	log	Erna from Pandora called in sick for this ev. . .	99324	[staffingIssue]	998
1	DM Log	log	928 and 481 are VIP's	92996	[VIP]	998
2	DM Log	log	had to move Mr Peters room 935 to 838 he is wit. . .	97855	[roomMove]	998
3	Nights	log	Guest in room 571 no wake up call 2 nd time in. . .	97507	[guestIncident, wakeUpCall]	998
4	Maintenance	log	!Urgent! - Please replace outside balcony door. . .	97411	[maintenance]	998
5	10.30am Briefing Notes	log	Welcome Inspection tomorrow, all staff in full. . .	97284	[internal]	998

Table 1.
A snippet of data from the dmbook pro data set

throughout hotels, even though the book might be used for the same category, e.g. one hotel might use “reception” and another front desk for the same book type.

- *book_type*: A short name for the book that is sometimes used to define its type, so different books might have the same type in a hotel (at least theoretically).
- *content*: The content of the ticket text a staff member might create when an event occurs in the hotel to log it. This is the main observation that we want to tag, and therefore, we are using it as input to predict tags.
- *id*: Unique identifier for entries to the ticketing system.
- *tags*: Our dependent variable, as this is what we want to predict. No fixed number of tags have been defined; therefore, it could be any amount of tags we are looking for. However, all tickets in the dmbook pro data set have a maximum of six tags.
- *user_id*: The user identifier for the user who generated the entry. There might be a correlation of users and book names, as a member of staff normally would work in one part of the hotel and therefore only use one or a few different books. However, in small hotels, there might be only one book, or one member of staff might have multiple functions and therefore use more books.

According to the statistical analysis, we have used three variables for prediction of tags: the book name as category of tickets, the book type for additional information in case several books are related to each other and the content of the ticket that provides the most significant information regarding potentially relevant tags. Our dependent variable and the variable we want to predict is tags.

Figure 2 shows an overview of all tags and their frequencies used in the 2,017 entries of our data set. How often each of the 33 tags occurs also tells us how important they are and how likely it is that the tag is included in the list of tags for a new entry. The idea is to enable new tags in the collection as well; however, some training will be required until a new tag becomes available for automatic tagging. Furthermore, we expect the system to stay within the limits of 40–50 tags in the long run, as a hotel ticketing system is very specific and will only require a relatively small finite amount of tags in total.

As we can see in Figure 2, two of the used tags are fairly predominant, namely, *guestIncident* and *maintenance*. Together, they make up 33 per cent of the total number of tags. It also indicates that we might be looking at different levels of tags, meaning that while some of them are rather general and others could be very specific and therefore used less frequently. An example for such a specific tag is *TvNotWorking*. It is used in only 0.46 per cent of the tickets in our data set and describes a very specific issue. Should a data set with a much larger amount of tags be used, we would suggest creating a taxonomy for tags to distinguish between general and specific ones. This would also enable us to assign tags on different levels depending on the use case or role of the staff member looking at the data.

Figure 3 shows the amount of tags used for each entry in the data set. We can see clearly that the majority of entries only use one tag and the vast minority three or more. We can infer from the graph that by using only three entries, we can cover the biggest part of the data set.

During meetings and discussions with the customer, we have determined that it is not a drawback to list additional tags as recommendations that might be irrelevant. Indeed, they might even be useful, as some of them might be relevant but not considered yet for a specific entry. However, this paper does not cover generating new tags or extracting additional information from the textual content. If we want to include new tags, we would have to

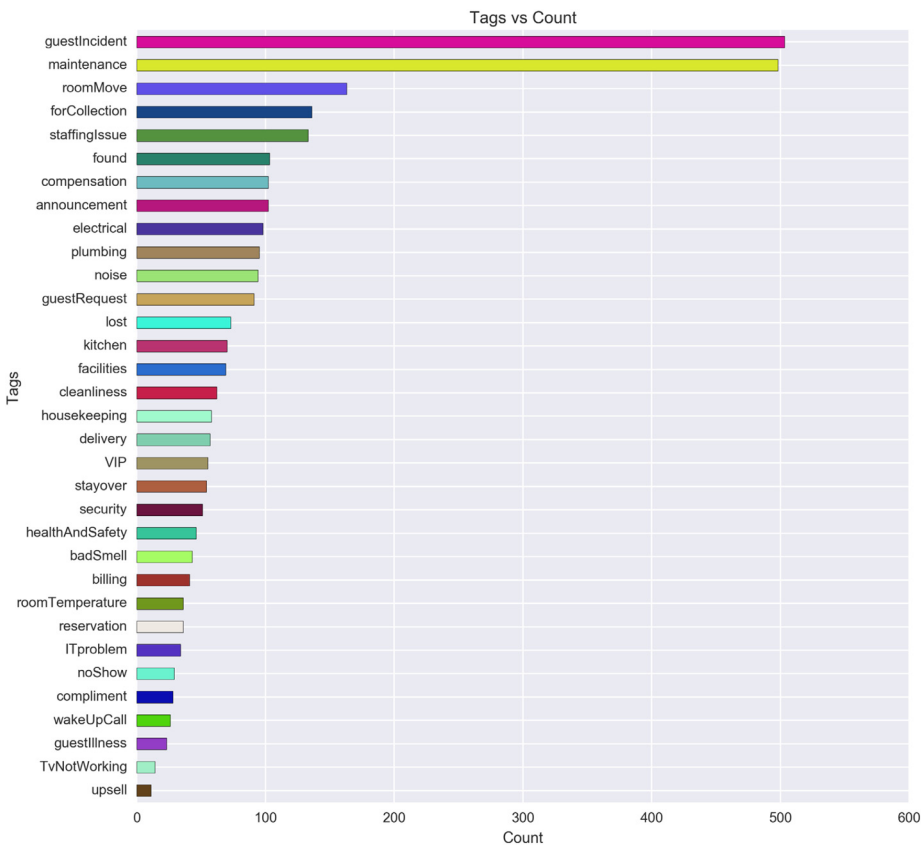


Figure 2.
An overview of all
dmbook pro tags
sorted by the
frequency of usage

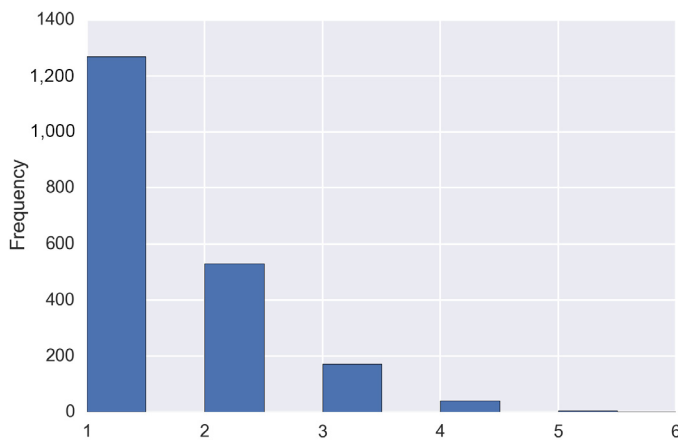


Figure 3.
Amount of tags for
each entry

update the set of tags used for tagging and rerun training so that they would be made available for recommendation by the system.

An alternative view of how tags are related to the amount of their occurrence in the set of total tags, as shown in Figure 4. We can see the percentage of a tag occurring alone in blue or with another tag in green, two other tags in red, etc. Hence, we can easily find out which of the tags are mostly used alone or which are only used in combination with others. For example, tags mostly used alone are *wakeUpCall*, *staffingIssue*, *noShow* and *delivery*. Whereas, tags that are used in combination with others are, e.g. *plumbing*, *noise*, *roomTemperature*, *electrical* and *TvNotWorking*. As mentioned earlier, this indicates general tags, which are preferably used alone, and specific tags, which normally appear in combination with others. Specific tags also indicate a more precise or exact tagging process, which could suggest reward users if we want to encourage this type of tagging behaviour.

4.2 The StackOverflow data set

The StackOverflow data set is a data dump of stackoverflow.com and contains questions and answers on the StackOverflow platform for the time span from 18 February to 7 June in the year 2009.

It contains the following features:

- *title*: The title of the question.
- *qid*: A unique identifier of the question.

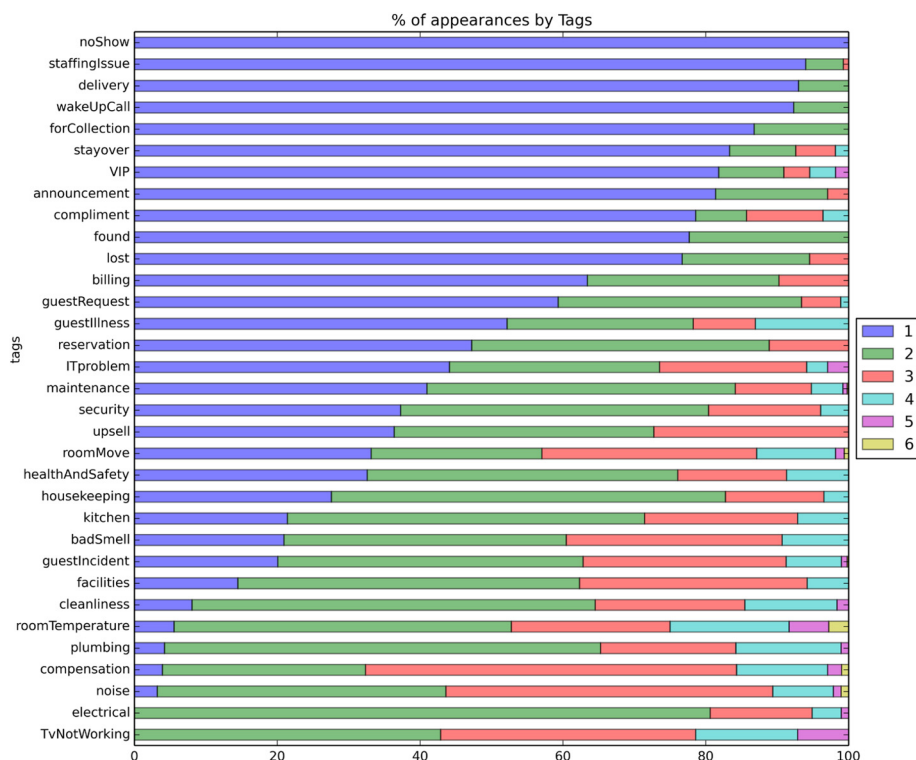


Figure 4.
Tags of ticketing
entries in dmbook pro
compared to the
amount of tags in
total measured when
they appear in the list
of tags

- *i*: A user identifier of the user who posted the question.
- *qs*: Score for the respective question based on the user rating.
- *qt*: Time when the question was posted, measured in epoch time.
- *tags*: The list of tags which the question has been tagged with, e.g. “html”, “R”, “mysql”, “python”, etc.; most questions have been tagged by two to six tags each.
- *qvc*: Amount of views for the question.
- *qac*: Amount of answers for the question.
- *aid*: Unique identifier for the answer.
- *j*: User identifier of the user who answered the question.
- *as*: Score for the answer given.
- *at*: Time when the answer was given.

An example of four different data points is shown in [Table 2](#). For example, the first example shows a best practices entry with question score of 3, the tags “best-practices” and “codesmell”, a question view count of 417, and an answer score of 1. As these features are the most meaningful ones, they will be our main features for predicting tags. Therefore, for this data set, we are using the score, time, number of views and number of answers to predict tags.

[Figure 5](#) shows the ten most popular tags in this data set. As we can see, with the exception of the tag *subjective*, all of them are programming languages. However, the data set includes 14,248 unique tags and 263,540 data points.

Finally, [Figure 6](#) shows a proportion map for StackOverflow tags that occur more than 2,000 times in the data set. In comparison to the dmbook pro data set, in the StackOverflow data set, the highest number of tags per entry is five. As we can see in the proportion map, most entries occur together with one or two others; however, it is very rare that there are four or more tags for an entry. Interestingly, some tags never occur alone (such as mysql or jsn), but other occur alone in more than 40 per cent of cases (such as python or c++).

5. Tag prediction approaches

In the section, we are giving a detailed description of the selected models and algorithms that we have tested and validated to find a recommendation of the best approach to solve the underlying problem. Our models represent their respective fields of research and span over different types of possible solutions to cover a broad area of research. Hence, we have applied and validated a multi-label classifier representing a supervised machine learning algorithm, a general statistical model which uses frequency heuristics, two similarity-based classifiers which are nearest centroid and *k*-nearest neighbours and lastly, one probabilistic classifier, namely, naïve Bayes.

In the following subsections, we describe each of the validated models in detail.

5.1 Multi-label classification

A multi-label classifier is a supervised machine learning model, and a detailed overview can be found in [Read et al. \(2011\)](#). Our implementation of this classifier is it uses gradient boosting, which is provided by scikit-learn [7]. Gradient boosting is usually performed on training data to improve the prediction results of the classifier. The implemented multi-label classifier uses an ensemble approach where a prediction model is built for every tag, and several classifiers, which are members of the ensemble, combine their outputs to decide

title	qid	i	qs	qt	tags	qvc	qac	aid	j	as	at
Data Access Layer, Best Practices	563493	25517	3	1235003886	best-practices, codesmell	417	9	563543	67561	1	1235005680
Table Within Div Firefox vs Chrome	563844	8409	1	1235014215	css,html, design, compatibility,floats	255	6	563851	65663	0	1235014479
Choosing specific Environment	564515	58175	2	1235035597	java, spring, ant, build-process, packaging	390	9	573106	37213	0	1235226171
to build using Apache ANT											
Where is the jar files cached	564936	60956	5	1235044226	java, web, start, jnlp	163	4	565132	16883	1	1235047138
for Java Web Start/JNLP applications?											

Table 2.
Snippet of data from
the StackOverflow
data set

Figure 5.
Overview of the ten
most popular tags in
the StackOverflow
data set

which tags are included in the list of predicted tags according to their fit to the textual content. This approach is very common in the area of multi-class classification.

5.2 Frequency heuristics

A statistical analysis model with frequency heuristics has been used to find co-occurrence of tags based on single words and phrases in the analysed textual content. The frequency

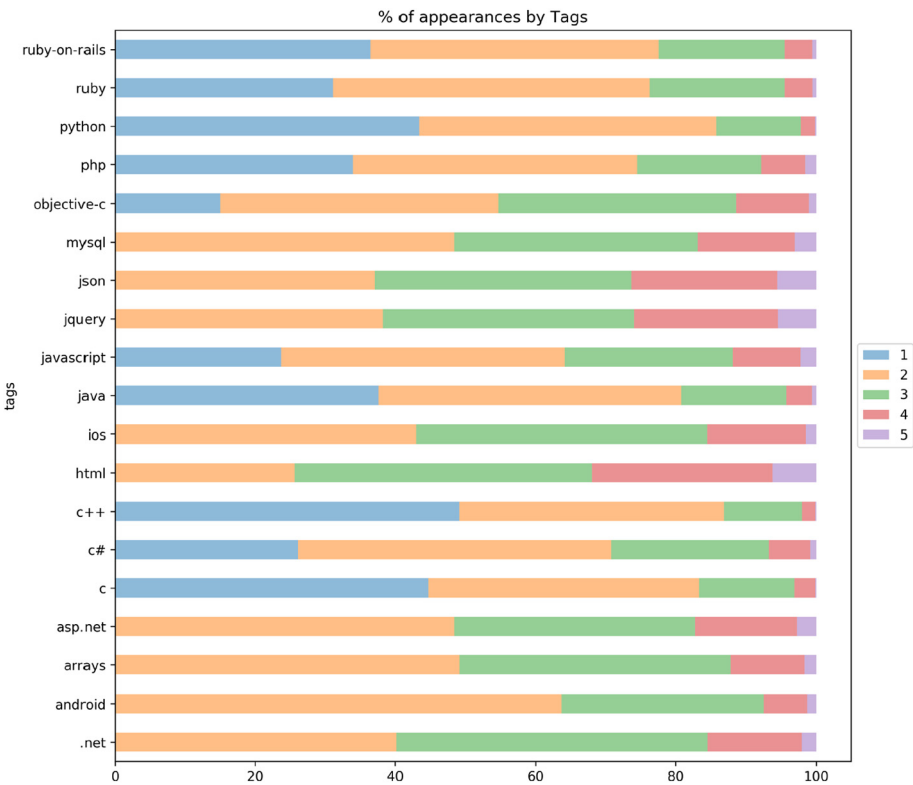
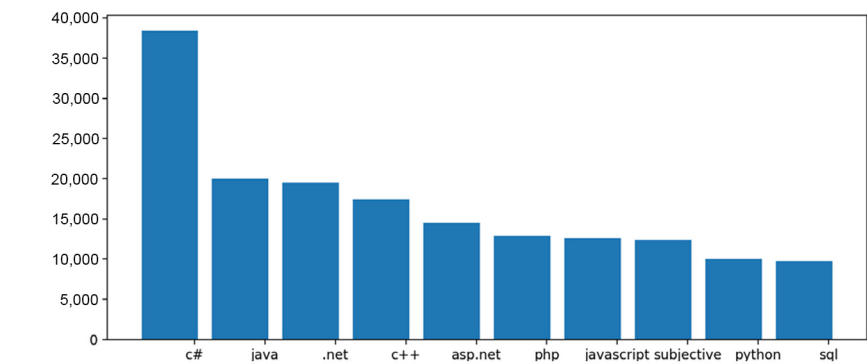


Figure 6.
Proportion map of all
entries and the
number of total tags
they appear in for
StackOverflow

heuristics algorithm browses the training data and creates a probability map based on frequency of occurring words. Each word is associated with tags that are a potential fit, and a probability of co-occurrence of tag and word is calculated. For example, if the word *broken* appears four times in the textual content of a ticket together with the tag *maintenance* and 11 times in total, then the probability of *broken* in the list of words for potential tag *maintenance* is 0.35. Every time a test entry is inspected, the top three tags are suggested that have the highest probability for the words of the textual content of the entry listed in the previously generated dictionary.

5.3 Nearest centroid

The nearest centroid algorithm is also known as Rocchio algorithm of classification; a detailed specification can be found at [Schütze et al. \(2008a\)](#). This algorithm creates groups or clusters of text entries from the training set that has a specific tag and finds the centroid of the cluster as a central point. Every ticket is assigned a similarity to the centroid of every cluster. The metric used is cosine similarity. Finally, the algorithm returns the three most similar tags and lists them as suggested tags.

5.4 K-nearest neighbours

One quite simple and broadly used algorithm is *k*-nearest neighbours (*k*-NN), which returns the IDs of the *k* most similar entries in the training set and selects the suggested tags from the tags of these entries. The idea is that neighbours use similar tags; therefore, the closer two entries are, the more likely it will be that their tags are similar as well. Any value of *k* can be used; however, different values will lead to different results in the tagging process. Therefore, we will try different values for *k* and select the best fit. The suggested tags will have the highest frequency within the *k* number of neighbours. In our evaluation, we used a range of *k* values from 1–10 and suggested up to three most frequent tags depending on the tags of neighbours. Different versions of the algorithms are implemented in sklearn, and a description of them as well as examples are provided at [Schütze et al. \(2008b\)](#).

5.5 Naïve Bayes

The naïve Bayes model [8] consists of a set of probabilistic supervised learning algorithms that apply Bayes' theorem with the naïve assumption of conditional independence. Therefore, we are using Bayes' theorem $P(y|x_1, \dots, x_n) = \frac{P(x_1, \dots, x_n|y)}{P(x_1, \dots, x_n)}$ with the assumption that $P(x_i|y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i|y)$, and we can simplify the relationship to

$$P(x_i|x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i|y)}{P(x_1, \dots, x_n)}, \quad \text{and} \quad \text{therefore, use the classification rule}$$

$$\hat{y} = \operatorname{argmax}_y P(y) \prod_{i=1}^n P(x_i|y). \quad \text{Therefore, we can use maximum } a \text{ posteriori (MAP) to estimate}$$

$P(x_i|y)$ and $P(y)$, which is the relative frequency of class *y* in the training set. For our experiments, we are using multinomial naïve Bayes in sklearn.

6. Experiment results

In our experiments, we have measured the performance of every model and algorithm on a test set of the dmbook pro data set and calculated the recall (Section 3). In [Table 3](#), an overview of recall results for every model and algorithm is shown, and we distinguish

between test runs with and without the feature book names (the frequency heuristics model is not multivariate and therefore can only be used with one feature, namely, text content).

Table 3 also shows that k -NN seems to perform best in terms of recall. Therefore, we have conducted further experiments and increased the values of k from 1 to 10. This led us to the observation that k has direct influence on the results, and the model improves when the right value for k is selected.

While this impact is not fundamental, it can still be clearly identified, and larger data sets may even show a much more significant difference with varying values of k . In any case, our experiments have shown that for this specific task, k -NN performs best with a value of $k = 8$, and increasing it further has no additional benefits. Therefore, we conclude that this algorithm is a good fit for tag prediction on the dmbook pro data set, but the exact value of k varies and needs to be reconfirmed whenever the size or nature of the data set changes.

6.1 Book names

For the dmbook pro data set, an additional feature has been identified as adding to the quality of retrieved results, namely, the book names. Hence, we have compared all models and algorithms to their respective results, including this additional feature. Our findings are that book names have the potential to improve results for multi-label classification significantly and for k -NN slightly; however, they even have a negative effect on the nearest centroid algorithm.

6.2 Runtime

In addition to classification performance, measured as recall, we have investigated runtime performance as well. Runtime performance was measured on a PC with an Intel Core i7-7600U CPU and 2.80 GHz processor as well as with 16 GB RAM. The evaluation was performed in two different categories. We have distinguished between models and algorithms with training and real-time tag detection as well as models and algorithms that run training before they are able to classify text entries and therefore need longer on a first configuration run.

The results of models and algorithms without training are shown in Figure 7. It can be seen that they require less time by far. The graphs show us that frequency heuristics has the shortest runtime and k -NN runtime depends on whether book names are used or not. When

Table 3.
Ten-fold cross-validated recall for all methods of the dmbook pro data set with and without book names and different sizes of k for k -NN

Method	Recall with book names	Without
Multi-label classification	0.735	0.707
Frequency heuristics	–	0.709
Nearest centroid	0.764	0.772
k -NN	0.986	0.979
k -NN	0.987	0.981
k -NN	0.990	0.983
k -NN	0.991	0.986
k -NN	0.992	0.990
k -NN	0.994	0.991
k -NN	0.995	0.992
k -NN	0.997	0.993
k -NN	0.997	0.993
k -NN	0.997	0.9930
Naïve Bayes	–	0.982

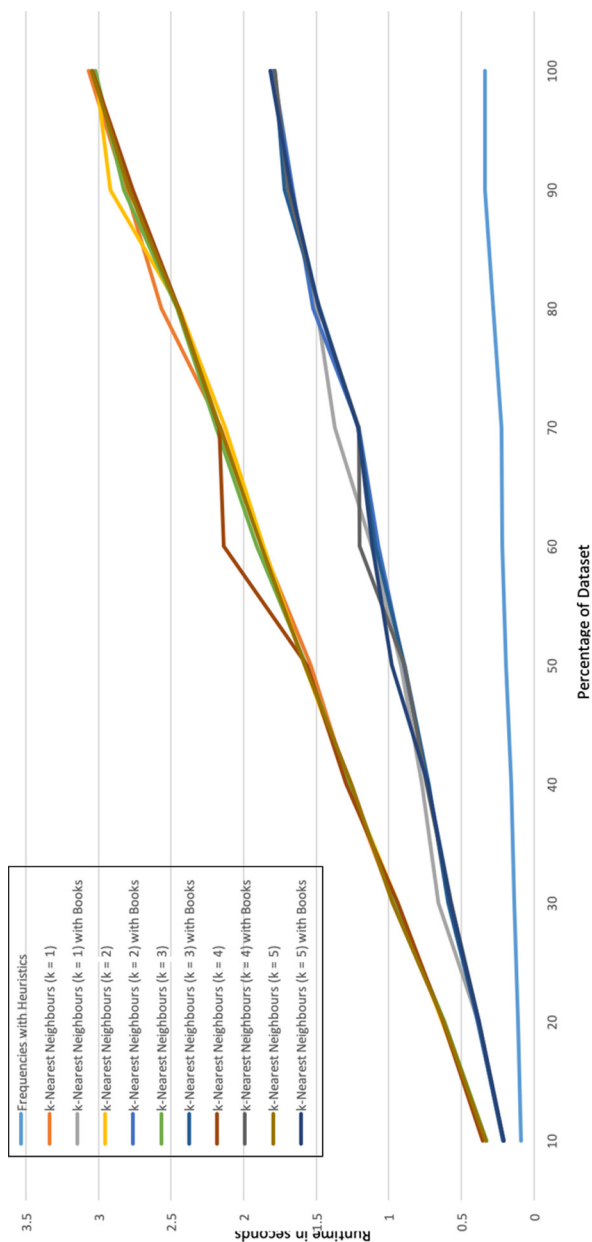


Figure 7.
Runtime for models
and algorithms
without training

book names are used, the runtime of the k -NN algorithm is shorter than without that feature. This is an unexpected result. Furthermore, increasing the value of k has not nearly as much negative impact on runtime as expected.

In contrast to the runtime performance graph with models and algorithms that require no training, Figure 8 shows an overview of models and algorithms with training times. Here, we can see that the multi-label classifier has significantly shorter runtime than the nearest centroid, and that there is almost no difference when book names are taken into account, which again is a surprising result. For the nearest centroid algorithm, including book names even improves runtime. Naïve Bayes is only slightly slower than the multi-label classifier, which performs best amongst the methods requiring training.

To summarise the findings, we can conclude that frequency heuristics has the shortest runtime within models and algorithms without training, while the multi-label classifier offers the best runtime performance within methods that require training. Taking into account the results, however, we would suggest using k -NN as the model of choice for methods that do not require training, rather than frequency heuristics, as the advantage in runtime does not justify the significantly worst classification performance. On the other hand, the multi-label classifier would be our recommended choice for models that require training, as its results are very close the other models, but it has superior runtime compared to nearest centroid as well as naïve Bayes. In any cases, we would add book names to the set of features because it gives us better results without the disadvantage of longer runtime; on the contrary, it seems even to have a positive impact on runtime.

6.3 The StackOverflow data set

We have used the same experiments for the StackOverflow data set. However, instead of book names, we have included ratings of questions in all experiments. Table 4 shows the recall for all StackOverflow experiments, which provides slightly lower, but very similar results to dmbook pro experiments.

The most important finding here is that we were able to confirm our results with a larger data set (magnitude of 100 compared to dmbook pro), which proves our hypothesis about the suggested model.

Again, the table shows that k -NN is the best fit in terms of recall. Furthermore, we observed that raising k in our k -NN experiments improves recall, but the raise stops being significant after $k = 5$.

7. Further considerations

Considering our evaluation results, we can deduce that several models represent a good fit for the underlying problem of tag prediction; however, if we want to apply one of the models to a running system or industrial setting, we need to consider several factors and limitations.

Every model or algorithm needs a data set to train and learn tag prediction to be able to classify tags on its own. This is independent of whether the model or algorithm applies training first or in real time. The important point is that a certain concept is captured from aforementioned training data. We are forced to assume that this data is a cross-section of the population, and therefore, as sample, represents the big whole every data set is trying to model. However, as time goes by, new tickets, questions or other type of entries might be added to the data set, and this concept might change. Our model will be ignorant to these changes and will undergo a so-called *concept drift*, which is described in Gama et al. (2014). Therefore, if we want to use any of the models in production, there has to be a solution that prevents concept drift from happening by

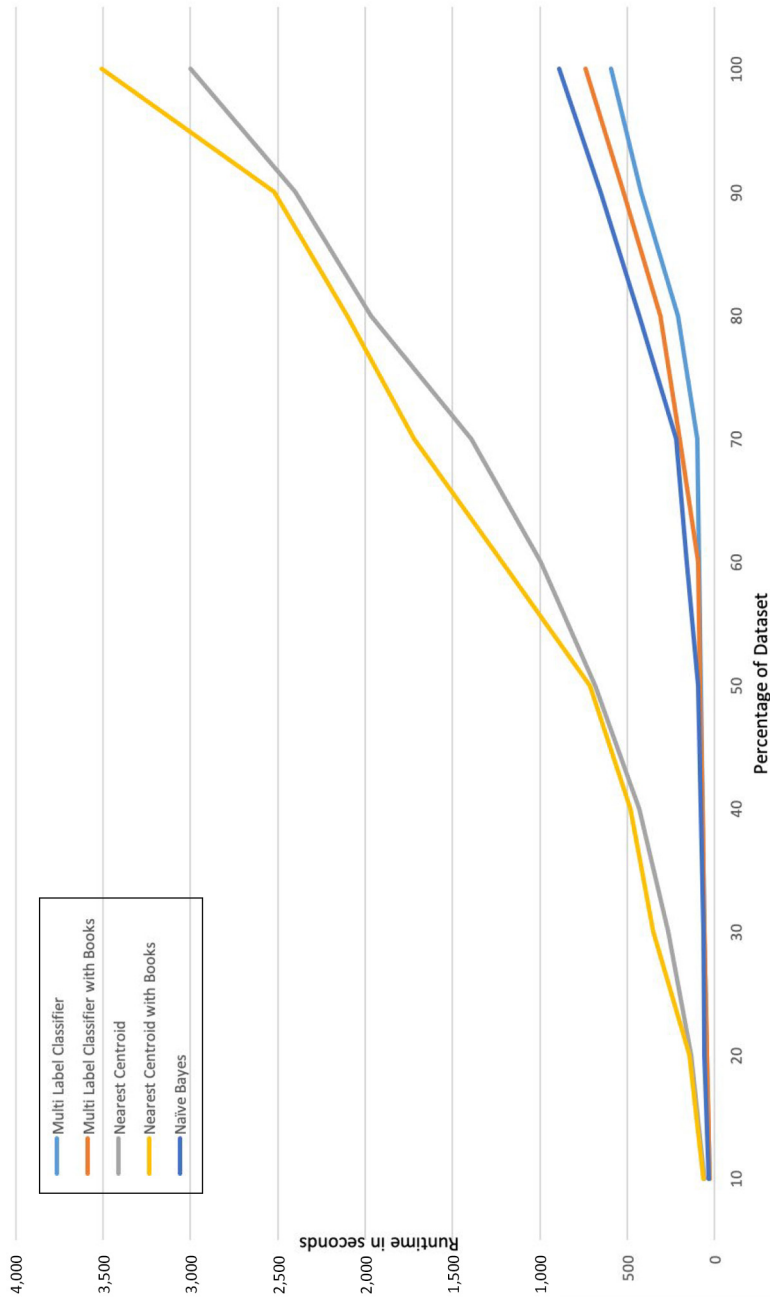


Figure 8.
Runtime for models
and algorithms with
training

keeping the model updated and the data set representative enough to keep up the quality of results.

Normally, recurring and frequent retraining of the training set as well as regular updates of the training set itself need to be part of the classification process, if we are using a classifier over a longer period. Model updates could result in new models, and these would need to replace old ones after a certain time to prevent them from becoming deprecated. In k -NN, this type of offline training would not be necessary as new data is dynamically added to the neighbourhood and the model is reconfigured every time classification happens. The only concern with k -NN is the balance between concept drift management and classification time, as increasing the size of the model and training data would also increase the time of finding the right position for a newly added neighbour. To avoid a high increase in classification time, the model needs to be pruned regularly, which would maintain an efficient training set. An example of editing algorithms to ensure this can be found in [Massie et al. \(2006\)](#).

8. Conclusion and future work

The main contribution of this paper consists of evaluating several models and algorithms for automatic tag prediction based on free textual content and several additional variables (of which, book names is the most influential one). This task has been investigated on several data sets (an industry use case data set – dmbook pro, and an open-source community data set – StackOverflow, to improve tag detection and assignment for data sets. The first part of our research consisted of the exploration of two different data sets, which helped us to gain understanding of the data in dmbook pro and StackOverflow. Furthermore, we have reviewed research papers to find candidates of models and algorithms that have the best potential of optimising both classification and runtime performance. Based on evaluation results and runtime tests, we recommend the k -NN model with the features textual content of ticket or question and book name. Our findings show that k -NN is the best candidate for classification performance, but also performs reasonably well in terms of runtime. Another significant advantage of k -NN is the robustness to concept drift; however, regular pruning and editing of the model might be needed in the case of long-term usage of the model. We also recommend further experiments to determine the exact type and time intervals these adoptions would become necessary. We have compared data sets that require offline training, such as multi-label

Table 4.
Ten-fold cross-validated recall for all methods of the StackOverflow data set with different sizes of k for k -NN

Method	Recall	k
Multi-label classification	0.702	–
Frequency heuristics	–	0.703
Nearest centroid	0.757	–
k-NN	0.963	1
k-NN	0.972	2
k-NN	0.973	3
k-NN	0.978	4
k-NN	0.981	5
k-NN	0.982	6
k-NN	0.982	7
k-NN	0.984	8
k-NN	0.983	9
k-NN	0.983	10
Naïve Bayes	0.952	–

classifier, nearest centroid and naïve Bayes, as well as “real-time” classifiers, such as a statistical analysis of frequency heuristics and k -nearest neighbours model.

In our future work, we are envisioning the following list of tasks:

- Further evaluation of the k -NN model, including additional tests for feature extraction depending on the used data set, which would help us find the optimal parameters and features for the model. This would also include testing the model on a much larger data set with a minimum size of 1,000,000 entries and especially more than 100 features. We have made the first step by including the StackOverflow data set, but this should be done on a significantly larger scale as well. Further work to exactly specify how to avoid concept drift for long-term models would complete our work in this area.
- Finally, we have implemented a prototypical software solution for the company, which is being tested and prepared for going live in a safe environment while our model is being evaluated for long-term use.

Notes

1. www.ics.uci.edu/dubois/stackoverflow/
2. www.dmbook.pro/en/demo
3. <https://stackoverflow.com>
4. <http://dbpedia.org>
5. www.wikipedia.org
6. <https://wordnet.princeton.edu>
7. <http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html>
8. https://scikit-learn.org/stable/modules/naive_bayes.html
9. www.dmbook.pro/en

References

- Allahyari, M. and Kochut, K. (2016), “Semantic tagging using topic models exploiting Wikipedia category network”, *IEEE Tenth International Conference on Semantic Computing (ICSC)*, pp. 63-70, available at: <https://doi.org/10.1109/ICSC.2016.34>
- Begelman, G. Keller, P. (2006), and F. and Smadja, “Automated tag clustering: improving search and exploration in the tag space”, *Proceedings of the Collaborative Web Tagging Workshop at the WWW, Edinburgh*, available at: http://pui.ch/phred/automated_tag_clustering/
- Belém, F.M., Almeida, J.M. and Gonçalves, M.A. (2017), “A survey on tag recommendation methods”, *Journal of the Association for Information Science and Technology*, Vol. 68 No. 4, pp. 830-844, available at: <https://doi.org/10.1002/asi.23736>
- Briscoe, T. and Carroll, J.A. (2002), “Robust accurate statistical annotation of general text”, in LREC.
- Campana, M. and Delmastro, F. (2017), “Recommender systems for online and mobile social networks: a survey”, pp. 75-97.
- Case study (2006), “Structured vs unstructured tagging – a case study”, *International Journal Edinburgh (2006), Archived proceedings of the WWW Tagging Workshops*.

- Cazzanti, L., Gupta, M.R. and Koppal, A.J. (2008), "Generative models for similarity-based classification", *Pattern Recognition*, Vol. 41 No. 7, pp. 2289-2297.
- Djuana, E., Xu, Y. and Li, Y. (2011), "Learning domain ontology for tag recommendation", *Science and Technology*, Vol. 1, available at: <http://eprints.qut.edu.au/45499/>
- Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M. and Bouchachia, A. (2014), "A survey on concept drift adaptation", *ACM Computing Surveys*, Vol. 46 No. 4, pp. 44:1-44:37, available at: <https://doi.org/10.1145/2523813>
- Golder, S.A. and Huberman, B. (2005), "The structure of collaborative tagging systems", 32, September.
- Kazi, S.H. and Ng, V. (2014), "Automatic keyphrase extraction: a survey of the state of the art", *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, Baltimore, pp. 1262-1273, available at: www.aclweb.org/anthology/P14-1119
- Krestel, R. and Fankhauser, P. (2009), "Tag recommendation using probabilistic topic models", ECML PKDD Discovery Challenge, p. 131.
- Kutner, M.H., Nachtsheim, C.J., Neter, J. and Li, W. (2005), *Applied Linear Statistical Models*, 5. McGraw-Hill Irwin Boston.
- Lin, H., Davis, J., (2009), and Y. and Zhou, "An integrated approach to extracting ontological structures from folksonomies", in Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M. and Simperl E. (Eds), *The Semantic Web: Research and Applications*, Springer, Berlin, Heidelberg, pp. 654-668.
- Massie, S., Craw, S., (2006), and N. and Wiratunga, "Complexity profiling for informed case-base editing", in Thomas, R., Göker, M.H. and Güvenir HA (Eds), *Advances in Case-Based Reasoning*, Springer, Berlin, Heidelberg, pp. 325-339.
- Miner, G., Elder, J., Fast, A., Hill, T., Nisbet, R. and Delen, D. (2012), *Practical Text Mining and Statistical Analysis for Non-Structured Text Data Applications*, Academic Press Cambridge.
- Noruzi, A. (2006), "Folksonomies: (Un)controlled vocabulary?", p. 33, January.
- Pascal, S. and Mineau, G.W. (2001), "A simple KNN algorithm for text categorization", *Proceedings IEEE International Conference on Data Mining. IEEE*, pp. 647-648.
- Preisach, C., Leandro, B.M. and Schmidt-Thieme, L. (2010), "Semi-supervised tag recommendation – using untaged resources to mitigate cold-start problems", *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 6118 LNAI*, 1, pp. 348-357.
- Prokofyev, R., Boyarsky, A., Ruchayskiy, O., Aberer, K., Demartini, G., (2012), and P. and Cudré-Mauroux, "Tag recommendation for large-scale ontology-based information systems", *Proceedings of the 11th International Conference on The Semantic Web – Volume Part II*, pp. 325-336.
- Read, J., Pfahringer, B., Holmes, G. and Frank, E. (2011), "Classifier chains for multi-label classification", *Machine Learning*, Vol. 85 No. 3, p. 333.
- Rish, I. (2001), "An empirical study of the naive bayes classifier", *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, Vol. 3, pp. 41-46.
- Schütze, H., Manning, C.D. and Raghavan, P. (2008a), *Introduction to Information Retrieval*, Vol. 39, Cambridge University Press, Cambridge, pp. 292-297.
- Schütze, H., Manning, C.D. and Raghavan, P. (2008b), *Introduction to Information Retrieval*, Vol. 39, Cambridge University Press, Cambridge, pp. 297-301.
- Shepitsen, A., Gemmell, J., Mobasher, B., (2008), and R.D. and Burke, "Personalized recommendation in social tagging systems using hierarchical clustering", in Pearl P., Bridge, D.G., Mobasher, B. and Ricci F. (Eds), *Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys, ACM, Lausanne*, pp. 259-266.

-
- Smith, G. (2007), *Tagging: People-Powered Metadata for the Social Web*, first ed., New Riders Publishing, Thousand Oaks, CA.
- Song, Y., Zhang, L. and Lee Giles, C. (2011), "Automatic tag recommendation algorithms for social recommender systems", *ACM Transactions on the Web*, Vol. 5 No. 1, pp. 1-31.
- Stanley, C. and Byrne, M. (2013), "Predicting tags for stackoverflow posts", *Proceedings of the International Conference on Cognitive Modelling, ICCM*, pp. 414-419.
- Trant, J. (2009), "Studying social tagging and folksonomy: a review and framework", *Journal of Digital Information*, Vol. 10, p. 1.
- Tsoumakas, G. and Katakis, I. (2007), "Multi-label classification: an overview", *International Journal of Data Warehousing and Mining (IJDWM)*, Vol. 3 No. 3, pp. 1-13.
- Wang, X., Zhang, D., Gu, T. and Hung, K.J. (2004), "Ontology based context modeling and reasoning using OWL", *Percom workshops, Citeseer*, Vol. 18, p. 22.

Corresponding author

Bojan Bozic can be contacted at: bojan.bozic@dit.ie