Multi-Label Classification of Research Papers Using Multi-Label K-Nearest Neighbour Algorithm

Shurui Li\textsuperscript{1,a,†} and Jiechen Ou\textsuperscript{2,b,†}

\textsuperscript{1}Bell Honors School, Nanjing University of Posts and Telecommunications, Nanjing, 210023, China;
\textsuperscript{2}School of Management and Engineering, Nanjing University, Nanjing, 210023, China.
\textsuperscript{a}Q18010118@njupt.edu.cn, \textsuperscript{b}181870170@smail.nju.edu.cn
\textsuperscript{†}These authors contributed equally.

Abstract. With the frequent interaction and cooperation between different disciplines in recent years, the number of research papers associated with multiple subjects increased. Correspondingly, some of the existing literatures belong to a single discipline, while others may simultaneously involve more than 2 subjects. At this time, the traditional single-label text classification is not conducive to people obtaining comprehensive and cutting-edge research papers in real life. Thus, it's of great importance to conduct a multi-label classification of research papers effectively. This paper tests the performance of multi-label learning tasks with text data obtained from the Kaggle website. Firstly, lemmatization and Term Frequency-Inverse Document Frequency (TF-IDF) are used for feature extraction in the pre-processing part. The critical information of text content is statistically analysed, and text content is converted into numerical and high-dimensional vector space. As the traditional single-label classification algorithm is not suitable for the above problem, this paper adopts the Multi-Label K-Nearest Neighbour (ML-KNN) algorithm framework for classification. Experimental results report that the ML-KNN algorithm has achieved better results in multi-label text classification problems than a traditional multi-label algorithm, which proves the effectiveness of the ML-KNN algorithm for text data prediction with multiple subjects. Moreover, the work in this paper is analysed and summarized.

1. Introduction
The classification of research papers analyzes the subject attributes and categories of the content of research papers, prompt the collection of books, and organize them into categories. The classification of research papers is very detailed and academic work. Determining which subject field or several subject fields a research paper belongs has important research value and practical significance.

Nowadays, research on interdisciplinary fields is very hot, so a paper often belongs to multiple categories [1]. For example, a paper titled 'Density large deviations for multidimensional stochastic hyperbolic conservation laws' belongs to Physics and Mathematics. The traditional classification assumes that each sample is assigned to one and only one label, which does not meet our actual needs, while multi-label classification can [2].

Multi-label classification is more difficult than traditional classification problems. In a traditional classification problem, each example is assumed to be classified into one of a number of predefined exclusive categories [3, 4]. However, in multi-label classification problems, the situation is completely
different. Assuming there are $L$ categories, there will be $2^L$ possible multi-labelled classes [5]. The number of labels is uncertain, some samples may have only one class label, and some samples may have all class labels. Also, there is a mutual dependence between class labels [6]. For example, a sample that contains blue sky class labels has a high probability of containing white clouds. How to solve the dependency problem between class labels is also a great difficulty.

Conventionally, a binary classification approach has been used, in which the multi-label classification problem is decomposed into independent binary classification problems [7, 8]. These methods are based on problem transformation. After transformation, traditional classification methods can be used to solve problems, such as support vector machines, naïve Bayes classifiers, and decision trees [9]. However, since the binary approach does not consider connections between the labels and a generative model of multi-labelled text, it has an important limitation when applied to the multi-labelled text categorization.

RAKEL (Random K-Label sets) [10] randomly selects a small subset of labels from the original label set to train each classifier. However, due to the randomness of the label space constructed by the RAKEL, the dependence between labels has not been fully investigated, resulting in low classification accuracy and a certain impact on generalization performance.

The core idea of Calibrated Label Ranking [11] is to decompose the multi-label classification problem and convert it into a label sorting problem. After sorting the label values, the output is the higher-ranked labels. Calibrated Label Ranking considers the pairwise combination of labels. The final model has relatively good generalization ability. The disadvantage is that only the combination of two labels is considered, and all the dependencies between labels are not considered.

Multi-label KNN [12] is derived from the K-nearest neighbor (KNN) algorithm and suitable for multi-label classification problems. ML-KNN makes use of statistical information gained from the label sets of the neighboring instances. The maximum a posteriori (MAP) principle is utilized to determine the label set for the unseen instances. ML-KNN considers the information of the surrounding samples, which contains the relationships between labels.

When ML-KNN is applied to classify research papers, it is inefficient to use it directly since there is no publicly processed data set. Therefore, this paper maps textual data to high-dimensional vector space on the first step.

The main contributions of this work can be summarized as follows:
1. The method maps textual data into high-dimensional vector space and retains more feature information.
2. The model makes a better prediction on research papers classification by comparing the results of ML-KNN and other common algorithms.

2. Methods

2.1. Data preparation and pre-processing

Dataset comes from Kaggle [13]. It contains 20972 accurate research paper samples with specific subjects like Computer Science, Physics, Mathematics, Statistics, Quantitative Biology, Quantitative Finance, e.g., the data structure is shown in Table 1. The number under the subject category in the table represents whether the article belongs to the subject category, the number 1 represents Yes, and the number 0 represents No.

| TABLE 1. The structure of classified text data |
|-----------------------------------------------|
| TITLE | ABSTRACT | Computer Science | Physics | Mathematics | Statistics | Quantitative Biology | Quantitative Finance |
|-------|----------|-------------------|---------|-------------|------------|----------------------|----------------------|
| T1    | Summary  | 1                  | 0       | 0           | 1          | 0                    | 1                    |
| T2    | Summary  | 0                  | 0       | 1           | 0          | 1                    | 0                    |
| Tn    | ……       | ……                | ……      | ……          | ……         | ……                   | ……                   |
In this paper, the first 10,000 instances from the complete classified dataset are used to evaluate the performance of multi-label learning tasks. These 10,000 instances will be split into training data (80%) and validation dataset (20%) randomly. Figure 1 reports a statistics summary of these 10,000 experimental data, of which the sample averages of words in abstract 149 and the sample variance is 60. Besides, the normal distribution fitting of the density histogram is carried out, which roughly states that the words in the text abstract obey the normal distribution. Table 2 statistics the distribution of subjects from experimental data and Table 3 reports the number of the text belonging to more than one discipline respectively, proving that text categorization for this dataset is a multi-label classification task.

Figure 1. Distribution of the number of words in abstracts

| Subject                  | Number of articles per subject |
|--------------------------|--------------------------------|
| Computer Science         | 4062                           |
| Physics                  | 2869                           |
| Mathematics              | 2702                           |
| Statistics               | 2500                           |
| Quantitative Biology     | 287                            |
| Quantitative Finance     | 120                            |

2.2. Feature extraction
The title and abstract of a research paper are textual data, which ML-KNN cannot directly use, so data preprocessing is necessary. In this part, previous papers map the title and abstract of each research paper to the high-dimensional vector space at a semantic level by Lemmatization [14] and TF-IDF [15] to make data dimension reduction. The traditional vector space model matches the single word, i.e., matching the word input by a user and the word existing in the vector space. Due to polysemy and synonymy, the model cannot provide information at the semantic level. For example, in the traditional vector space, a research paper using the keyword mobile phone and a paper using the keyword cellphone may be far away. But in fact, these two research papers are similar.

Lemmatization in linguistics is the process of grouping together the inflected forms of a word to be analysed as a single item, identified by the word’s lemma or dictionary form.

2.2.1. Lemmatization. In many languages, words appear in several inflected forms. For example, in English, the verb ‘to walk’ may appear as ‘walk’, ‘walked’, ‘walks’, or ‘walking’. The base form, ‘walk’, that one might look up in a dictionary, is called the lemma for the word. The association of the base form with a part of speech is often called a lexeme of the word.

Lemmatization is closely related to stemming. The difference is that a stemmer operates on a single word without knowledge of the context, and therefore cannot discriminate between words that have different meanings depending on the part of speech. However, stemmers are typically easier to implement and run faster. The reduced "accuracy" may not matter for some applications. In fact, when used within information retrieval systems, stemming improves query recall accuracy, or true positive rate, when compared to lemmatisation. Nonetheless, stemming reduces precision or the proportion of positively-labeled positive instances for such systems [16-18].

2.2.2. TF-IDF. TF-IDF is a statistical method used to evaluate the importance of a word. The importance of a word increases in proportion to the number of times it appears in the document, but at the same time, it decreases in inverse proportion to the frequency of its appearance in the corpus. 

TF is short for term frequency. Term frequency, \( tf(t,d) \), is the frequency of term \( t \), as formula (1),

\[
 tf(t,d) = \frac{f_{t,d}}{\sum_{t',d} f_{t',d}} \quad (1)
\]

where \( f_{t,d} \) is the raw count of a term in a document, i.e., the number of times that term \( t \) occurs in document \( d \).

The IDF (Inverse Document Frequency) measures how much information the word provides, i.e., if it is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term and then taking the logarithm of that quotient), formula (2) as follows:

\[
 idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|} \quad (2)
\]

where \( N \) is the total number of documents in the corpus \( N = |D| \). \(|\{d \in D : t \in d\}|\) is the number of documents where the term \( t \) appears.

Then TF-IDF is calculated as formula (3),

\[
 tfidf(t,d,D) = tf(t,d) \cdot idf(t,D) \quad (3)
\]

A high term frequency in a particular document and a low document frequency of the term in the entire document set can produce a high-weight TF-IDF. Therefore, TF-IDF tends to filter out common words and keep important words.
2.3. **ML-KNN**

ML-KNN is a non-parametric classification approach proposed by Zhi-Hua Zhou and his group. It is a more flexible and functional lazy learning method derived from k-NN. In traditional k-NN classification, the plurality vote of the nearest neighbors is utilized to decide on labels. In detail, new instances are classified by assigning the most frequent or common label among the k training samples nearest to that query point. When it comes to the ML-KNN model, conditional probability determines whether the sample belongs to a class label or not. Specifically, it associates the sample with a particular class for each unseen instance as long as the posterior probability of the event that a certain instance belongs to a class is greater than not. Furthermore, according to the Bayesian rule, the classifier can get the prior probability from training data directly and make multi-label judgement efficiently.

2.3.1. **K nearest neighbours.** Let \( L = \{1,2,3,...,K\} \) be the finite set of labels within a specific classification task, and \( l_t \) \((l_t \in L)\) represents the label set of a training sample \( t \). To gain statistics information from the training set, let \( C(x) \) be member set of the k nearest neighbors of instance \( x \) in the training data. In addition, \( \bar{Z}_x \) is defined as the category vector, for instance \( x \), where the \( y^{th} \) element \( \bar{Z}_x(y) \) \((y \in L)\) takes value of 1 if \( y \in l \) and 0 otherwise. Thus, given the former definition and relationship, a neighbors counting vector can be defined as formula (4),

\[
\bar{D}_j(y) = \sum_{a \in C(x)} \bar{Z}_a(y), \ y \in L
\]

(4)

2.3.2. **Maximum a posteriori (MAP) principle.** Similar to the traditional binary KNN algorithm, ML-KNN identify the k nearest neighbours \( C(j) \) in the training set for each new instance \( j \) firstly. In order to assign the instance to more than one possible class simultaneously, ML-KNN aims to figure out the biggest posterior probability and make the decision. Let \( \gamma_j(y) \) be the posterior probability of the event that \( j \) belongs to label \( y \) \((y \in L)\) given the statistics information, e.g., the number of neighbouring instances belonging to each possible class. Thus, let \( S^y_i \) \((i \in \{0,1,2,...,Q\})\) denote the event that there are exactly \( i \) instances belonging to label \( y \) among the KNNs of \( j \). Meanwhile, whether an instance belongs to a certain label is obviously a binary choice in each iteration. Hence, \( E^y_i \) is defined as the event that \( j \) has label \( y \) and \( E^y_i \) as the event that \( j \) has not label \( y \). Subsequently, based on the neighbour counting vector \( D(j) \) and statistical data \( S^y_i \), ML-KNN determines whether the new sample is associated with multiple classes using the following maximum a posteriori principle, as formula (5):

\[
\bar{Z}_j(y) = \arg \max \ P\left( E^y_b \mid S^y_{D(y)} \right), \ b \in \{0,1\}, \ y \in L
\]

(5)

Eq (2) can be rewritten using the Bayesian rule as formula (6),

\[
\bar{Z}_j(y) = \arg \max \frac{P\left( S^y_{D(y)} \mid E^y_b \right) \cdot P\left( E^y_b \right)}{P\left( S^y_{D(y)} \right)}
\]

(6)

\[
= \arg \max \frac{P\left( S^y_{D(y)} \mid E^y_b \right) \cdot P\left( E^y_b \right)}{P\left( S^y_{D(y)} \right)}
\]

The real-valued function computing posterior probability can be defined as formula (7),

\[
\gamma_j(y) = P\left( E^y_b \mid S^y_{D(y)} \right) = \frac{P\left( S^y_{D(y)} \mid E^y_b \right) \cdot P\left( E^y_b \right)}{P\left( S^y_{D(y)} \right)}
\]

(7)

\[
= \frac{P\left( S^y_{D(y)} \mid E^y_1 \right) \cdot P\left( E^y_1 \right)}{P\left( S^y_{D(y)} \mid E^y_1 \right) \cdot P\left( E^y_1 \right) + P\left( S^y_{D(y)} \mid E^y_0 \right) \cdot P\left( E^y_0 \right)}
\]
3. Results and Discussions

3.1. Performance Metrics

Before discussing the ML-KNN algorithm’s performance, five popular indicators proposed in Ref [2] are first elaborated to evaluate multi-label classification learning tasks. These performance metrics mainly compare the difference of label relevance and posterior probability ranking between the actual label set of the sample and the prediction set obtained by the classifier. For a test set \( S = \{ (x_1,l_1), (x_2,l_2), \ldots, (x_T,l_T) \} \) where \( l_i \) denotes the proper category vector of the instance \( x_i \), the following metrics are used in this paper:

1. Hamming loss was originally used in text classification algorithms. Formula (8) calculates the average number of the event that instance labels of the test set are incorrectly classified, including an instance belonging to the label is not correctly predicted or an instance not belonging to the label is precisely associated with that label. The smaller the value of Hamming loss is, the greater the similarity between the two sets and the better the classification performance is:

\[
\text{HLoss}(h) = \frac{1}{T} \sum_{i=1}^{T} \frac{1}{K} |h(x_i) \Delta l_i|
\]

(8)

In formula (8), \( T \) stands for the total amount of test set. \( \Delta \) stands for the symmetric difference between two sets. Then, \( h(x_i) \) denotes the prediction category vector obtained by the classifier. While the hamming loss is based on the multi-label classifier \( h(\cdot) \), the following metrics are defined based on the real-valued function \( \gamma_{x_i}(\cdot) \) which concern the ranking quality of different labels for each instance.

2. One-error calculates the average times that the top-ranked label is not in the set of proper labels of the instance. The ranking function refers to sorting the real-value function \( \gamma_{x_i}(\cdot) \) in a specific descending order and then outputs the index value, such that if \( \gamma_{x_i}(y_1) > \gamma_{x_i}(y_2) \) then \( \text{rank}(x_i,y_1) < \text{rank}(x_i,y_2) \) \((y_j \in L)\). The smaller the value of one-error is, the better the performance is. The performance is perfect when one-error = 0. One-error shows as formula (9):

\[
\text{One-error}(\gamma) = \frac{1}{T} \sum_{i=1}^{T} \left\lfloor \arg \max_{y \in l_i} \gamma_{x_i}(y) \right\rfloor \notin l_i
\]

(9)

3. Coverage evaluates how far it is needed to go on average to cover all the correct tags of the instance. It can be easily calculated with the ranking function. The smaller the value of coverage is, the better the performance is. Coverage shows as formula (10):

\[
\text{Coverage}(\gamma) = \frac{1}{T} \sum_{i=1}^{T} \max_{y \in l_i} \text{rank}(x_i,y) - 1
\]

(10)

4. Ranking loss calculates the average fraction of label pairs that are reversely ordered, for instance. It focused on evaluating the degree of correlation between different labels and was originally used in the Rank-SVM learning tasks. The performance is perfect when Ranking loss = 0. The smaller the value of the Ranking loss is, the better the performance is. Ranking loss shows as formula (11):

\[
\text{Rloss}(\gamma) = \frac{1}{T} \sum_{i=1}^{T} \frac{1}{|l_i|} \sum_{y \in l_i} \left| \text{rank}(x_i,y) \leq \text{rank}(x_i,y') \right| \left| (y_1,y_2) \in l_i \times l_i \right|
\]

(11)

5. Average precision calculates the average proportion of labels ranked above a particular label \( y \) \((y \in l_i)\) which actually are in the correct labels set. The bigger the value of Average precision is, the better the classification performance is. Average precision shows as formula (12):

\[
\text{avgprec}(\gamma) = \frac{1}{T} \sum_{i=1}^{T} \frac{1}{|l_i|} \sum_{y \in l_i} \left\lfloor \frac{\left| \{ y' \mid \text{rank}(x_i,y') \leq \text{rank}(x_i,y), y' \in l_i \} \right|}{\text{rank}(x_i,y)} \right\rfloor
\]

(12)
The above five metrics can comprehensively evaluate the performance of a multi-label classifier. However, it is difficult to ensure that all these five indicators achieve the optimal value, as the measurement and method of each metric are distinct.

3.2. Experimental results

In this paper, the effectiveness of multi-label classification learning tasks is evaluated by predicting the subjects of the articles mentioned above. Recalling the data set, words in both titles and abstracts are firstly converted into standardized numerical feature vectors by morphological reduction and TF-IDF. Table 4 reports the corresponding feature matrix of the text set after preprocessing, where $A_{ij}$ is equal to a particular value if the word $j^{th}$ used in the article $i^{th}$, otherwise it sets to 0.

| Vector space | Str1  | Str2  | Str3  | Str4  | Str5  | Str6  | Str7  | Str8  | Str9  | …    | Str34633 |
|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|
| Article1    | 4.483 | 30.570| 18.022| 6.671 | 5.801 | 8.373 | 4.122 | 2.206 | 12.935| …    | …        |
| Article2    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | …    | …        |
| Article3    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 2.206 | …    | …        |
| Article4    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | …    | …        |
| Article5    | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     | …    | …        |

Similar to traditional algorithm optimizations, the optimal ML-KNN classifying model is selected by changing parameter settings after data preprocessing. For instance, the number of nearest neighbors identified ($k$) value specifies the range of adjacent training samples for each new one. At the same time, the distance metrics are the standard to compute samples' similarity. Thus, various parameter determinations may lead to a variety of label combinations in adjacent tag sets, which changes the prediction results of the classifier and affects the performance of the algorithm. In classification problems, Euclidean distance and cosine similarity are the most commonly used distance metrics. The $k$ value is determined by simple cross validation. Based on the above preprocessing and parameter thinking, this paper will compare the prediction performance of the ML-KNN algorithm using Euclidean distance and cosine similarity, respectively. Subsequently, the better method of ML-KNN is compared with other popular multi-label learning tasks like Decision Tree and Extra Tree to find the most efficient algorithm for text categorization.

| Evaluation criterion | K = 3 | K = 4 | K = 5 | K = 6 | K = 7 | K = 8 | K = 9 | K = 10 |
|----------------------|-------|-------|-------|-------|-------|-------|-------|--------|
| Hamming Loss         | 0.242 | 0.230 | 0.232 | 0.236 | 0.239 | 0.235 | 0.229 | 0.233  |
| One-error            | 0.651 | 0.642 | 0.649 | 0.694 | 0.690 | 0.676 | 0.677 | 0.676  |
| Coverage             | 1.707 | 1.667 | 1.664 | 1.707 | 1.713 | 1.682 | 1.682 | 1.692  |
| Ranking Loss         | 0.288 | 0.276 | 0.275 | 0.287 | 0.289 | 0.283 | 0.285 | 0.286  |
| Average precision    | 0.576 | 0.582 | 0.580 | 0.560 | 0.561 | 0.570 | 0.570 | 0.569  |

This paper suppose that Hamming Loss and Average precision are relatively important than the other three indicators. Table 5 reports how the performance of ML-KNN using Euclidean distance varies when the $k$ value changes from 3 to 10. Besides, Figure 2 shows the performance fluctuations of ML-KNN when using Euclidean distance and cosine similarity, where the $k$ value does not affect Hamming Loss of these two types of distance metrics significantly. More strictly, the MK-KNN algorithm using Euclidean distance obtains a lower average Hamming Loss than that of the other method. It takes a relatively low value when $k$ is equal to 4. The impact of the $k$ value on Average precision is more meaningful, noting that the metric with Euclidean distance is generally higher, and it
reaches the top when $k$ is equal to 4. Thus, the performance of ML-KNN when $k$ set to 4 will be the optimal result in the data set and will be utilized in subsequent comparison with other algorithms. Given the former result, the ML-KNN is more effective when calculating the nearest neighbor according to Euclidean distance.

Figure 2. Experimental results of ML-KNN with different distance metrics

Table 6 shows the comparison of the performance of different algorithms for text prediction. Obviously, the ML-KNN algorithm has significant advantages in these five evaluation criteria. In the next paragraph, the reason why various distance metrics and algorithms lead to a big difference in evaluation will be elaborated in more detail.

| Evaluation criterion | Algorithm     |
|----------------------|---------------|
|                      | ML-KNN        | Decision Tree: | Extra Tree | KNN         |
| Hamming Loss         | 0.230         | 0.279          | 0.286      | 0.281       |
| One-error            | 0.642         | 0.665          | 0.681      | 0.728       |
| Coverage             | 1.667         | 3.22           | 3.290      | 3.051       |
| Ranking Loss         | 0.276         | 0.586          | 0.602      | 0.542       |
| Average precision    | 0.582         | 0.448          | 0.432      | 0.456       |

3.3. Discussions
In the experiment, this paper used two metrics, Euclidean distance, and cosine similarity, for comparison. In all experiments, the algorithm using Euclidean distance gets better results. Also, no matter which kind of metric, the performance is better than other algorithms which is used for comparison. Next, the focus of the discussion is why the Euclidean distance is better than cosine similarity.

Euclidean distance measures the absolute distance of points in space, which is directly related to the position coordinates of each point. Cosine similarity uses the cosine value of the angle between two vectors in the vector space to measure the difference between two individuals. Compared with the distance measure, cosine similarity pays more attention to the difference in the direction of two vectors rather than distance or length. Suppose there are three three-dimensional vectors respectively $(10 8 9)^T$, $(4 2 3)^T$ and $(8 10 9)^T$. The vector represents the judges' scores for three contestants. In Euclidean distance, the first judge is more like the third judge. In cosine similarity, the first judge is more similar to the second judge, but the two have different standards for full scores. So, the cosine distance is more to distinguish the difference from the direction and is not sensitive to the absolute value. Usually, cosine similarity is more used to deal with the problem of inconsistent metrics that may exist among users. In this case, the problem is the difference in the number of words in the abstract. However, this problem has been solved in preprocessing. TF-IDF method is used to extract
feature words and solve this problem. As a consequence, there is no need to use cosine similarity. From another perspective, the data is papers' titles and abstracts. The use of some keywords in abstracts in the same field may be similar. If the value of the feature word in a paper is low, since IDF considers the entire corpus, it is only possible that the word frequency of the keyword is low in the abstract. Euclidean distance should also be used.

When ML-KNN is compared with other algorithms, ML-KNN has better performance. This is because ML-KNN makes use of the statistical characteristics of the vectorized data. ML-KNN utilizes the MAP principle to determine the label set for the unseen instance. ML-KNN considers the probability of the same situation in the entire data set and uses the Bayesian formula to calculate this posterior probability.

4. Conclusion
This paper applies a multi-label K-nearest neighbor algorithm to classify research papers with papers' titles and abstracts. In the data preparation, lemmatization is used to get the base form of the word and use the TF-IDF algorithm to map textual data to high-dimensional vector space, which can be used in ML-KNN. Then calculate prior probability in the training set and utilize the MAP principle to mark the label automatically. The results show that the performance using ML-KNN is much better than Decision Tree, Rank-SVM, and KNN.

References
[1] Tsoumakas, Grigorios and Ioannis Katakis. "Multi-Label Classification: An Overview." IJDWM 3.3 (2007): 1-13.
[2] Schapire, R.E., Singer, Y. BoosTexter: A Boosting-based System for Text Categorization. Machine Learning 39, 135–168 (2000).
[3] Godbole S., Sarawagi S. (2004) Discriminative Methods for Multi-labeled Classification. In: Dai H., Srikant R., Zhang C. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2004. Lecture Notes in Computer Science, vol 3056. Springer, Berlin, Heidelberg.
[4] Andrew Kachites McCallum, Multi-label text classification with a mixture model trained by EM, AAAI 99 Workshop on Text Learning.
[5] Ueda, N., and K. Saito. "Parametric Mixture Models for Multi-Labeled Text." International Conference on Neural Information Processing Systems MIT Press, 2002.
[6] Dembczyński, Krzysztof, Willem Waegeman, Weiwei Cheng, et al. "On Label Dependence and Loss Minimization in Multi-label Classification." MACHINE LEARNING 88.1-2 (2012): 5–45. Print.
[7] EL Mencía, and J Fürnkranz. "Efficient Pairwise Multilabel Classification for Large-Scale Problems in the Legal Domain." Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II Springer-Verlag, 2008.
[8] Spyromitros, E., G. Tsoumakas, and I. Vlahavas. "An Empirical Study of Lazy Multilabel Classification Algorithms." Springer, Berlin, Heidelberg (2008).
[9] Vens, C., et al. "Decision trees for hierarchical multi-label classification." Machine Learning 73.2(2008):185-214.
[10] Tsoumakas, et al. "Random k-Labelsets for Multilabel Classification." IEEE Transactions on Knowledge and Data Engineering (2011).
[11] Fürnkranz, J., Hüllermeier, E., Loza Mencía, E., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. Machine Learning, 73(2), 133–153.
[12] Zhang, M. L., and Z. H. Zhou. "A k-nearest neighbor based algorithm for multi-label classification." IEEE International Conference on Granular Computing IEEE, 2005.
[13] Data from https://www.kaggle.com/shivanandmn/multilabel-classification-dataset?select=train.csv.
[14] Jongejan, B., and H. Dalianis. "Automatic training of lemmatization rules that handle
morphological changes in pre-, in- and suffixes alike.” Joint Conference of the Meeting of the Acl & the International Joint Conference on Natural Language Processing of the Afnlp: Volume Association for Computational Linguistics, 2009.

[15] Aizawa, A. An information-theoretic perspective of tf-idf measures. Inf. Process. Manag. 39, 45–65 (2003).

[16] N Green, et al. "WebBANC: Building Semantically-Rich Annotated Corpora from Web User Annotations of Minority Languages." 2009.

[17] T Müller, et al. "Joint Lemmatization and Morphological Tagging with Lemming." Conference on Empirical Methods in Natural Language Processing 2015.

[18] Bergmanis, T., and S. Goldwater. "Context Sensitive Neural Lemmatization with Lematus." Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) 2018.