GTAGLIB: a library to generate tags and tag clouds

Guilherme C. Mattos
Programa de Engenharia de Sistemas e Computação/COPPE
Federal University of Rio de Janeiro
Rio de Janeiro, Brazil
mattosgc@cos.ufrj.br

***Abstract*—Inside and outside of the digital world, tags are an important way to describe or categorize documents and sets of documents, allowing humans and automated systems to more quickly and easily identify and understand their nature and their contents. Meanwhile, tag clouds, or word clouds, are a way to present tags in a graphical format where special attention is given to the relevance that the tags have. That is done to allow human users to identify which are the terms that better represent the data.**

In this work, the GTAGLIB, a library to generate tags and tags clouds for documents and sets of documents, is presented. That library implements two algorithms present in the literature that aim to produce tags and tag clouds to summarize and differentiate documents. Finally, this work also shows how the library performs on a small set of texts and provides a brief discussion about the results.

***Index Terms*—tag, tag cloud, tag recommendation, library, python**

I. Introduction

From physical archives to digital repositories of information, tags have been widely used to index data and facilitate access to information. In Archival Sciences, for example, the NOBRADE (*Norma Brasileira de Descrição Arquivística*, or Brazilian Archival Description Standard) has a field dedicated to that purpose [1], while on the internet, multiple social networks, like Twitter[1] and Instagram[2], heavily rely on the use of tags (or "hashtags") to provide information discovery capabilities to their users.

However, while social networks and many other types of systems usually rely on users to add keywords to their own content, tags assigned by users might not be reliable [2] or the amount of content to be tagged might make that task unfeasible to humans. For those reasons, automatic tag generation, also known as *tag recommendation* in the field of Information Retrieval, is of particular relevance to a multitude of systems and has been a target of research for decades. In [3], for example, works covering several tag recommendation methods are presented, as well as a taxonomy to classify them.

In addition to the tags themselves, another important aspect of text tagging is the way that information is displayed for human use. On the aforementioned Twitter, for example, trending tags are always present, as a list, on the top right area of the pages (on the desktop version of its website) and there is also a page dedicated to that same purpose. Blogs, on the other hand, frequently have tags listed at the end or at the beginning of posts. However, there are other ways to present tags to users and one such way is through tag clouds, that are visual representations where tags have their visual attributes and placement decided based on relevance, where relevance can be determined by features like popularity (frequency) and recency [4], [5].

Given the utility of automatic tag generation systems and tag clouds, in this paper, the GTAGLIB, a library to generate tags and tag clouds from documents, based on the methods of [6] and [7], is presented. In section II, the details about the methods that the library is based upon are covered, while Section III talks about the library itself and the implementation decisions behind it. Section IV describes the evaluation approach taken and Section V shows the results of that approach. This paper is then concluded in Section VI, with possible future works being mentioned in that same section.

II. Differential and Set Summary Tags and Tag Clouds

Introduced by [6] and [7], set summary and differential tag clouds are visual representations of tags that, as their name imply, seek to summarize a set of documents and differentiate individual documents (with respect to the set summary tags), respectively. In both works, a formal model for tag cloud construction is given and, despite not being the focus of the present text, it is important to quickly introduce the basic concepts brought by it.

The first is "semantic field", that comes from [8] and, in the words of [6], "is a set of abstract concepts that, somehow, can be applied to an object aiming to build some understanding of it" (an object, in this case, can be a document). Directly tied to it, the second is "tag field", that is a set of tags that represent those concepts. The third is "abstract tag cloud", which is basically the combination of tags (from the tag field) with features for visual representation (like text color and size), where those features may reflect the relevance of their respective tags.

Based on those fundamental definitions, the authors introduced the abstract set summary tag cloud and the abstract differential tag cloud. The first assumes the existence of a semantic field where the object is the union of all documents, and a summarization function that determines which tags (with their attributes) from that semantic field better describe the

[1]Twitter: https://twitter.com/.
[2]Instagram: https://www.instagram.com/.

document set. Then, the tag cloud is built from those selected tags.

In the differential tag cloud, on the other hand, the focus is document individuality. For this reason, it assumes the existence of a differentiation function that will determine if a member of the abstract tag cloud is enough to differentiate the document from the entire set. That is done by not considering tags that are present in the set summary tag cloud.

With those basic definitions presented, [7] introduces two concrete methods, or algorithms, to generate set summary and differential tag clouds. The steps of those algorithms are presented in the next subsections in order to clarify the basic behaviors that the GTAGLIB aims to implement.

A. Method 1

In this method, the documents are first preprocessed, being tokenized, stemmed (using the Porter Stemmer [9]) and having stop words removed. Additionally, only words identified as nouns (by a preceding POS-tagging[3] step) are kept. After that, bigrams are generated and the data is then represented under a TF-IDF[4] (term frequency, inverse document frequency) model and the following steps are performed.

1) Build a semantic field for each document, choosing the 40 most relevant terms (based on TF-IDF) for a given document as tags.
2) Use the semantic field to create an abstract tag cloud for each of those documents, where the attributes should be based on term relevance.
3) Generate the differential tag cloud from the abstract tag cloud (in other words, both are the same).
4) Create a term-document frequency matrix.
5) Apply Latent Semantic Analysis [10], decomposing and reconstructing the term-document frequency matrix. In this step, it is important to note that [7] does not explain exactly what is done to the decomposed matrices before reconstruction.
6) Apply the Pearson Correlation Coefficient on the reconstructed matrix in order to get a table with the correlation between the terms.
7) Pick a root term (possibly chosen by the user) and build a list with the most positively correlated terms to that root until 40 are selected (additional bigrams related to the root can also be selected).
8) Build the set summary tag cloud from the terms from the previous item.

B. Method 2

In method 2, the same preprocessing steps from method 1 are applied and the documents are also represented under a TF-IDF model. One of the main differences is the expansion of the semantic field with synonyms of the terms already there. The steps of the algorithm are as follows.

1) Build a semantic field for each document, choosing the 40 most relevant terms (based on TF-IDF) for a given document as tags.
2) Expand the semantic fields with related concepts. In this case, "related concepts" mean synonyms of terms in the semantic field, obtained from *synsets* from the WordNet [11].
3) Use the semantic field to create an abstract tag cloud for each of those documents, where the attributes should be based on term relevance. In this part, [7] does not cover what the relevance of the additional terms should be.
4) Sort, in descending order, the union of terms from all abstract tag clouds by the number of semantic fields where they appear.
5) Build the set summary tag cloud from the first 40 terms obtained from the previous step.
6) For each document, build the differential tag cloud from the terms in its abstract tag cloud that are not present in the set summary tag cloud. Color synonyms added from *synsets* with a different color.

In both methods, the stemming is reversed before the creation of a tag cloud, but no information is provided on how to deal with situations where different words produce the same stem.

When it comes to specifics of the tag cloud presentation, [6] and [7], establish no definition, regardless of method. In any case, experiments and discussions about that matter can be found in [5].

III. GTAGLIB

The GTAGLIB (where the first "g" in its name stands for "generation") is a Python library (currently in version 0.1) that aims to provide the community with yet another tool to help with the task of tag and tag cloud generation. In its current form, it attempts to implement the methods proposed by [6] and [7], and is publicly available on GitHub[5].

When compared to what is specified in the original algorithms of the differential and set summary tag clouds, the GTAGLIB also seeks to provide more flexibility in parameter selection and, when it comes to this matter, its internal preprocessing steps allow the use of semantic fields of different sizes, two stemming techniques (Porter [9] and Snowball/Porter2 [12]), one lemmatizer (WordNet[6], instead of using a stemmer), the ability to use only a term-document frequency matrix across the algorithms and the possibility of turning off the generation of bigrams. By default, and considering that the library, in its current version, aims to support only the English language, it also converts all word characters to ASCII characters. However, given the fact that it can be undesirable in some circumstances (like in cases where the text is multilingual), this can also be disabled.

The implementation of the algorithms also has slight differences when compared to what is originally proposed. They are as follows.

[3]POS-Tagging: https://en.wikipedia.org/wiki/Part-of-speech_tagging.
[4]TF-IDF: https://en.wikipedia.org/wiki/Tf-idf/.
[5]GTAGLIB: https://github.com/GuilhermeCaeiro/gtaglib/.
[6]WordNet: https://www.nltk.org/_modules/nltk/stem/wordnet.html.

A. Method 1

The implementation of the method 1 is mostly similar to what is specified in [7], possibly differing mainly in three aspects, where the first is the paper of Latent Semantic Analysis (LSA). It is claimed to be used, but no information about what exactly is done between the decomposition and the reconstruction of the term-document matrix is provided. However, in the literature review section of the same work, while talking about that technique, it focuses on dimensionality reduction. For that reason, in GTAGLIB (version 0.1), it was decided to apply that procedure during the LSA step, maintaining only the n most important concepts in the reconstructed term-document matrix, where, by default, n is the the size of the semantic field in the original algorithm (40). Additionally, considering that [10] states that the number of dimensions to be kept for the LSA to be effective is an "empirical issue", the desired value of n can be passed to the library as a parameter.

The second possible difference is related to what is done if the term passed as root to generate the set summary tag cloud is not present in the model (if it is not a noun present in the set of documents). The description of the original algorithm does not cover that case, while GTAGLIB's implementation simply returns an empty list of tags or does not generate a tag cloud.

The last difference resides in the fact that, during the generation of the set summary tags, no additional bigrams are added after the 40 terms related to the root are selected.

B. Method 2

In method 2, the algorithm provided by [7] also has steps that are unclear and, for that reason, GTAGLIB's implementation has some differences, one of them being the way the expansion of the semantic fields occur. While the original algorithm provides no information on how many or what type of synonyms should be selected, the GTAGLIB allows users to select an upper limit on how many synonyms should be added per each word in the original semantic field. Additionally, the synonyms used are limited to *synset* names that are nouns. That is done to reduce the inclusion of unwanted words (like verbs). If the number of available synonyms is greater than an upper limit u stipulated, the algorithms returns u random synonyms from those available.

C. Tag Cloud Generation

The tag cloud creation relies on two aspects: word relevance and styling. The word relevance of the tags passed to the tag cloud depends on which algorithm is being run and what are the parameters being used. For abstract and differential tag clouds, the library uses TF-IDF scores or term-document frequencies with respect to the document associated with the tag cloud. The deciding factor is if the use of TF-IDF is enable or not. Regarding set summary tag clouds, the type of the score depends on the same factor, but the final score is the sum of the TF-IDF scores or term-document frequencies for a given word across all documents.

As for tag clouds created using the method 2, the additional tags will always receive the score of the least relevant tag originally present in the semantic field. This is done in order to reduce the impact of any eventual "noise" added by the inclusion of synonyms, because, in many cases, they may have little to no real relation with the original documents. In the current version of the library, synonyms in the differential tag cloud are not colored differently.

Finally, regarding styling, that is done through the use of the library WordCloud[7], that is a mature, resourceful Python library for that purpose. That library can also be used in two ways. The first is by having GTAGLIB creating a default WordCloud object and using it to generate the tag clouds, while the other is by passing an already set up WordCloud object to GTAGLIB and having it being used instead. That approach was taken in order to give users nearly full control of the tag cloud styling.

IV. Evaluation Approach

In order to evaluate the library presented in this work, the dataset fao30[8], that was introduce by [13], was utilized. It is a small corpus composed of 30 documents related to agricultural themes that was tagged by a team of professional annotators. When considered all documents, after the removal of words with only one character, it has 129142 words and, on average, 4304.73 per document. The total number of tags is 997, having an average of 33.23 per document.

Before providing the documents to GTAGLIB, the only preprocessing steps taken were the removal punctuation, numbers and words with length of one character (as mentioned previously). Stopword removal and other possibly relevant steps were left to be treated by the internal preprocessing capabilities that the library has. The parameters used with both methods can be seen in Table I and, except for the option to convert characters to ASCII, they were chosen to match specifications present in [7].

TABLE I
Main Testing Parameters

Parameter	Value
Semantic field size	40
Stemmer	Porter
Generate bigrams	Yes
Use only nouns	Yes
Use TF-IDF	Yes
Characters to ASCII	Yes

As for the information that was sought to be evaluated, it was the abstract and the differential tags generated by both algorithms proposed by [7]. Abstract tags, despite not being a main product of the second algorithm, they can be retrieved in GTAGLIB's implementation and, for this reason, were considered for testing in both methods. Also for the particular case of the algorithm 2, it was tested with and without the

[7]WordCloud: https://github.com/amueller/word_cloud.

[8]Dataset fao30: https://github.com/LIAAD/KeywordExtractor-Datasets.

expansion of the semantic fields with synonyms and, in the case where those additional words were included, the impact of the inclusion of 3, 5, 7 and 10 synonyms per tag in the semantic field of each document was verified.

The metrics employed in the tests were the precision, the recall and the F1 scores, that, according to [14] are defined defined by Equations (1), (2) and (3), respectively. In those equations *tp* (true positives) is the number of tags produced that are present among the tags that are expected, *fp* (false positives) is the number of tags produced that are not present among the tags that are expected and *fn* (false negatives) is the number of tags that are expected, but were not generated. As for their meaning, [14] defines precision as "the fraction of relevant instances among the retrieved instances" (where "relevant instances" are the expected tags) and recall as "the fraction of relevant instances that were retrieved". The F1 score, on the other hand, is a metric that tries to combine both metrics in a balanced way (for example, to achieve a high F1 score, precision and recall must also be high).

$$precision = \frac{tp}{tp + fp} \tag{1}$$

$$recall = \frac{tp}{tp + fn} \tag{2}$$

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{3}$$

When it comes to the time taken to generate the tags from the documents, it was also measured, making it important to mention the testing environment: a PC with an Intel Core i7-4720HQ CPU, 16GB of DDR3 1600 RAM and a SSD Crucial BX500 of 480GB. The operating system was Windows 10 Home 21H1, but the tests were run on Windows Subsystem for Linux (WSL 1), that ran Ubuntu 18.04.4 LTS.

Lastly, regarding tag cloud generation, given the fact that, in this work, it was relegated to another library, it is something difficult to properly asses. For this reason, this text will only comment a few examples of what the GTAGLIB (alongside the WordCloud library) is capable of generating. After that, at the end of the results section, a known issue found with GTAGLIB's implementation will also be briefly commented.

V. Results

Following the approach established previously, in this section, the first results to be presented are for method 1. In that method, abstract and differential tags are the same and achieve, across the 30 documents used for testing, an average precision of 0.095, a recall of 0.119 and a F1 score of 0.104, meaning that, out of the expected tags, only a very small number are being successfully generated by GTAGLIB.

When it comes to method 2, the results for the abstract tags are displayed in Figure 1. In that figure, it is possible to observe that, with the addition of synonyms to the semantic fields, average precision and, consequently, average F1 score fall. Recall, on the other hand, shows a marginal gain with the addition of up to 3 synonyms per tag, and stays the same with further additions. That difference in behavior when compared to precision is caused by the fact that precision is negatively impacted by the additional, frequently irrelevant (noise), synonyms, while recall only takes into consideration what was generated correctly.

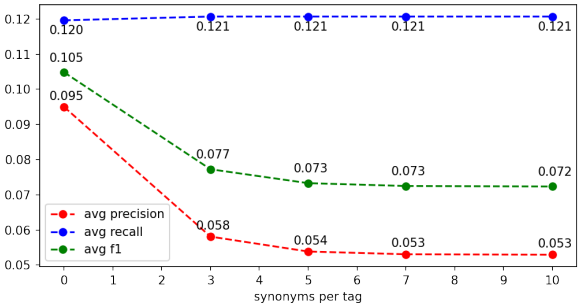

Fig. 1. Results for abstract tags using method 2.

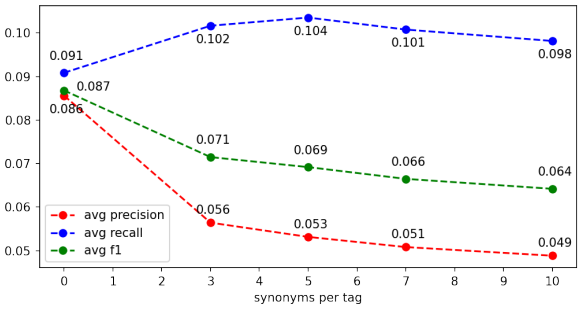

Fig. 2. Results for differential tags using method 2.

As for the differential tags, Figure 2 shows a slightly different scenario, with recall starting at a lower value, growing with the addition of up to 5 synonyms per tag, and then falling with more additions. That's caused by the removal of tags that are among the set summary tags. In other words, with zero synonyms, some relevant, expected tags end up as set summary tags and are not considered as differential tags, impacting all metrics negatively. However, with the inclusion of synonyms, some of these additional words might end up among the set summary tags as well, while some relevant, expected abstract tags go to the set of differential tags, impacting recall positively. Later, when recall falls, that might still be an effect of what is being included in the set summary tags and what is not. It "might" because the selection of the synonyms to be included in the semantic field is a random procedure, making the additional tags vary if the number of additions is smaller than the number of available synonyms.

Now, focusing on the scores obtained by both methods, and disregarding the differences in evaluation methodology, the precision, recall and F1 scores are less than half the numbers presented by [13] for those same metrics on a much larger dataset related related to the one used in the present work.

However, their method generated tags based on a domain-specific controlled vocabulary and not from words extracted directly from the text.

Fig. 3. A set summary tag cloud from method 1.

Moving on to the evaluation of tag cloud generation, in Figure 3, the set summary tag cloud for the entire corpus was created using the method 1 and the word ”agriculture” as root. In that tag cloud, it is possible to observe that, using that word as root, several tags have a clear logical connection with ”agriculture”, like ”rice production”, ”food production” and ”agriculture policy”. It is also noticeable that the most relevant terms are represented with bigger font sizes, while the less relevant are smaller. However, it is important to note that it was not evaluated if the tags presented summarize well the set of documents under the perspective of the concept of ”agriculture”.

When it comes to tag clouds produced using method 2, the situation is different. Despite having the most relevant tags highlighted with bigger fonts, the tag clouds are prone to contain noise, that are terms unrelated to the document or set of documents that is being represented. In Figure 4, for example, the differential tag cloud has the word ”capacitance”, that probably was included as a synonym of ”capacity”. However, ”capacitance” has no relation with the document, making its presence in the tag cloud awkward.

Next, covering the processing times, Table II lists the time taken to generate tags (without tag clouds) using method 2 (values obtained while producing the previous results). The average time in this case is 50.79 seconds, which represents, approximately, 1.69 seconds per document. From Table II, it is noticeable that, with the small corpus employed in this work, the increase in the number of synonym additions did not result in a relevant increase in processing time. It is also important to mention that, within that time, all three types of tag clouds are generated.

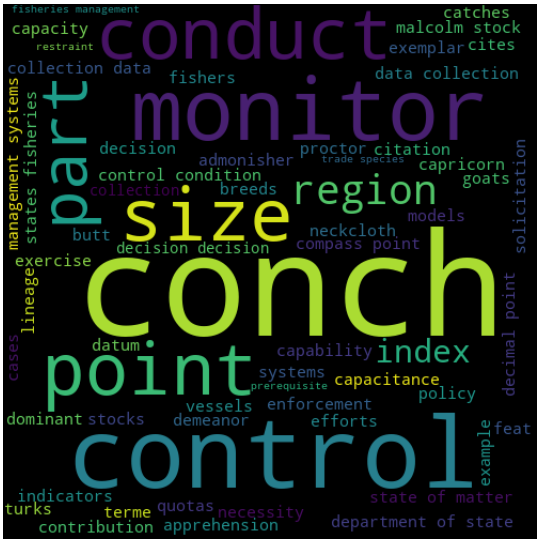

Fig. 4. A differential tag cloud from method 2.

As for method 1, if considered only tag generation, it took 50.13 seconds to produce abstract and differential tags, without set summary tags or tag clouds. On the other hand, if set summary tags and tag cloud are generated, that time grows substantially because of the LSA step and the calculation of correlations. In one example, it took 224.03 seconds to produce all 3 types of tags and tag clouds. In method 1, if tag clouds are produced, the time also increases, but to a smaller extent, taking only 70.65 seconds.

TABLE II
PROCESSING TIMES FOR METHOD 2

Synonyms/Tag	Time (s)
0	50.18
3	50.37
5	51.45
7	50.98
10	50.98

Finally, covering a known issues with the library, during the tests, it was found that, by applying the POS-tagger after tokenization and stop word removal, but before stemming, it is possible for the POS-tagger to let words that are not nouns to be recognized as nouns. In [7], the POS-tagging is executed last, after stemming. However, as of the moment of the writing of this text, it is not possible to assess if that would solve the issue.

VI. CONCLUSIONS

In this work, a portion of the capabilities of the GTAGLIB were shown and the results demonstrated that it can achieve some success within the proposals of the algorithms it implements. However, it is unfortunate that those same results were not good, and considerably low if compared to what is presented in [13], although it is important to note that the

methods implemented in the library, despite being contemporary to that other work, do not rely on a controlled vocabulary. Another relevant aspect that can impact the results, but was not evaluated in this work, is the fact that some tags that are used as reference are not explicitly present in the text, like "disease surveillance" for the first document of the corpus.

Additionally, given the known issue presented in the previous section, in the current version, potential users might want to completely preprocess the documents before providing then to the library.

Considering that the library is in its early version (0.1) and that the tests were mostly limited to parameters stipulated by the original algorithms, when it comes to future works, they might include fixing bugs, testing the library with more parameter variations, testing it on a bigger corpus (specially one that is more commonly used for benchmarking, so that the results could be compared to other works), improving the performance of the library, implementing other tag generation techniques and bundling the library into add-ons for data mining tools, like KNIME[9] and Orange[10].

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

REFERENCES

[1] Conarq, *Norma Brasileira de Descrição Arquivística*. Rio de Janeiro: Arquivo Nacional, 2006. [Online]. Available: http://www.siga.arquivonacional.gov.br/images/publicacoes/nobrade.pdf

[2] G. Koutrika, F. A. Effendi, Z. Gyöngyi, P. Heymann, and H. Garcia-Molina, "Combating spam in tagging systems," *ACM Transactions on the Web*, vol. 2, no. 4, pp. 1–34, Oct. 2008. [Online]. Available: https://doi.org/10.1145/1409220.1409225

[3] F. M. Belém, J. M. Almeida, and M. A. Gonçalves, "A survey on tag recommendation methods," *Journal of the Association for Information Science and Technology*, vol. 68, no. 4, pp. 830–844, Dec. 2016. [Online]. Available: https://doi.org/10.1002/asi.23736

[4] M. J. Halvey and M. T. Keane, "An assessment of tag presentation techniques," in *Proceedings of the 16th international conference on World Wide Web - WWW '07*. ACM Press, 2007. [Online]. Available: https://doi.org/10.1145/1242572.1242826

[5] S. Bateman, C. Gutwin, and M. Nacenta, "Seeing things in the clouds," in *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia - HT '08*. ACM Press, 2008. [Online]. Available: https://doi.org/10.1145/1379092.1379130

[6] G. Xexeo, F. Morgado, and P. Fiuza, "Differential tag clouds: Highlighting particular features in documents," in *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, vol. 3, 2009, pp. 129–132.

[7] F. F. Morgado, "Representação de documentos através de nuvens de termos," Master's thesis, Federal University of Rio de Janeiro, Brazil, 2010. [Online]. Available: https://www.cos.ufrj.br/index.php/pt-BR/publicacoes-pesquisa/details/15/2172

[8] I. Marinchev, "Practical semantic web – tagging and tag clouds," *Journal Cybernetics and Information Technologies*, vol. 6, no. 3, pp. 33–39, 2006.

[9] M. F. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems*, vol. 14, no. 3, pp. 130–137, Mar. 1980. [Online]. Available: https://doi.org/10.1108/eb046814

[10] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse Processes*, vol. 25, no. 2-3, pp. 259–284, Jan. 1998. [Online]. Available: https://doi.org/10.1080/01638539809545028

[11] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995. [Online]. Available: https://doi.org/10.1145/219717.219748

[12] M. Porter. The english (porter2) stemming algorithm. Accessed: 2022-06-23. [Online]. Available: http://snowball.tartarus.org/algorithms/english/stemmer.html

[13] O. Medelyan and I. H. Witten, "Domain-independent automatic keyphrase indexing with small training sets," *Journal of the American Society for Information Science and Technology*, vol. 59, no. 7, pp. 1026–1040, 2008. [Online]. Available: https://doi.org/10.1002/asi.20790

[14] I. Wikimedia Foundation. Precision and recall. Accessed: 2022-06-22. [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall

[9]KNIME: https://www.knime.com/.

[10]Orange: https://orangedatamining.com/.