Published in final edited form as:
Health Informatics J. 2020 March ; 26(1): 21–33. doi:10.1177/1460458219869490.

Recognizing software names in biomedical literature using machine learning

Qiang Wei, **Yaoyun Zhang**, **Muhammad Amith**
The University of Texas Health Science Center at Houston, USA

Rebecca Lin,
Johns Hopkins University, USA

Jenay Lapeyrolerie
Baylor University, USA

Cui Tao, **Hua Xu**
The University of Texas Health Science Center at Houston, USA

Abstract

Software tools now are essential to research and applications in the biomedical domain. However, existing software repositories are mainly built using manual curation, which is time-consuming and unscalable. This study took the initiative to manually annotate software names in 1,120 MEDLINE abstracts and titles and used this corpus to develop and evaluate machine learning-based named entity recognition systems for biomedical software. Specifically, two strategies were proposed for feature engineering: (1) domain knowledge features and (2) unsupervised word representation features of clustered and binarized word embeddings. Our best system achieved an F-measure of 91.79% for recognizing software from titles and an F-measure of 86.35% for recognizing software from both titles and abstracts using inexact matching criteria. We then created a biomedical software catalog with 19,557 entries using the developed system. This study demonstrates the feasibility of using natural language processing methods to automatically build a high-quality software index from biomedical literature.

Keywords

biomedical literature; biomedical software; biomedical software index; named entity recognition; natural language processing

Corresponding author: Hua Xu, School of Biomedical Informatics, The University of Texas Health Science Center at Houston, Houston, TX 77030, USA. Hua.Xu@uth.tmc.edu.

Declaration of conflicting interests
The author(s) declared the following potential conflicts of interest with respect to the research, authorship, and/or publication of this article: Dr Xu and The University of Texas Health Science Center at Houston have research-related financial interests in Melax Technologies, Inc.

Introduction

Biomedical research is a data-intensive field. With the growth of advanced high-throughput technologies, large amounts of biomedical data are being generated at an exponential rate. Thus, software tools, which are used to manage, normalize and analyze data, become essential for biomedical research and applications. As the ease of sharing data and therefore reuse of data has amplified tremendously, there is immediate need to be able to utilize the appropriate means for data analyses. This coupling of the data and the software tools is also necessary for reproducibility of results from existing datasets, an essential step to build further on interesting hypothesis and moving the field forward. To facilitate the reproducibility of biomedical research and applications, software is an indispensable part in the biomedical digital ecosystem for making biological objects Findable, Accessible, Interoperable and Reusable (FAIR). Thus, it is critical to build a comprehensive, high-quality software index for the biomedical domain.

Biomedical researchers have recognized the importance of software in the digital ecosystem, and existing efforts have focused on building software repositories. For example, Bioconductor is a collection of open-source tools for analysis of high-throughput genomic data,[1] BioJS provides tools for biological data visualization,[2] and BioCatalogue is a universal catalog of web services,[3] to name a few. Moreover, to maintain a foundation of sustainable updates and quality assurance of software repositories in the long term,[4] community efforts have been spent on aggregating multiple existing software collections into more unified, well-organized large-scale repositories. For example, OMICtools[5] collect web-accessible tools related to analysis of omics data. ELIXIR[4] is the European infrastructure for biological information charged with developing various resource and software repositories. A recent repository of biomedical tools and resources, Aztec,[6] hosts software collected from publications, other indexes, user submissions and funding sources.

Given the rapid growth of the number of new software tools in the biomedical domain, the biomedical communities have recognized that manual curation and voluntary user registration is inefficient for maintaining an up-to-date software collection. One emerging direction is extracting software information from published articles automatically using natural language processing (NLP)-based methods. As an assistance to manual curation, Ozyurt et al.[7] developed a tool to extract URLs from papers to screen resource candidates for the Neuroscience Information Framework (NIF) Registry. Wang et al.[6] also identified public software repository URLs (e.g. GitHub) and used heuristic rules to extract software names from publication titles if no URL is available. However, no systematic evaluation and software recognition system was reported in their work so far.[6] Duck et al. used dictionaries and heuristic rules to extract biomedical databases and software names from the full text of literature. In addition, a machine learning-based classifier was developed based on the matched rules to further filter false-positive errors.[8] However, modest performance was obtained by their approach, with F-measures lower than 70%.[8] Besides, the aim of their research was to investigate the usage of biomedical databases and software across different studies by checking their occurrences in literature, rather than building a software index for the biomedical domain.[8]

In fact, the task of identifying software names from biomedical literature is not trivial, with multiple variations and ambiguous expressions of software names.[8] Besides, the auxiliary software information such as operation systems and backend supporting computational tools are also important for the configuration, implementation and practical application of biomedical software. Aimed at building an automatic software recognition system for practical use, in this study, we created a relatively large corpus with annotated software names and used it to develop a high-performance machine learning–based software recognition system. Multiple groups of features were applied, including common NLP features, domain knowledge features of dictionaries, software name patterns and locations and more advanced features of unsupervised word representations generated by deep-learning-based methods from a large unlabeled biomedical literature corpus. We also conducted a detailed analysis of the system's errors and classified ambiguities between software names and other biomedical entities, enabling future improvement. Furthermore, we applied the automatic software recognition system on MEDLINE titles and generated a catalog of biomedical software with 19,557 entries.

Materials and method

System overview

Figure 1 illustrates the study design for automated software extraction from biomedical literature. First of all, a software corpus was built from literature in PubMed by manual annotation, from which the software recognition system was developed. The software corpus included 1120 abstracts and titles, in which software names are labeled. The corpus was divided into training set and test set for developing software recognition system and evaluation. The details of the software corpus were described in section "Dataset and annotation". Specifically, the software recognition problem was treated as a typical named entity recognition (NER) task. First of all, a preprocessing step of sentence segmentation and tokenization was performed to obtain all tokens of each abstract and title. Then, various types of features were extracted for each token (details are in section "Software entity recognition"). The sentences were then represented as sequential labels of B, I and O, and each token of a sentence could be one of them (details are in section "Software entity recognition"). The automated software recognition system included the machine-learning model and rule-based post processing: (1) based on these features and sequential labels of sentences in the training set, the machine-learning algorithm of conditional random fields was used to generate the machine-learning model and (2) then some rules and patterns were applied to further reduce errors generated by the model. We evaluated our system on the test set. Finally, we use our system to generate a high software index on a large collection of MEDLINE titles.

Dataset and annotation

The Medline collection of research articles were retrieved from PubMed with keywords of "software," "tool," "toolkit" and "system" in the fields of title or abstract. The top 1120 abstracts with titles from the result list were used for annotation. A guideline was developed to annotate software name manually. The longest noun phrase representing a software name was labeled, which included the software name, its modifiers, abbreviation and version number. The software annotated in this study consists of both the core software as the main topic of the article and auxiliary software related to its configuration environment. The mentions of databases and general programming languages were excluded. The toolkit CLAMP was used for annotation in this study.[9] Figure 2 illustrates one example of annotated biomedical literature for software names.

In total, the software corpus contained 1120 articles with annotated titles and abstracts. For systematic evaluation of the generated software recognition system, two-thirds of the dataset were randomly selected as the training set and the remaining one-third were used as the test set.

Software entity recognition

The recognition of software name is a typical NER problem in the area of NLP, which could be viewed as a sequential labeling problem.[10] Each token in the sentence can be represented as one of "B,""I" and "O" labels. "B" means the beginning of a software name, "I" represents the other words in the software name and "O" represents words that are not inside software names. In this way, the object of this task is converted into finding the hidden labels for all tokens inside a sentence simultaneously (Figure 3).[10]

In this study, we used multiple types of features to recognize software name. In order to better examine their effects on recognition of software name, they are classified into three groups. The first group includes the most common NER features,[11] such as word shape feature, n-gram feature, prefix and suffix feature, sentence feature and part-of-speech feature, which were used to build a baseline system. Moreover, we also investigated a group of features derived from domain knowledge, as well as a group of word representation features generated by unsupervised learning from biomedical literature text. All the features are described in detail in the following. Table 1 showed an example of all extracted features for the word WoMMBAT from Figure 2. Each type of feature had some specific features, which were represented as "NAME=[VALUE]".

Feature extraction

Basic features

Word shape feature: stemmed tokens and shapes of tokens. For example, the feature of token "Medicine" is "medicin" and "Aaaaaaaa".

N-gram feature: bi-grams and tri-grams representing the context of the token with a window size of [−3,3].

Sentence feature: sentence attributes such as the length of the sentence, the POS-tag of the beginning word and the ending punctuation.

Prefix–suffix feature: the first and last m (m = 1, 2, 3) characters of a token.

Domain knowledge features

Section feature: whether the token is inside the title or the abstract.

Dictionary feature: a dictionary of software names collected from the AZTEC[6] and SourceForge[12] was used to match candidate software names, which included 52,496 software names. Tokens and their bi-gram within windows size of [−2, 2] of the token will be examined whether they are in the dictionary. If they are in the dictionary, the feature was set to its semantic type in the dictionary, otherwise it was set to "TK".

Orthographic feature: software names are typically formed by mixtures of characters with uppercases, lowercases, digits and some specific punctuations (e.g., "-"). Regular expression rules were used to capture the orthographic characteristics of tokens for candidate software names and generate multiple features. If the token met one orthographic characteristics, the feature of the token on this characteristics would be set to "TRUE", otherwise it will be set to "FALSE".

Unsupervised word representation features

Word embedding feature: Word representation features were generated from a corpus of unlabeled abstracts from PubMed. Specifically, we used word embeddings that produced a distributional word representation for each word in an unlabeled corpus as a real-valued vector using neural networks.[13–15] First, we clustered tokens into 1000 groups based on the similarity among their real-valued word embeddings and used the cluster labels as features (word embedding feature, clustering). Moreover, we used the binarized word embedding feature proposed in 2014 by Guo et al.[16] (word embedding feature, discretization). The intuition of the binarized embedding feature is to discretize the original real-valued matrix of word embeddings[15] and omit the insignificant dimensions. Thus, the less frequent terms are generalized together with other syntactically/semantically relevant terms of higher frequency.

Machine-learning method

The state-of-the-art machine learning–based algorithm of Conditional Random Fields (CRF) is used for NER. Particularly, CRFsuite (http://www.chokkan.org/software/crfsuite/) was used as the implementation of CRF. A java script was used to generate features mentioned above for each word. We trained the CRF models using the training set. And by using different combination of features, we obtained multiple models, which were used for evaluating the effects of these features on the task. Then, we used these models to recognize software names in the test set.

Rule-based post-processing

Some rules were applied to the output of the machine learning-based software recognition model to fix obvious errors and further enhance the recognition performance:

1. Common patterns of software names were used to identify software entities, especially in titles. Basically, common key words and patterns were used to locate software names, as summarized in Table 2. Based on these patterns, regular expressions were used to extract software names. For example, "GOAL" is a software name in the title of "*GOAL*: a software tool for assessing biological significance of genes groups."

2. Although a software name can occur multiple times in an abstract, some of the mentions may be misrecognized due to the rare surrounding context. To address this problem, we conducted a dictionary lookup by exact match in the abstract, using the recognized entities as a lexicon. If there was a string that matched the recognized entity, then the string was labeled as a new entity.

3. Additional dictionaries of programming languages were used to filter out false-positive software names.

Experiments and evaluation

In this study, we started with a baseline system that implemented common features including bag-of-word, word shape information, morphological information and part of speech (POS). We also used a software name recognition system bioNerDS as baseline.[17] The bioNerDS aimed to recognize database and software mentions in literature, which had the same goal as us. The bioNerDS was a rule-based system that first used dictionary lookup to recognize software names and then used some patterns to recognize software names not in dictionary. The system was developed on a set of 30 full-text articles and tested on two sets of 25 and 5 full-text articles, respectively, which had a lenient F1 score of 0.63 and 0.91 on two test sets. Then, we evaluated the effects of domain knowledge-based features and unsupervised word representation features. Features in different groups were added to the feature set incrementally to examine their impact on the performance of the system. Finally, in the post-processing step, rules were applied to the prediction from the machine-learning model.

The performance of micro-averaged precision, recall and F-measure evaluated by exact and inexact match was reported. In the exact match, the predicted entity has the same offset as that in the Gold standard set; while in the inexact match setting, the prediction and the Gold standard annotation have at least an overlap in their offsets. Besides, as mentioned in the "Materials and method" section, both biomedical software as the core target of this study and software related to its configuration environment were recognized by our automatic system. We examined the performance of the systems for recognizing all the annotated software, as well as only recognizing biomedical software, that is, the core target, separately. Since most of the biomedical software names have their first occurrence in the titles, the performance of our systems on software recognition from titles was also reported.

We further applied the software recognition system with the optimal performance to a large collection of MEDLINE titles to generate a high-quality software index. In total, 117,546 MEDLINE titles were retrieved from PubMed using the software keywords as queries.

Results

Table 3 shows the performance of the software recognition system on the test set. The baseline system bioNerDS had a lower F-measure of 23.69% for exact match and 49.05% for inexact match. The performance of inexact match was much higher than that of exact match may be because their definition of software name was not exactly the same as ours. The model with baseline feature yielded an F-measure of 68.57% for exact match and 77.99% for inexact match, with very low recalls (59.38% for exact match and 67.54% for inexact match). Surprisingly, using the dictionary of software names collected from existing repositories did not have much influence on the performance, whereas the orthographic and section features increased the performance consistently. Since software names in the dictionary were mainly collected from general domain and the size of the dictionary was not very large, only a few software names in our corpus were covered by it. Besides, software names are usually composed of all upper letters or some words with the initial letter capitalized and usually occur at the beginning of titles. Therefore, for software names not present in the dictionary, the orthographic and section features are more helpful. By adding discrete word representation features, the recall (exact: 60.32% vs 63.82%; inexact: 68.92% vs 73.31%) and F-measure (exact: 68.93% vs 70.73%; inexact: 78.76% vs 81.24%) were increased significantly, with slight sacrifice of precision (exact: 80.40% vs 79.31%; inexact: 91.86% vs 91.10%). Furthermore, the clustering-based word representation features improved the performance of both the precision and recall. Among all the rules used in the post-processing step, the rule (2) contributed the most to increase the recall (exact: 65.32% vs 72.20%; inexact: 75.36% vs 87.79%). Finally, the post-processing step boosted the recall (exact: 64.59% vs 71.53%; inexact: 73.81% to 87.07%) and achieved the optimal F-measure of 86.35% for inexact match.

The performance of our systems on software recognition from titles is reported in Table 4. Overall, the improvements of performance followed similar trends as in Table 3, by adding different features incrementally. Notably, the optimal performance of recognizing biomedical software from the titles was much higher than recognizing all types of software from the abstracts, with an F-measure of 80.84% for exact match and 91.79% for inexact match, respectively.

Finally, our optimal system extracted 19,557 software names from 18,409 titles of 4636 journals. A prototype version of the generated biomedical software index can be accessed at https://sbmi.uth.edu/ccb/resources/biomedicalSoftware.htm.

Discussion

Biomedical software is one of the critical and fundamental resources for biomedical research and applications. This study created a corpus of software from biomedical literature by manual annotation and built automatic software recognition systems based on it. Our best system achieved an F-measure of 91.79% for recognizing the biomedical software in titles and an F-measure of 86.35% for recognizing both biomedical software and the auxiliary software in its configuration environment in titles and abstract, demonstrating the feasibility of using machine learning-based methods to build high-quality software repositories automatically for the biomedical domain. To the best of our knowledge, this is the first attempt to build practical software recognition systems for the biomedical domain.

In order to further improve our software recognition systems, we manually analyzed the current prediction errors and summarized the major reasons as listed in Table 5: (1) The main causes of false-positive errors are that some biomedical concepts have similar orthographic characteristics or similar surrounding contexts as the software names. For example, the AD in example (a) is the abbreviation of Alzheimer’s disease, and the SMAT80 in example (b) is actually a substitution matrix for protein alignment. Similar to software names, they are also composed of upper letters and digits. Another challenge is caused by the similar context of some expressions with software names (such as in example (c)), which cannot be handled by our current features. In fact, there are multiple types of non-software concepts bearing similar orthographic characteristics or contexts as software names. As illustrated in Figure 4, biological concepts account for 32% and biomedical method names account for 22% of such errors. One possible solution is to employ additional dictionaries of these types of concepts to remove false positives. Besides, the complex syntactic structures of sentences are another cause for false-positive cases. In example (d), the sentence contains the word "algorithm," indicating that related entities are not software names. However, our system failed to recognize the parallel structure among "TSP", "k-TSP", "TST" and "DIRAC" currently. Pattern-based rules or syntactic features need to be added in the future to resolve such type of errors. (2) For false-negative errors, lack of sufficient context (example (e)) and rare patterns of context (examples (f)–(i)) are the two major causes. Despite the fact that some contexts contain words or syntactic structures that can help to indicate the presence of software names, these diverse patterns have relatively low frequencies in the corpus, and as a result, they are poorly modeled by the machine-learning algorithm. Enlarging the training corpus may help to increase the coverage of effective features in the next step.

Our systems had a much higher performance evaluated by inexact match than by exact match (F-measure: 70.94% vs 86.35%). One possible reason could be that our current guideline required annotating the longest noun phrase of software names, which include articles, adjectives and other modifiers. The system may have only labeled the software names without modifiers. For example, in the sentence "… and computer-assisted PredictAD tool.," the noun phrase "computer-assisted PredictAD tool" is annotated as the complete software name. However, only "PredictAD tool" was recognized as the software name by our system, which already contained the key information for retrieving the software from biomedical resources. Therefore, we argue that inexact matching could be reasonable in software recognition and the current system has achieved a performance that would be useful for practical applications. Furthermore, we may further expand the scope of inexact match by using semantic types and synonyms in the hierarchical structures of existing ontologies to catch the most informative part of predictions.

There are several limitations to our work. Currently, we only used a short list of keywords to retrieve literature of biomedical software from PubMed. An expanded keyword set will be used to enrich the corpus in the next step, in order to cover more software types and diverse features for a more robust software recognition system. Besides, additional domain knowledge and patterns-based features can be employed to further enhance the software recognition performance. Amith et al. developed an ontology-driven method to recognize software names.[18] In their work, a corpus of 185 titles and abstracts was used, which was insufficient to build machine learning-based models. The performance of the proposed ontology-driven method (F-measure: 0.53) also did not reach the requirement for practical applications, due to the limited coverage of the employed ontology for the large amount of software names and patterns in biomedical literature. Similarly, the reason why the baseline system bioNerDS had a lower recall (recall: 0.39; F-measure: 0.49) was potential that the bioNerDS was a rule-based system, and the rules developed on training set of limited journals cannot cover patterns in both of their and our test sets. In future, we will develop a system that combines machine learning–based methods and ontology-driven methods for potential improvement of performance. Moreover, we will extend our work to extract additional metadata of software from other structured fields in MEDLINE and linked software portals (e.g. GitHub), such as journal names, author and funding information, official website, functions of software and so on, to build a software repository with comprehensive information for the biomedical research community. We will create a repository of biomedical software that can update software from multiple sources automatically using our software recognition system, to assist researchers with an efficient access to the most updated software resources.

Conclusion

Biomedical software is one of the critical and fundamental resources for biomedical research and applications. This study takes the initiative to create a corpus of software from biomedical literature and build automatic software recognition systems based on it. The promising performance of the systems indicates the feasibility of building high-quality software repositories automatically for the biomedical domain.

Acknowledgements

We thank Dr Jun Xu and Jingqi Wang for their valuable assistance.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship and/or publication of this article: This project was supported by BioCADDIE: Biomedical and healthCAre Data Discovery and Indexing Engine center (NIH 1U24HL126126-01) and the UTHealth Innovation for Cancer Prevention Research Training Program and its Summer Intern Program (Cancer Prevention & Research Institute of Texas grant no. RP160015).

References

1. Gentleman RC, Carey VJ, Bates DM, et al. Bioconductor: open software development for computational biology and bioinformatics. Genome Biol 2004; 5(10): R80. [PubMed: 15461798]
2. Gómez J, García LJ, Salazar GA, et al. BioJS: an open source JavaScript framework for biological data visualization. Bioinformatics 2013; 29(8): 1103–1104. [PubMed: 23435069]
3. Bhagat J, Tanoh F, Nzuobontane E, et al. BioCatalogue: a universal catalogue of web services for the life sciences. Nucleic Acids Res 2010; 38: W689–W694. [PubMed: 20484378]
4. Ison J, Rapacki K, Ménager H, et al. Tools and data services registry: a community effort to document bioinformatics resources. Nucleic Acids Res 2015; 44: D38–D47. [PubMed: 26538599]
5. Henry VJ, Bandrowski AE, Pepin A-S, et al. OMICtools: an informative directory for multi-omic data analysis. Database 2014; 2014: bau069.
6. Wang W, Bleakley B, Ju C, et al. Aztec: a platform to render biomedical software findable, accessible, interoperable, and reusable, 2017, https://arxiv.org/abs/1706.06087
7. Ozyurt IB, Grethe JS, Martone ME, et al. Resource Disambiguator for the web: extracting biomedical resources and their citations from the scientific literature. PLoS ONE 2016; 11(1): e0146300. [PubMed: 26730820]
8. Duck G, Nenadic G, Filannino M, et al. A survey of bioinformatics database and software usage through mining the literature. PLoS ONE 2016; 11(6): e0157989. [PubMed: 27331905]

9. Soysal E, Wang J, Jiang M, et al. CLAMP—a toolkit for efficiently building customized clinical natural language processing pipelines. J Am Med Inform Assoc 2018; 25: 331–336. [PubMed: 29186491]

10. Cho H-C, Okazaki N, Miwa M, et al. Named entity recognition with multiple segment representations. Inf Process Manag 2013; 49(4): 954–965.

11. Tang B, Feng Y, Wang X, et al. A comparison of conditional random fields and structured support vector machines for chemical entity recognition in biomedical literature. J Cheminform 2015; 7: S8. [PubMed: 25810779]

12. Van Antwerp M and Madey G. Advances in the sourceforge research data archive. In: Workshop on public data about software development (WoPDaSD) at the 4th international conference on open source systems, Milan, 2008, https://flosshub.org/sites/flosshub.org/files/srda2008.pdf

13. Collobert R and Weston J. A unified architecture for natural language processing: deep neural networks with multitask learning In: Proceedings of the 25th international conference on machine learning, Helsinki, 5–9 7 2008, pp. 160–167. New York: ACM.

14. Mnih A and Hinton GE. A scalable hierarchical distributed language model In: Advances in neural information processing systems, Vancouver, BC, Canada, 2009, pp. 1081–1088, https://papers.nips.cc/paper/3583-a-scalable-hierarchical-distributed-language-model.pdf

15. Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space, 2013, https://arxiv.org/abs/1301.3781

16. Guo J, Che W, Wang H, et al. Revisiting embedding features for simple semi-supervised learning. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), Doha, Qatar, 25–29 10 2014, pp. 110–120. Stroudsburg, PA: Association for Computational Linguistics.

17. Duck G, Nenadic G, Brass A, et al. bioNerDS: exploring bioinformatics' database and software use through literature mining. BMC Bioinformatics 2013; 14: 194. [PubMed: 23768135]

18. Amith M, Zhang Y, Xu H, et al. Knowledge-based approach for named entity recognition in biomedical literature: a use case in biomedical software identification In: International conference on industrial, engineering and other applications of applied intelligent systems, Arras, 27–30 6 2017, pp. 386–395. Cham: Springer.

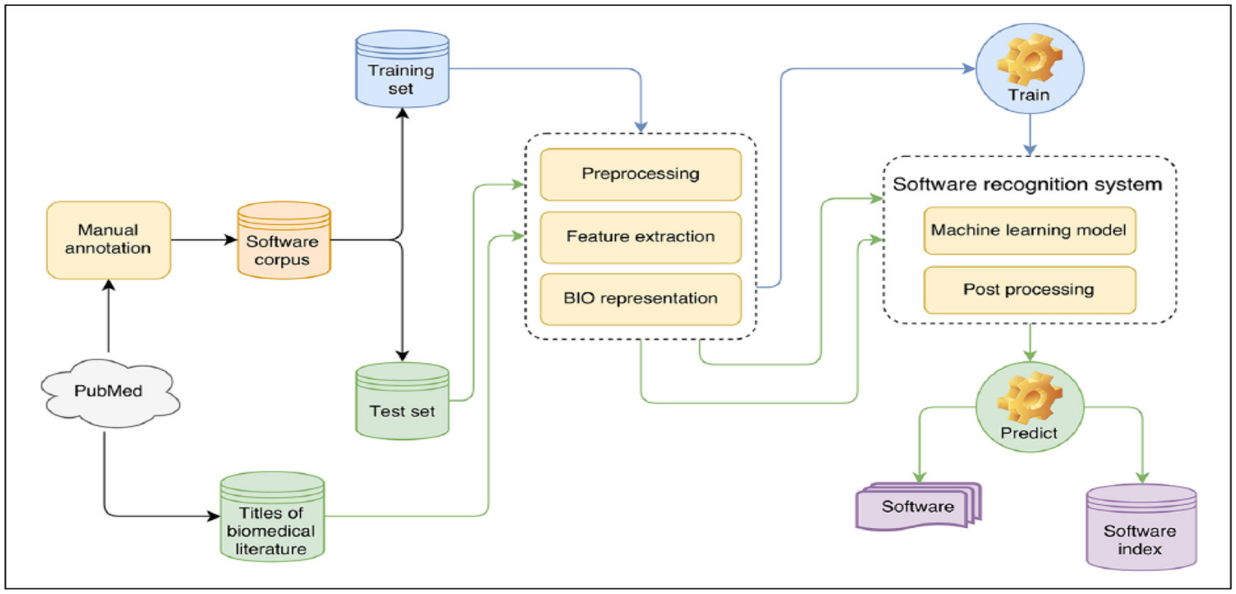
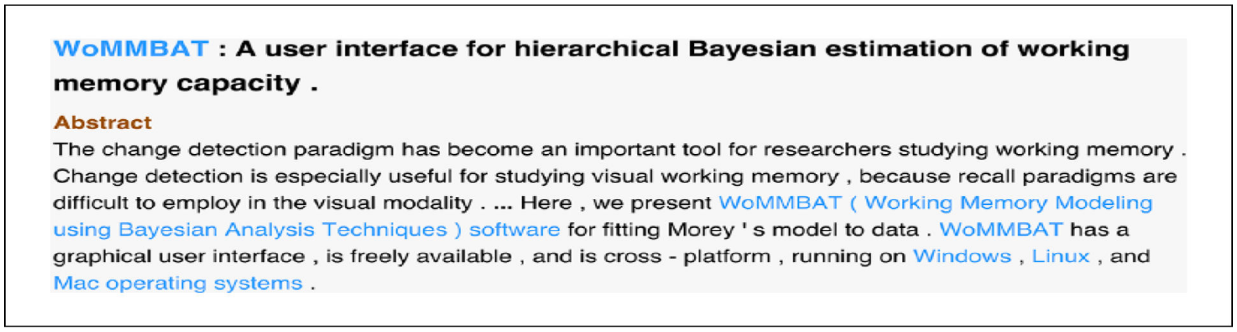


Figure 1.
Study design for automated software recognition from biomedical literature.



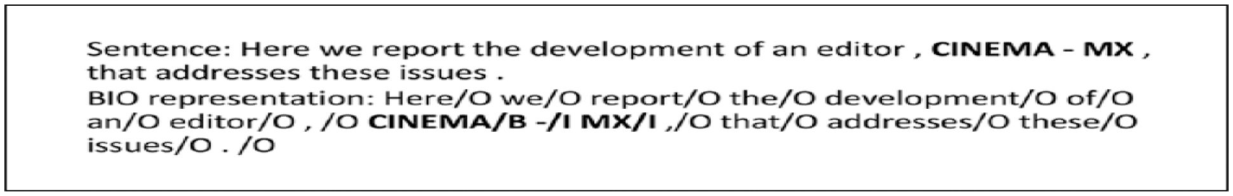
WoMMBAT : A user interface for hierarchical Bayesian estimation of working memory capacity .

Abstract

The change detection paradigm has become an important tool for researchers studying working memory . Change detection is especially useful for studying visual working memory , because recall paradigms are difficult to employ in the visual modality . ... Here , we present WoMMBAT (Working Memory Modeling using Bayesian Analysis Techniques) software for fitting Morey ' s model to data . WoMMBAT has a graphical user interface , is freely available , and is cross - platform , running on Windows , Linux , and Mac operating systems .

Figure 2.

An example of annotated biomedical literature for software names.



Sentence: Here we report the development of an editor , **CINEMA - MX** , that addresses these issues .
BIO representation: Here/O we/O report/O the/O development/O of/O an/O editor/O , /O **CINEMA/B -/I MX/I** ,/O that/O addresses/O these/O issues/O . /O

Figure 3.
An example of BIO representation of software names.

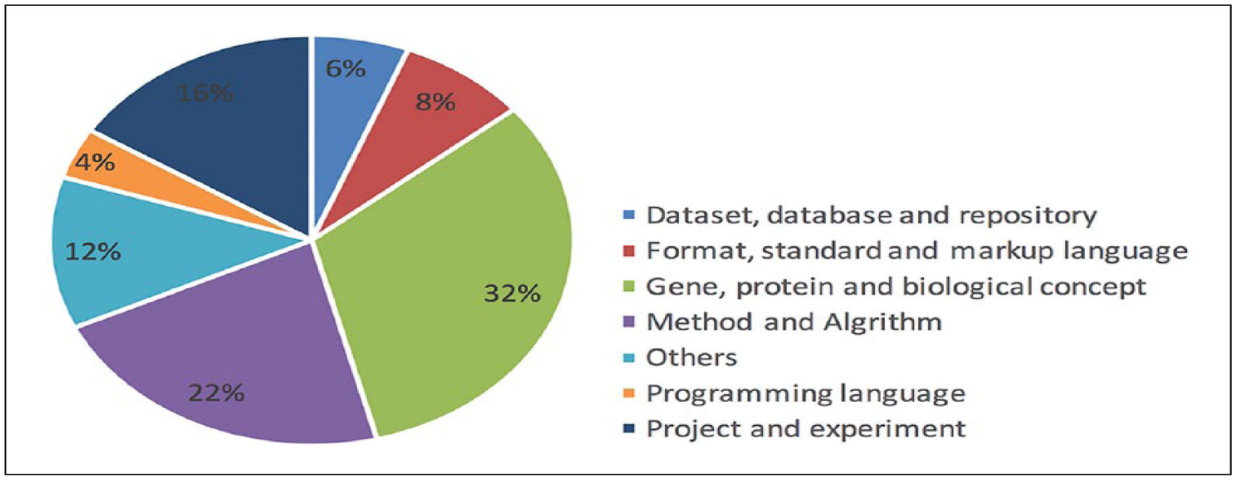


Figure 4.

Distribution of types of concepts misrecognized as software names.

Table 1.

Example of features for developing machine-learning model.

Feature type	Feature values
Word shape feature	StemWord=[wommbat], WordShapel=[AaAAAAA], …
N-gram feature	…, TRIGRAM0=[present+wommbat+(], …, BIGRAM-2=[we+present], …, BIGRAM0=[wommbat +(] …, BIGRAM2=[work+memori] …
Sentence feature	SentFeaLen=[6+], SEN_STARTWITH_ENUM=[FALSE], …
Prefix-suffix feature	Prefix1=[W], Prefix2=[Wo], Prefix3=[WoM], …, Suffix1=[T]
Section feature	Section=[ABSTRACT]
Software name dictionary feature	DictFeaUNI-1=[TK], DictFeaUNI-0=[TK], DictFeaUNI+1=[TK],…
Orthographic feature	RegCAPSMIX=[TRUE], RegEND_PUNCTATION=[FALSE], RegHAS_CAP=[TRUE], RegIS_DASH=[FALSE], …
Word embedding feature (clustering)	EB_0=[NEU], EB_1=[NEU], EB_2=[NEU], EB_3=[NEU], EB_4=[POS], EB_5=[NEU], …
Word embedding feature (discretization)	DLFeaUNI-1=[642], DLFeaUNI-0=[N], DLFeaUN+1=[382], …

Table 2.

Summary of rules for post processing.

	Description
Patterns	(a)The string that is at the beginning of a title and followed by a colon, hyphen and so on could be a software name.
	(b)The string has a pattern of “the * software | package | library | tool | toolkit | bundle | browser” could be a software name.

Table 3.

Performance of software name recognition from biomedical literature (%).

		Precision	Recall	F-measure
Baseline system bioNerDS	Exact	31.52	18.98	23.69
	Inexact	65.25	39.29	49.05
Baseline feature	Exact	81.12	59.38	68.57
	Inexact	92.27	67.54	77.99
Domain knowledge feature				
Dictionary feature	Exact	81.07	59.43	68.59
	Inexact	92.20	67.59	78.00
Orthographic feature	Exact	80.78	59.93	68.81
	Inexact	92.15	68.37	78.50
Section feature	Exact	80.40	60.32	68.93
	Inexact	91.86	68.92	78.76
Word representation feature				
Discrete word embedding feature	Exact	79.31	63.82	70.73
	Inexact	91.10	73.31	81.24
Clustering of word embedding feature	Exact	79.84	64.59	71.41
	Inexact	91.22	73.81	81.60
Post-processing: rule (1a)	Exact	79.28	64.76	71.29
	Inexact	91.24	74.53	82.04
Post-processing: rule (1b)	Exact	78.78	65.32	71.42
	Inexact	90.90	75.36	82.40
Post-processing: rule (2)	Exact	69.65	72.20	70.90
	Inexact	84.69	87.79	86.21
Post-processing: rule (3)	Exact	70.36	71.53	70.94
	Inexact	85.64	87.07	86.35

Each type of feature was added into the software recognition system incrementally.

Table 4.

Performance of software name recognition from titles of biomedical literature (%).

		Precision	Recall	F-measure
	Exact	38.73	21.47	27.63
Baseline system bioNerDS	Inexact	76.88	42.77	54.96
	Exact	90.91	70.51	79.42
Baseline feature	Inexact	97.11	75.32	84.84
Domain knowledge feature				
	Exact	90.98	71.15	79.86
Dictionary feature	Inexact	96.72	75.64	84.89
	Exact	87.27	74.68	80.48
Orthographic feature	Inexact	94.38	80.77	87.05
	Exact	86.19	74.04	79.66
Section feature	Inexact	93.66	80.45	86.55
Word representation feature				
	Exact	88.89	76.92	82.47
Discrete word embedding feature	Inexact	95.19	82.37	88.32
	Exact	87.41	77.88	82.37
Clustering of word embedding feature	Inexact	94.24	83.97	88.81
	Exact	84.25	78.85	81.46
Post-processing: rule (1a)	Inexact	94.18	88.14	91.06
	Exact	84.25	78.85	81.46
Post-processing: rule (1b)	Inexact	94.18	88.14	91.06
	Exact	81.23	80.45	80.84
Post-processing: rule (2)	Inexact	92.23	91.35	91.79
	Exact	81.23	80.45	80.84
Post-processing: rule (3)	Inexact	92.23	91.35	91.79

Each type of feature was added into the software recognition system incrementally.

Table 5.

Reasons and examples of false-positive and false-negative errors in software recognition from biomedical literature.

Error type	**Reasons**	**Examples**
False positive	Similar orthographic characteristics	(a) Predicting AD conversion: comparison between prodromal AD guidelines and computer-assisted PredictAD tool.
		(b) Similarly, one of *the SMAT80* detected proteases was predicted to be a rhomboid protease.
	Similar context	(c) The Sanger *FASTQ* file format for sequences with quality scores, and the Solexa/ Illumina FASTQ variants.
	Complex syntactic structure	(d) One family of algorithms that has proven useful for disease classification is based on relative expression analysis and includes the *Top-Scoring Pair (TSP), k-Top-Scoring Pairs (k-TSP), Top-Scoring Triplet (TST) and Differential Rank Conservation (DIRAC)* algorithms.
False negative	Lack of context Rare pattern	(e) The time consumption was as following: at analysis by *CAMI*, …
		(f) The purpose of this work is to introduce the reader to an Addin implementation, *Decom*.
		(g) RESULTS: A Perl script package called *emerencia* is presented.
		(h) MSDB also contains other two subprograms: *SWR*, which is …, and *SWP*, which is ….
		(i) A thorough user's guide is available within *T4*.

AD: Alzheimer's disease; MSDB: Microsatellite Search and Building Database; SWR: search within results; SWP: sliding window plot.