Comparison of Keyword Extraction and Word Vector Generation Methods for Use in Identifying Related Genomic Datasets

This manuscript (permalink) was automatically generated from J-Wengler/NLP_Paper@4c1275f on March 30, 2021.

Authors

- **James Wengler**
  · J-Wengler
  College of Life Science, Brigham Young University

- **Stephen Picco**
  · srp33
  College of Life Science, Brigham Young University

Abstract

Data-sharing requirements have led to wide availability of genomic datasets in public repositories. Researchers can reuse and combine these datasets to address novel hypotheses. However, after identifying one or more datasets that are relevant to a particular research question, a researcher may have difficult identifying other datasets that are also relevant, due to the quantity of available datasets and lack of structure with which they are described. In this study, we focus specifically on Gene Expression Omnibus, a repository that contains genomic data from hundreds of thousands of experiments. Notable efforts have been made to manually annotate these data but not been able to keep pace as new datasets are submitted. To address this problem, we use natural language processing (NLP). Under the assumption that a researcher has manually identified a subset of available datasets related to a particular research topic, we use NLP algorithms to extract keywords from the abstract associated with each dataset. Next we summarize the keywords using diverse embedding algorithms. (TODO: I'm sure there's a better way to say this.)

TODO: Describe briefly here about the theoretical approaches that we compared more so than the specific software packages. Without making it too long, make sure to cover all of the approaches. Concerning word vector generation we test two primary factors, domain and training algorithm. Domain refers to the type of text that each model is trainined on.

We compare biomedical specific text (StarGEO abstracts) with readily available models trained on wikipedia data. Once a domain has been identified there are two options for training algorithms. The first is continous-bag-of-words (CBOW). CBOW trains by predicting the target word from the context words that surround it. The second option is skip-gram. Skip-gram trains by predicting the context words from the target word essentially the reverse of CBOW.

We test nine keyword extraction methods. Three of the methods are statistical models while the other six are graphical methods. The statistical methods are TF-IDF, KP-Miner, and YAKE. TF-IDF works by comparing the frequency of each word found in the passage to its frequency in other passages that exist in the corpus. KP-Miner evaluates each word based on the context surrounding the words to identify keywords. YAKE combines elements of both TF-IDF and KP-Miner by using the context while also taking into account the frequency at which the word appears in the document. The first graphical approach we test is TextRank which is based off of a web technique called PageRank which is used for identifying related webpages through hyperlinks. TextRank performs a similar analysis with text by creating a graph where each word is represented as a node. Relationships between words are drawn as connected nodes. These relationships are used to identify keywords. TopicRank is a process similar to TextRank but the text is preprocessed to create n-grams of nouns and adjectives as keyphrase candidates before creating a graph with them to identify keywords. SingleRank is another extension of TextRank with each node having a weight value assigned to it. PositionRank is a more complicated extension of TextRank where the position of the word within the sentence is assigned a weight along with actual context as in TextRank. TopicalPageRank is another extension of TextRank that seeks to improve experience by weighting those words that appear more in the document. The last graphical approach is MultipartiteRank which uses a multipartite graph to construct the initial graph and calculate weights between nodes.

We found that different combinations of keyword extraction methods and word vector generation yield very different results. This variety was also reflected across the query domains. These results show that natural language processing is a powerful tool that can be harnessed for data collection and more research needs to be done in this area.

Introduction

Natural Language Processing is computational technique that allows computers to process human language [1]. In the past, Natural Language Processing has been used in several biomedical applications such as concept extraction, electronic health record analysis, and text mining. [2,3,4]. However there is a lack of papers detailing natural language approaches to data collection, specifically the collection of relevant datasets for analysis. Recently an article was published that details the difficulty in applying natural language processing techniques to datasets [5]. Some of these challenges are a lack of widely-accepted metadata format, lack of available tools, and an exponential rise in available datasets [5]. In this paper we detail an alternative approach to address this problem.

The major obstacle to data collection for a researcher is a lack of available tools. The aforementioned paper details an approach to help address this issue, but to our knowledge no other approaches exist and no approach is designed for a researcher who has already identified a niche area of research they wish to pursue. Our methodology utilises two techniques widely used in natural language processing, namely keyword extraction and word vector generation [6,7]. Using these two tools, our approach can take several pre-identified datasets and identify other related datasets, no matter how niche the subject area. We test a variety of these different techniques to identify those that are most promising.

Methods

Data Collection

To measure the accuracy of our method, we only used data that had already been manually identified as related. To achieve this we used the Search Tag Analyze Resource for GEO application (StarGEO). StarGEO is a collection of datasets from the Gene Expression Omnibus that have manually annotated by biomedical graduate students to facilitate the task of collecting related datasets [8]. For each entry in StarGEO the dataset has an abstract, title, and Gene Expression Omnibus accession number. Each dataset in StarGEO has been hand curated and attached to several tags that seek to categorize on a broad level the type of data encapsulated [8]. To ensure a mixture of broad and narrow queries, as well as different domains, we selected six queries to test. While StarGEO has a variety of species to choose from, all queries were filtered to only return human genomic data. Two of which are broad with ~100 articles returned, two are medium with ~20 articles returned and two are small with ~10 articles returned. These queries cover a wide range of domains. It is important to to note that since StarGEO is an ongoing project, it is likely that these queries currently have more articles than at the time of writing.

Keywords	***Number of Articles Returned by StarGeo***
Family History + Breast Cancer	6
Liver Damage + Hepatitis	9
Monozygotic Twins	25
Kidney + Tumor + Cell Line	16
Diabetes + Type 1	97
Osteosarcoma	112

The purpose of widely varying the amount of articles returned is to ensure that our method works for narrowly defined topics as well as broad topics as well as to test the difference. All data accession was performed using the StarGEO API [9]. The API was accessed and stored in a dictionary keyed by accession number to combined abstract and title.

Model Collection

Once the data was collected the next step was identifying the techniquesResearch has already shown that a variety of natural language processing models are effective on biomedical literature [10,11]. However we felt it was necessary to test a variety of techniques for both keyword extraction and word vector generation to identify the combination most suited to our unique task.

Keyword Identification is the first step that our data in the dictionary goes through. To accomplish quick querying in a standardized framework we sued the PKE package available in Python. All documentation is available on GitHub [12]. The PKE packages allowed us to use one single package to access all the keyword embedding techniques, instead of individually querying each technique. The keyword extraction techniques that were tested are the following: TFIDF, KPMiner, YAKE, TextRank, SingleRank, TopicRank, TopicalPageRank, PositionRank and MultipartiteRank [12]. Using each of these techniques keywords were identified from the target text. Each of these techniques is technologically diverse and we chose them due to the expectation that the techniques would yield different keywords.

After the keywords have been identified from the target text, word vectors are generated from each keyword using a word vector model. These models use large amounts of unlabeled text to identify the meanings of words and express those as a numeric vector. For our project we selected 6 different models that encompass a variety of techniques and training data. The two major frameworks used in this project were Spacy and FastText. These models are the two most widely used frameworks in natural language processing and both have been used in biomedical applications [10,13,14]. However there are differences in how these models train on sample text and generate word vectors. For these reason we chose to test both platforms.

The source of training data is an important aspect of generating word vectors. Recent literature supports matching the training data to the testing data [10]. However the benefit of keeping the training and testing data in the same domain is not supported in all the literature [11]. To test this effect we have models trained on both biomedical literature and other sources such as web blogs, news, and Wikipedia [15]. This training data has several possible algorithms for processing and generating word vectors. The two majors options are Continuous-Bag-Of-Words (CBOW) and SKIPGRAM, a complete explanation of these algorithms is outside the scope of this paper. Both of these algorithms have been shown effective on biomedical natural language processing, but small differences have been shown between the word vectors generated from either algorithm [16]. Due to this, both algorithms are tested. There are a total of 6 models tested in this paper. FastText and Spacy are compared head to head, as well as different algorithms and training data. A summary of each model and brief details are shown below.

Model	***Summary***
BioWordVec	FastText Model trained on generic biomedical data with SKIPGRAM
FastTextWiki	FastText model trained on Wikipedia data with CBOW
FastTextCBOW	FastText model trained on GEO data using CBOW
FastTextSKIPGRAM	FastText model trained on GEO data using SKIPGRAM
SciSpacy	A Spacy model designed for usage in biomedical applications
SpacyWebLG	A Spacy model desined for generic usage

Vector Generation

The initial step of vector generation is the identification of the top ten keywords from the text source. Each one of these keywords is turned into a word vector. This word vector is stored as a list of numeric values. The models vary in the length of this numeric vectors from 100 - 300. Once each vector has been generated from each keyword these vectors are added element-wise and then divided by the number of keywords. This technique has been shown to be the simplest and most accurate way to generate a singular word vector from various word embeddings and is usually used to generate document-wide word embeddings [17].

Manual Gene Expression Omnibus Evaluation

Gene Expression Omnibus (GEO) is the parent corpus from which StarGEO is derived [18]. To compare our technique directly to GEO we use a manual technique. We first use the advanced search option on GEO to input the exact queries we used from StarGEO. To maintain consistency with StarGEO, the results are limited to series and human genomic data. A summary file of all the results is downloaded and analyzed. To ensure equal comparision the results are filtered to only include those datasets that exist in StarGEO's corpus. Using the same technique for the StarGEO evaluation the top

1,10,100,500 articles are identified and compared against the relevant articles to identify the number relevant.

Model Evaluation

All model evaluation is performed in a Docker container to allow other researchers to perform the same analysis described in this section [19].The Docker image used to build the container is the python:3.8.5 image available on the Docker website [20]. Running the docker container as pulled from github will run a bash script that performs the following steps. 1. StarGEO is queried to prepare the six queries. The prepareQueryData.py script takes two arguments. The first is a list of GEO identifiers and the second is the query number that these identifiers should belong to. PrepareQueryData.py creates a file system that contains all the abstract and titles of the series that correspond to each identifier. The file system will put each text file into the directory for the corresponding query. 2. GetGeoQueries.py is run. This script uses text files generated by GEO to evaluate the performance of GEO. A detailed explanation of this is found below in the manual comparision section. 3. A do loop iterates over the numbers 10,20, and 30. This numbers are the number of keywords that each model should try to identify from the text. Each iteration performs the following analysis. i. Six scripts that correspond to SciSpaCy, BioWordVec, FastTextWiki, SpaCy, FastTextSkipGram, and FastTextCBOW are run. Each of these scripts takes the following three arguments : number of keywords, vector size, and number of StarGEO articles. Each script performs the following steps: a. All candidate articles from StarGEO are queried b. The specific word vector model is loaded (SciSpaCy, BioWordVec, ...) c. For each query and keyword combination findSimilarity() is run in Helper.py and added to a multiprocessing thread. This script prints to an output file the calculated similarity of each article using each combination #FIXME -> Should we go into detail here?

Results

The two main techniques in this paper are keyword identification and word vector generation. Both of these methods are described below.

Keyword Identification

Keyword extraction is a vital part of the analysis. There are a variety of techniques to achieve this, and we test the most common 9 techniques in this paper. Each technique performs the analysis slightly differently and this leads to variation in the keywords identified. An example of the variation is shown below.

Sample Abstract -> "BRCA1 and BRCA2 are the genes related with breast and ovarian cancer. They have function in DNA repair processes and thus they are tumor suppressor genes. There are hundreds of mutations identified in these genes. Functional deficiencies due to these mutations impair DNA repair and cause irregularities in the DNA synthesis. The standard method for the laboratory assessment of these BRCA genes includes comprehensive sequencing and testing of broad genomic rearrangements. Members of the families with BRCA mutations have an increased risk for early onset of breast cancer and ovarian cancer occurring at any age."

Keyword Extraction Technique	***Top 3 Keywords returned***
TopicRank	'mutations', 'breast', 'dna repair processes'
TextRank	'dna repair processes', 'tumor suppressor genes', 'serum ca-125 levels'
SingleRank	'brca mutations', 'brca genes', 'breast cancer'
TopicalPageRank	'brca genes', 'brca mutations', 'tumor suppressor genes'
MultipartiteRank	'mutations', 'genes', 'dna repair processes'

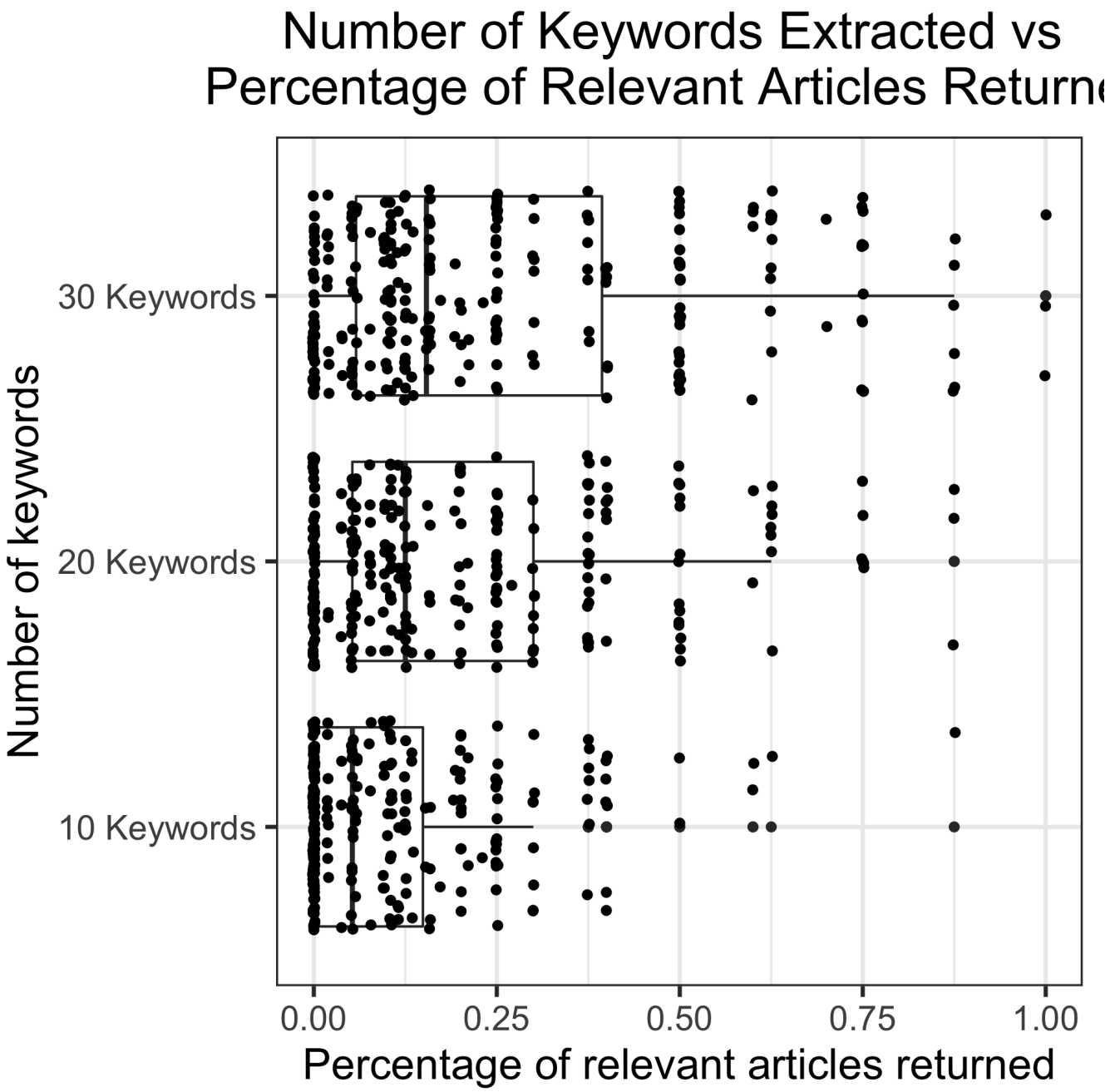
Word Vector Generation

Vector generation is how similarities between articles are calculated. This allows us to give a numerical percentage to quantify the relationship between two datasets. In our analysis we test 6 models that can generate vectors. Each model is trained on unique text and will yield slightly different word vectors. This in turn will generate slightly different cosine similarities. An example of vector generation is shown below:

Word	***Vector***
Database	[1.3863622 1.0939984 -2.1352 -1.9841313 -0.31141075 1.3959851 ...]
Gene	[1.4969006 2.7855976 -4.313326 -2.5572329 -0.9275282 0.43499815 ...]
Mutation	[2.7130241e+00 2.5561374e-01 -2.1098554e+00 -2.1719341e+00 ...]
Disease	[1.9606729e+00 3.5872436e-01 -2.9315462e+00 -2.3048987e+00 ...]

Evaluation Results

Effect of Number of Keywords Returned on the Percentage of Relevent Articles Returned at 100 Articles



Summary_Graph_Keywords.

30 keywords

This graph is an example of a graph generated by the AllGraphs script in /images/. All other keywords and models are contained in the appendix. This graph is 30 keywords using the SciSpacy model.
SciSpacy.

Query Specific Results

This table shows the best performance for each query evaluated by the percentage of relevant article returned in the top 100 returned by the model + keyword extraction technique.

Query	***Model***	***Keyword Method***	***Percentage***
BRCA + Cancer	FastTextWiki	TFIDF	21.1%
Acute_Leukemia + MLL + Progression	Spacy	TopicRank	70.0%
Helicase + Replication + Deletion	FastTextWiki	TFIDF	87.5%
H1N1 + Infection + Mouse + Lethal	Spacy	KPMiner	100.0%
Metastasis + Brain	FastTextWiki	TextRank	25.0%

Discussion

Overview

The purpose of this project was to illustrate the usage of Natural Language Processing in the data collection phase of any project and to identify techniques to use in future projects. NLP has already been shown to be useful to find related articles of scientific nature [21,22,23]. However to our knowledge no project has been done comparing word vector generation and keyword extraction techniques for usage in data collection. This is addressed in our paper in the head to head comparision of these techniques. We hope this will further our knowledge as to how natural language processing might help researchers in future studies.

Observations

The results show a wide variety of accuracy across the queries. This pattern of the same natural language processing technique giving very disparate results on intrinsic evaluations is one commonly seen in natural language processing papers [7,16,24]. Of note is the fact that the two queries that heavily under-perform ate the most broad queries ("BRCA + Cancer" and "Brain and Metastatsis"). These queries were added to test the efficacy of this method with ill-defined queries. Both of these queries also return the highest number of articles (19 and 52). Part of the issue with these articles is choosing three random articles may choose three niche articles that do not represent the wider query. The best performing query is "H1N1 + Infection + Mouse + Lethal" which is arguably the most specific query because it represents a disease, model, and outcome. With the Spacy model and KPMINER keyword extraction method, 100% of the relevant articles are found in the top 100 articles returned.

Practical Utility

Our results show a practical utility for this technique to a researcher who is interested in a very specific knowledge base. If a researcher has previously identified several articles that deal with a narrowly defined subject area, using this technique to query a larger database (not StarGEO) would result in the discovery of potentially all the related datasets that exist in that database. Using this technique, the researcher can bypass the arduous process of collecting datasets and trying to determine which are useful. This technique can also be used to look at multiple datasets within in a broad context. While not useful for potentially finding a niche dataset from a broad query, the ability of this technique to find even distantly related dataset could be used to facilitate a broad understanding of the datasets related to a concept.

It is important to note that this is not a well-polished tool free of bugs. This is a proof of concept that can be applied in various situations to yield useful results. Any customization would require the manual editing of the code to fit the use cause.

Limitations

There are several limitations to our approach. Most obvious is a lack of methods to compare it against. There exists no other tool for gathering related datasets from an intial cohort of datasets. This means that the only external evaluation possible is to compare it to hand curation of datasets by other researchers. We did consider this option but rejected it due to the subjective nature of human curation and the time it would require of the participants. There are also other keyword extraction and word vector generation techniques. Our keyword extraction techniques were limited to those available in the PKR Python package to ensure consistency. Word vectors were limited to the those

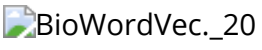
that were most commonly tested and available in other NLP related papers [11,25]. We considered testing a wide variety of techniques across all

This technique also ran into computational limitations. We initially started with 10 keywords, then increased to 20 and eventually 30. We saw an increase in percentage of related articles but also an increase in time taken for the analysis to be performed. 10 keywords took approximately a week, with an increase to approximately 3 weeks for 30 keywords. All analysis were performed on a Dell PowerEdge R730xd server with two Intel Xeon E5-2640 v4 2.4GHz CPUs that each support 10 cores with two threads apiece with a total of 256gb of RAM. All analyses were performed using the Multiprocesing Python package to take advantage of all cores available.

Appendix

30 Keywords

20 Keywords



10 Keywords

References

1. **Natural language processing: an introduction**
   Prakash M Nadkarni, Lucila Ohno-Machado, Wendy W Chapman
   Journal of the American Medical Informatics Association (2011-09-01) https://doi.org/c3r4n3
   DOI: 10.1136/amiajnl-2011-000464 · PMID: 21846786 · PMCID: PMC3168328

2. **BioBERT: a pre-trained biomedical language representation model for biomedical text mining**
   Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang
   Bioinformatics (2019-09-10) https://doi.org/ggh5qq
   DOI: 10.1093/bioinformatics/btz682 · PMID: 31501885 · PMCID: PMC7703786

3. **Natural language processing (NLP) tools in extracting biomedical concepts from research articles: a case study on autism spectrum disorder**
   Jacqueline Peng, Mengge Zhao, James Havrilla, Cong Liu, Chunhua Weng, Whitney Guthrie, Robert Schultz, Kai Wang, Yunyun Zhou
   BMC Medical Informatics and Decision Making (2020-12-30) https://doi.org/ghs7xp
   DOI: 10.1186/s12911-020-01352-2 · PMID: 33380331 · PMCID: PMC7772897

4. **Overview of the First Natural Language Processing Challenge for Extracting Medication, Indication, and Adverse Drug Events from Electronic Health Record Notes (MADE 1.0)**
   Abhyuday Jagannatha, Feifan Liu, Weisong Liu, Hong Yu
   Drug Safety (2019-01-16) https://doi.org/ghs53b
   DOI: 10.1007/s40264-018-0762-z · PMID: 30649735 · PMCID: PMC6860017

5. **A content-based dataset recommendation system for researchers—a case study on Gene Expression Omnibus (GEO) repository**
   Braja Gopal Patra, Kirk Roberts, Hulin Wu
   Database (2020) https://doi.org/ghktfx
   DOI: 10.1093/database/baaa064 · PMID: 33002137 · PMCID: PMC7659921

6. **pke: an open source python-based keyphrase extraction toolkit**
   Florian Boudin
   Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations (2016-12) https://www.aclweb.org/anthology/C16-2015

7. **BioWordVec, improving biomedical word embeddings with subword information and MeSH**
   Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin, Zhiyong Lu
   Scientific Data (2019-05-10) https://doi.org/gf63th
   DOI: 10.1038/s41597-019-0055-0 · PMID: 31076572 · PMCID: PMC6510737

8. **Precision annotation of digital samples in NCBI's gene expression omnibus**
   Dexter Hadley, James Pan, Osama El-Sayed, Jihad Aljabban, Imad Aljabban, Tej D. Azad, Mohamad O. Hadied, Shuaib Raza, Benjamin Abhishek Rayikanti, Bin Chen, … Atul J. Butte
   Scientific Data (2017-09-19) https://doi.org/gbv379
   DOI: 10.1038/sdata.2017.125 · PMID: 28925997 · PMCID: PMC5604135

9. **STAR | Redefining the meaning of disease… Together!** http://stargeo.org/api_docs/

10. **How to Train good Word Embeddings for Biomedical NLP**
    Billy Chiu, Gamal Crichton, Anna Korhonen, Sampo Pyysalo

Association for Computational Linguistics (ACL) (2016) https://doi.org/gfxvff
DOI: 10.18653/v1/w16-2922

11. **A comparison of word embeddings for the biomedical natural language processing**
Yanshan Wang, Sijia Liu, Naveed Afzal, Majid Rastegar-Mojarad, Liwei Wang, Feichen Shen, Paul Kingsbury, Hongfang Liu
Journal of Biomedical Informatics (2018-11) https://doi.org/ggbx8b
DOI: 10.1016/j.jbi.2018.09.008 · PMID: 30217670 · PMCID: PMC6585427

12. **boudinfl/pke**
Florian Boudin
(2021-03-30) https://github.com/boudinfl/pke

13. **SMAC, a computational system to link literature, biomedical and expression data**
Stefano Pirrò, Emanuela Gadaleta, Andrea Galgani, Vittorio Colizzi, Claude Chelala
Scientific Reports (2019-07-19) https://doi.org/ghm6ct
DOI: 10.1038/s41598-019-47046-2 · PMID: 31324861 · PMCID: PMC6642118

14. **Fast and scalable neural embedding models for biomedical sentence classification**
Asan Agibetov, Kathrin Blagec, Hong Xu, Matthias Samwald
BMC Bioinformatics (2018-12-22) https://doi.org/ghm6cv
DOI: 10.1186/s12859-018-2496-4 · PMID: 30577747 · PMCID: PMC6303852

15. **Enriching Word Vectors with Subword Information**
Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov
arXiv:1607.04606 [cs] (2017-06-19) http://arxiv.org/abs/1607.04606

16. **Word2vec convolutional neural networks for classification of news articles and tweets**
Beakcheol Jang, Inhwan Kim, Jong Wook Kim
PLOS ONE (2019-08-22) https://doi.org/ghg3sp
DOI: 10.1371/journal.pone.0220976 · PMID: 31437181 · PMCID: PMC6705863

17. **An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation**
Jey Han Lau, Timothy Baldwin
arXiv:1607.05368 [cs] (2016-07-18) http://arxiv.org/abs/1607.05368

18. **Gene Expression Omnibus: NCBI gene expression and hybridization array data repository**
R. Edgar
Nucleic Acids Research (2002-01-01) https://doi.org/fttpkn
DOI: 10.1093/nar/30.1.207 · PMID: 11752295 · PMCID: PMC99122

19. **An introduction to Docker for reproducible research**
Carl Boettiger
ACM SIGOPS Operating Systems Review (2015-01-20) https://doi.org/gdz6f9
DOI: 10.1145/2723872.2723882

20. **Docker Hub** https://hub.docker.com/_/python

21. **Editorial: Mining Scientific Papers: NLP-enhanced Bibliometrics**
Iana Atanassova, Marc Bertin, Philipp Mayr
Frontiers in Research Metrics and Analytics (2019-04-30) https://doi.org/ghcpfz
DOI: 10.3389/frma.2019.00002

22. **NLP Scholar: An Interactive Visual Explorer for Natural Language Processing Literature**
Saif M. Mohammad
arXiv:2006.01131 [cs] (2020-05-31) http://arxiv.org/abs/2006.01131

23. **tl;dr: this AI sums up research papers in a sentence**
Jeffrey M. Perkel, Richard Van Noorden
Nature (2020-11-23) https://doi.org/ghmnjj
DOI: 10.1038/d41586-020-03277-2 · PMID: 33230274

24. **An Empirical Evaluation of doc2vec with Practical Insights into Document Embedding Generation**
Jey Han Lau, Timothy Baldwin
arXiv:1607.05368 [cs] (2016-07-18) http://arxiv.org/abs/1607.05368

25. **Probabilistic FastText for Multi-Sense Word Embeddings**
Ben Athiwaratkun, Andrew Gordon Wilson, Anima Anandkumar
arXiv:1806.02901 [cs, stat] (2018-06-07) http://arxiv.org/abs/1806.02901