Keyword Extraction for Analyzing Popularity of Algorithms

Preksha Mutha pjm526@nyu.edu
Kiranmai Nallamalli kn1429@nyu.edu
Snehal Kenjale svk304@nyu.edu
Omkaar Khanvilkar opk207@nyu.edu

1 Acknowledgement

We would like to express our deepest appreciation to all those who provided us with key information related to the project topic. A special gratitude we give to our course professor Prof. Juan C Rodriguez, whose contribution in stimulating suggestions and encouragement helped us to complete this project.

2 Introduction

In research and news articles, keywords form an important component since they provide a concise representation of the article's content. Keywords play a crucial role in locating the article from information retrieval systems, bibliographic databases and for search engine optimization. Keywords also help to categorize the article into the relevant subject or domains.

Conventional approaches of extracting keywords involve manual assignment of keywords based on the article content and the authors' judgment. This involves a lot of time and effort and also may not be accurate in terms of selecting the appropriate keywords. With the emergence of Natural Language Processing (NLP), keyword extraction has evolved into being effective as well as efficient.

In this project, we'll be applying NLP on a collection of research papers to extract keywords.

3 Technologies Used

This section will give a briefing of the various technologies used for realizing the goal of identifying the popularity of algorithms.

3.1 Big Data Technologies

Here we will discuss the Big Data concepts applied on the Data in order to make it scalable and to feed it to the Machine Learning Algorithms.

3.1.1 MongoDB

MongoDB supports a rich and expressive object model and is flexible in query modelling. It is a NoSQL database where data is stored in a relatively unstructured fashion. It is a leading database used as big data sources. MongoDB provides high partition tolerance and maintains the data consistency. The MongoDB Spark Connector (version 2.4) integrates MongoDB and Spark environment, providing users with the ability to process data in MongoDB with the massive parallelism of Spark.

3.1.2 Spark

Spark is an open-source distributed general-purpose cluster-computing framework. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance. Spark is a unified analytics engine for large-scale data processing. It provides high-level APIs in Scala, Java, Python, and R, and an optimized engine that supports general computation graphs for data analysis.

It also supports a rich set of higher-level tools including Spark SQL for SQL and DataFrames, MLlib for machine learning, GraphX for graph processing, and Structured Streaming for stream processing. Spark makes our project scalable.

3.1.3 Spark ML

Spark ML is a new package introduced in Spark 1.2, which aims to provide a uniform set of high-level APIs that help users create and tune practical machine learning pipelines.Machine learning can be applied to a wide variety of data types, such as vectors, text, images, and structured data. Spark ML adopts the SchemaRDD from Spark SQL in order to support a variety of data types under a unified Dataset concept. SchemaRDD supports many basic and structured types; see the Spark SQL datatype reference for a list of supported types. In addition to the types listed in the Spark SQL guide, SchemaRDD can use ML Vector types. A SchemaRDD can be created either implicitly or explicitly from a regular RDD.

3.2 Python Libraries

Python is a programming language that lets you work quickly and integrate systems more efficiently. The python libraries are used to process the data in order to be feed to the ML algorithm in order to achieve superior results.

3.2.1 Natural Language Processing Toolkit

(NLTK)[1]. is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more.

3.2.2 NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.The core functionality of NumPy is its "ndarray", for n-dimensional array, data structure. These arrays are strided views on memory. In contrast to Python's built-in list data structure (which, despite the name, is a dynamic array), these arrays are homogeneously typed: all elements of a single array must be of the same type.

3.2.3 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

3.2.4 WordCloud

Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. Significant textual data points can be highlighted using a word cloud. Word clouds are widely used for analyzing data from social network websites.For generating word cloud in Python, modules needed are – matplotlib, pandas and wordcloud.

[1]Natural Language Processing Toolkit:`https://www.nltk.org/`

4 Dataset

In this project, we will be extracting keywords from a dataset that contains about 7939 research papers. The dataset is from Kaggle - (NIPS Paper)[2]. Neural Information Processing Systems (NIPS) is one of the top machine learning conferences in the world. This dataset includes the title and abstracts and paper text for all NIPS papers to date (ranging from the first 1987 conference to the 2016 conference).

We will be processing on the paper_text column of the dataset which holds the entire text from the research paper.


```
_id: ObjectId("5cdc5ae47c689e05703cf114")
id: "the MRE solution pee"
year: " "
title: ") . For example"
event_type: " {e \ b"
pdf_name: " b"
abstract: " "
paper_text: "} provides such a factorization ."
```


Figure 1: Representation of Dataset in JSON format

5 Architecture

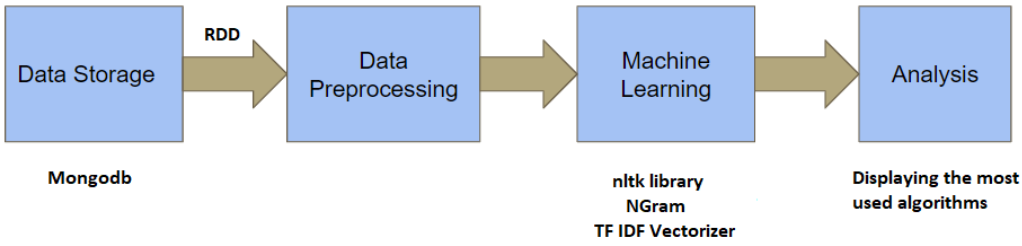


Figure 2: Data Flow Architecture

The first stage of the architecture is to load the dataset from MongoDB using the MongoDB Spark Connector in RDD format in Spark. This RDD is then sent for data preprocessing. The in depth preprocessing methods are described below. After data is fully ready, it is used in the Spark ML models like NGram and TF-IDF Vectorizer. Finally, the result is then analyzed to get the top most frequently used algorithms.

6 Data Preprocessing Steps

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues.In Real world data are generally incomplete: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data. Noisy: containing errors or outliers. Inconsistent: containing discrepancies in codes or names.

[2]Kaggle NIPS Paper:https://www.kaggle.com/benhamner/nips-papers/downloads/nips-papers.zip/

6.1 Lower Case Representation

The dataset is made uniform by converting it to lowercase. This is a pretty vital step as the algorithms will create two separate instances for the same word in different cases, hence in order for the algorithm to function efficiently we convert the text to lower case.

6.1.1 Tokenization

Tokenization is a step which splits longer strings of text into smaller pieces, or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. Further processing is generally performed after a piece of text has been appropriately tokenized. Tokenization is also referred to as text segmentation or lexical analysis. Sometimes segmentation is used to refer to the breakdown of a large chunk of text into pieces larger than words (e.g. paragraphs or sentences), while tokenization is reserved for the breakdown process which results exclusively in words. For Analysis, we first break down the text into sentences and the obtained sentences are further broken down into words. This improves the speed of execution.

6.2 Removal of Stop words

Stop words are those words which are filtered out before further processing of text, since these words contribute little to overall meaning, given that they are generally the most common words in a language. Stop words include the large number of prepositions, pronouns, conjunctions etc in sentences. These words need to be removed before we analyze the text, so that the frequently used words are mainly the words relevant to the context and not common words used in the text.There is a default list of stop words in python nltk library.

6.3 Removal of Special Characters and Punctuation

Since the special characters and punctuation do not constitute a meaningful word, hence they should be removed from the data before feeding it to the Algorithm. If special characters are not removed they could adversely effect the performance the Count Vectorizer.

6.4 Lemmatization

Stemming is the process of eliminating affixes (suffixed, prefixes, infixes, circumfixes) from a word in order to obtain a word stem. Lemmatization is related to stemming, differing in that lemmatization is able to capture canonical forms based on a word's lemma or root.

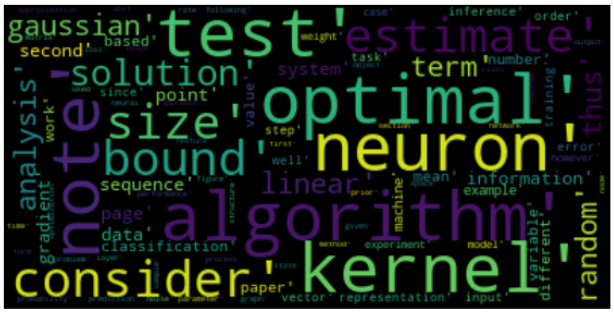


Figure 3: Wordcloud Corpus

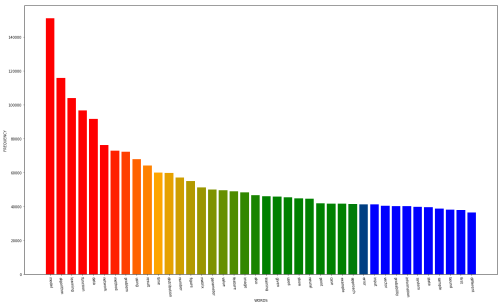
7 Models Used

In this section we will demonstrate the learning algorithms used in order to find out most used words and the most popular techniques currently in use. For this project we have used N-grams and TF-IDF models.

```
In [27]: filteredWords

Out[27]: [('model', 150984),
          ('algorithm', 115837),
          ('learning', 104005),
          ('function', 96738),
          ('data', 91776),
          ('network', 76318),
          ('method', 73024),
          ('problem', 72250),
          ('using', 67961),
          ('result', 64272),
          ('time', 60113),
          ('distribution', 59938),
          ('number', 57145),
          ('figure', 54992),
          ('matrix', 51319),
          ('parameter', 49976),
          ('value', 49575),
          ('feature', 49096),
          ('image', 48323),
```

(a) Uni-grams by frequency

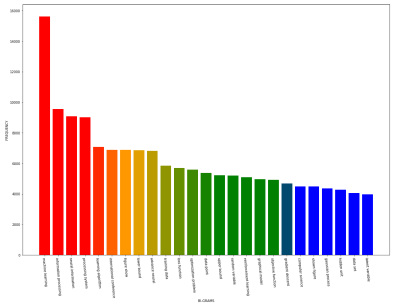


(b) Uni-grams plot

Figure 4: Uni-gram Visualizations

0	machine learning	15631
1	information processing	9566
2	neural information	9098
3	processing system	9024
4	learning algorithm	7092
5	international conference	6897
6	figure show	6890
7	lower bound	6857
8	advance neural	6820
9	training data	5853
10	loss function	5692
11	optimization problem	5590
12	data point	5374
13	upper bound	5222

(a) Bi-grams by frequency



(b) Bi-grams plot

Figure 5: Bi-gram Visualizations

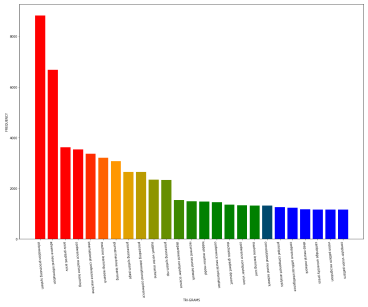
7.1 N-grams

N-grams are contiguous sequences of n-items in a sentence where n can be 1, 2 or any other positive integers. However n cannot be very large as these grams will rarely occur. In the project we calculate uni-grams first to get an idea of the most occurring words in the paper text. Finding uni-grams helped us to further clean the data by removing customized stop words. Since we aim to analyze which algorithms/models have been researched upon, we calculated bi-grams as well. In order to achieve accurate results, we calculated tri-grams using the NGram Model. This gave us a list of all the trigrams. We calculate the aggregate count of each trigram to get the most occurring trigrams in the list.

```
In [45]: tri

Out[45]: ['information processing system',
          'advance neural information',
          'arxiv preprint arxiv',
          'conference machine learning',
          'international conference machine',
          'machine learning research',
          'journal machine learning',
          'processing system page',
          'proceeding international conference',
          'support vector machine',
          'processing system nip',
          'department computer science',
          'recurrent neural network',
          'hidden markov model',
          'conference neural information',
          'stochastic gradient descent',
          'conference computer vision',
          'machine learning icml',
          'convolutional neural network',
          'principal component analysis'
```

(a) Tri-grams by frequency



(b) Tri-grams plot

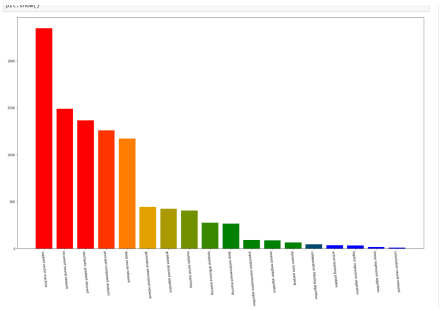
Figure 6: Tri-gram Visualizations

In [53]: final_df

Out[53]:

	Algorithms	Count
9	support vector machine	2345
12	recurrent neural network	1489
15	stochastic gradient descent	1363
19	principal component analysis	1259
21	deep neural network	1171
96	generative adversarial network	441
106	gradient descent algorithm	424
114	multiple kernel learning	404
208	temporal difference learning	274
217	deep reinforcement learning	263
1250	expectation maximization algorithm	91
1371	nearest neighbor algorithm	85
2162	dynamic time warping	65
3974	collaborative filtering algorithm	46
6380	active learning problem	34
7640	logistic regression algorithm	31
22898	linear regression algorithm	16
47637	convolution neural network	10

(a) Words by frequency



(b) Most Researched Algorithm

Figure 7: Analysis of Algorithms used in Research Papers

7.2 TF-IDF

TF-IDF, which stands for term frequency-inverse document frequency, is a scoring measure widely used in information retrieval. TF-IDF is intended to reflect how relevant a term is in a given document by giving equal importance to every word provided.

8 Results

The results of our analysis are shown in figure 7. The output contain a list of tri-grams which are actually relevant. When observed, we found a few generic tri-grams and hence filtered the list on the word “Algorithm”. This gave us a much accurate list of all the algorithms used in the papers with their counts. In order to explicitly see how much a particular algorithm is used, we made a list of all 62 algorithms used till date and filtered out these algorithms. The plot below gives us an idea about the most researched algorithm and the least researched ones. The biggest challenge faced during the research was to clean the data in order to make it ready to feed it to the ML Algorithm. We also tried to test the LDA model and clustered the papers but providing labels to the clusters was challenging.

9 Conclusion

In this way we conclude that Support Vector Machine, Recurrent Neural Networks, Stochastic gradient descent, Principal component analysis are the most researched ones till 2017. However we can see that Convolution Neural Network is comparatively researched less. Hence there is a scope of more research in this domain which might help to exploit the usage of these algorithms in efficient manner. In the future we look forward to extending this approach to classify papers based on the Algorithms used and also Dynamic Classification for Search Engines.

References

[1] Automated Keyword Extraction: https://medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34

[2]Textrank for Keyword extraction: https://towardsdatascience.com/textrank-for-keyword-extraction-by-python-c0bae21bcec0

[3]Understanding keyword extraction: https://www.knime.com/blog/keyword-extraction-for-understanding

[4] Natural Language Processing Toolkit: http://www.nltk.org/book/

[5]Spark Mllib: https://spark.apache.org/mllib/