Contents

[Contents 1](#_Toc98838388)

[1 Introduction 4](#_Toc98838389)

[1.1 Scope 5](#_Toc98838390)

[1.2 Objectives of the research 5](#_Toc98838391)

[1.3 Overview of the report 6](#_Toc98838392)

[1.4 Summary 6](#_Toc98838393)

[2 The role of Software in Scientific research 7](#_Toc98838394)

[2.1 Introduction 7](#_Toc98838395)

[2.2 General roles of software in a research 8](#_Toc98838396)

[2.3 Domain specific examples 9](#_Toc98838397)

[2.4 The role of software in research breakthroughs 10](#_Toc98838398)

[2.5 Summary 10](#_Toc98838399)

[3 Software usage purpose 11](#_Toc98838400)

[3.1 Introduction 11](#_Toc98838401)

[3.2 Analysis of literature 11](#_Toc98838402)

[3.3 Analysis of software ontologies 12](#_Toc98838403)

[3.3.1 WikiData 12](#_Toc98838404)

[3.3.2 The software ontology (SWO) 16](#_Toc98838405)

[3.3.3 OntoSoft 17](#_Toc98838406)

[3.4 Analysis of Sci-Crunch repository 19](#_Toc98838407)

[3.5 Types of software usage purposes 20](#_Toc98838408)

[3.5.1 Data collection 20](#_Toc98838409)

[3.5.2 Data pre-processing 21](#_Toc98838410)

[3.5.3 Data Analysis 21](#_Toc98838411)

[3.5.4 Data visualization 21](#_Toc98838412)

[3.5.5 Simulation 21](#_Toc98838413)

[3.5.6 Stimulation 22](#_Toc98838414)

[3.5.7 Modelling 22](#_Toc98838415)

[3.5.8 Programing 22](#_Toc98838416)

[3.6 Summary of software usage purposes and examples 23](#_Toc98838417)

[4 Data set 24](#_Toc98838418)

[4.1 Introduction 24](#_Toc98838419)

[4.2 SoMeSci data set 24](#_Toc98838420)

[4.2.1 SoMeSci Articles 24](#_Toc98838421)

[4.2.2 SoMeSci Annotations 25](#_Toc98838422)

[4.3 Annotation tool 25](#_Toc98838423)

[4.3.1 Annotation of SoMeSci with software purpose labels 26](#_Toc98838424)

[4.3.2 Assumptions in the annotation 26](#_Toc98838425)

[4.3.3 Challenges during Annotation 26](#_Toc98838426)

[4.4 Data Pre-processing 27](#_Toc98838427)

[4.4.1 Handling annotation errors and missing annotations 27](#_Toc98838428)

[4.4.2 Merging annotations 27](#_Toc98838429)

[4.4.3 Transformation to IOB format 28](#_Toc98838430)

[4.4.4 Data Splitting 28](#_Toc98838431)

[4.5 Analysis of Annotated Data 28](#_Toc98838432)

[4.5.1 Co-reference resolution of software entities 28](#_Toc98838433)

[4.5.2 Analysis results 29](#_Toc98838434)

[4.6 Summary 30](#_Toc98838435)

[5 Classifier models 31](#_Toc98838436)

[5.1 Introduction 31](#_Toc98838437)

[5.2 Machine learning Models 31](#_Toc98838438)

[5.2.1 Hidden Markov models (HMMs) 31](#_Toc98838439)

[5.2.2 Maximum Entropy Markov Models (MEMMs) 32](#_Toc98838440)

[5.2.3 Linear Conditional Random Fields (CRFs) 32](#_Toc98838441)

[5.3 Deep Learning Models 33](#_Toc98838442)

[5.3.1 Long Short-Term Memory (LSTM) 33](#_Toc98838443)

[5.3.2 Bi-Long Short-Term Memory (BLSTM) 34](#_Toc98838444)

[5.3.3 Bi-LSTM-CRF 34](#_Toc98838445)

[5.3.4 Word embeddings 35](#_Toc98838446)

[6 Model Training and Optimization 36](#_Toc98838447)

[6.1 Introduction 36](#_Toc98838448)

[6.2 Data set split optimization 36](#_Toc98838449)

[6.3 Model training with inclusion/exclusion part of data 37](#_Toc98838450)

[6.4 The impact of context on classifier’s performance 38](#_Toc98838451)

[6.4.1 Left context vs right context sentence 38](#_Toc98838452)

[6.4.2 Context outside a paragraph 39](#_Toc98838453)

[6.5 Classification with 2-layers 40](#_Toc98838454)

[6.6 Model training Bio-BERT vs Sci-BERT 42](#_Toc98838455)

[6.7 Summary of Training Results 42](#_Toc98838456)

[6.7.1 Software entity classification result 43](#_Toc98838457)

[6.7.2 Software purpose 43](#_Toc98838458)

[7 Conclusion and Future work 45](#_Toc98838459)

[7.1 Introduction 45](#_Toc98838460)

[7.2 Conclusion 45](#_Toc98838461)

[7.3 Future work and limitations 45](#_Toc98838462)

# Introduction

Today’s scientific research relies on a use as well as production of various types of research artifacts, a.k.a. research resources, ranging from digital to physical {schindler2019annotation}. One of key digital research artifacts, broadly used in a scientific investigations, is software. Many scientists *use* already existing software for various purposes during their research as well as *create* a new software as part of their research work {goble2014better , hannay2009scientists }.

Almost every software has information associated with it which can be extracted. Version, name of developer, license, abbreviation, URL, citation, release, extension, etc. are among the most obvious examples.

However there are also other kinds of information about a software that are implicit and hidden in the text of a scientific paper. *The purpose of use of a given software in a scientific paper is one example of software information which is often times implicit.*

Other examples of implicit information about a software are regarding usage versus creation of a software. Such information describe weather a software is created in a given scientific investigation or a usage citation of already existing software tool.

Further examples of implicit information about a given software might indicate repository informat

challenging to extract. For example, in a given line of text in a research paper a software can be mentioned to indicate weather a researcher is *using* a software for a particular purpose, introducing a novel software, providing repository information about the deposition of the software, or just mentioning the name of the software. In addition information about category of a software: such as application, plugin, programming environment, operating system, etc. can also be concealed in a textual description in a scientific paper {schindler2021somesci}.

Extraction of all such variations of information about software from the scientific publications has critical importance. This is because, information about a software can be used to uniquely identify each software and avoid ambiguity regarding which software or version researchers have used in their literature. Being able to uniquely identify a software with its specification is also advantageous to guarantee reproducibility of research results as well as providing clear understanding how results of research have been produced {kruger2019literature}. In addition, knowledge about a software’s purpose of use can help to determine which set of software artifacts can be suitable for a given study or to compare results obtained from various software in a given study. Furthermore, knowledge about software use and purpose of use in the literature, supports semantic analysis and retrieval of scientific publications based on use of particular software {schindler2019annotation}.

Even though software citation principles have been already established by a scientific community {callaghan2014joint, smith2016software}, software citation practice in reality is still informal and incomplete {schindler2021somesci}. This makes it difficult to extract information about software that would help to attribute credits to the creators of software, reproduce research results, unambiguously distinguish one software from another, etc.

Various manual and rule based techniques has been attempted in the past to extract information about software. However, machine/deep learning based techniques have not been exploited to their potential until very recently. The main reason was lack of training data which can support training of a classifier for software information extraction {schindler2021somesci}. Producing reliable ground truth data could be accomplished by crowd sourcing for general domains but it is expensive particularly for domain-specific and scientific publications as it requires expert domain knowledge {beltagy2019scibert}. Fortunately, identification of software mentions from scientific articles has drawn more attention over the past years and now various labelled data sets, such as BioNerDs {duck2013bionerds} , SoftCite {du2021softcite}, are available. Recently a more comprehensive data set, SoMeSci, has also been published. SoMeSci contains high quality manually annotated data sets that cover broader range of information about software paving a way for a use of machine learning based approach for the automatic extraction of information about software {schindler2021somesci}.

## Scope

This thesis work tries to apply machine learning technique using SoMeSci data set to automatically extract information about software mentions, particularly, to identify for what purpose a software is used in a given context of text.

To accomplish this, first possible list of software usage purposes have been identified via extensive analysis of literature and other sources like software ontologies and repositories. Then already existing annotations of software usage mentions in the SoMeSci data set has been extended with software purpose labels.

Once software usage mentions in the SoMeSci data set has been labelled with software purpose labels, the data set has been cleaned, analyzed, transformed, and used for classification purpose.

## Objectives of the research

The main objectives to be accomplished in this thesis work are:

* To perform literature review on the importance of software in a research.
* To carry out analysis of literature and software ontologies to identify main types of purposes of software use in a research.
* To extend SoMeSci data set with software usage purpose annotations.
* To perform analysis of SoMeSci data set to drive interesting facts about the data set.
* To select and train a suitable classifier model.
* To optimize and evaluate the model for improved performance.

## Overview of the report

Chapter 1 makes a soft introduction about why it is important to extract information about software, specifies scope and objective of the thesis.

Chapter 2 focuses on highlighting the role of software in a research to indicate driving information about software from scientific publications is an important task.

Chapter 3 focuses to identify possible types of software usage purposes from literature and software ontology. This is an important step taken to extend software usage statements in the SoMeSci data set with software purpose annotations.

Chapter 4 is about the data set. It explains how SoMeSci data set has been extended with software purpose annotations, annotation tool used, and the annotation process. In addition explains about data pre-processing, transformation to suitable format and splitting for classification purpose. At the end, results of analysis of the extended SoMeSci data set has been presented.

Chapter 5 discusses and compares a various models suitable for classification of software purpose from a text.

Chapter 6 focuses model training , optimization and evaluation of various scenarios.

Chapter 7 discusses about classification and results of the evaluation.

Chapter 8 summarizes results and provides conclusion.

## Summary

This chapter has presented a gentle introduction into types of information associated with software artifacts, extraction approaches and why it is important to extract information about software. The data set to be used, scope and goal of the work has also been discussed.

The next chapter presents the role of software in a modern research to give more elaborate understanding about the impact of software in a scientific investigations indicating why it is worth to extract information about software.

# The role of Software in Scientific research

## Introduction

Nowadays scientific research is unthinkable without a use of software and investigations in various areas of science are becoming increasingly reliant on software tools {goble2014better, storer2017bridging, hannay2009scientists, jimenez2017four}.

A software is very important asset for building a scientific knowledge and more discoveries in a research are made possible than ever by a use of software tools that automate processing of huge amount of data {jimenez2017four}. Typically a software is used in a research for data processing tasks such as data analysis, modeling, simulation, control processes, knowledge dissemination, etc. {hannay2009scientists, pan2016disciplinary}.

In modern research, a scientific software is as important as any lab-equipment {wilson2014best}. However, the development of scientific software is much more complicated and fundamentally different from an ordinary commercial software like accounting software. Scientific software requires specialized domain knowledge for its development and requires a direct involvement of domain expert or scientist {wilson2014best, segal2008developing}. Due to this, an increasing number of scientists are developing a software as part of their research work or directly taking part in the development process of a research software {jimenez2017four, kanewala2014testing}.

According to surveys conducted in the UK and USA, 2008 and 2017 respectively, most scientists agree that software plays an important role in their research work {hettrick2014uk, nangia2017track}. Participants of the survey, in UK, were 2000 researchers working in various areas of science in roles ranging from student to senior academic staff whereas participants of the survey, in USA, were members of the US National Postdoctoral Association.

The results from of UK survey {hettrick2014uk} indicate that :

* 38% of researchers spend at least 20% of their time developing a software.
* Almost half of scientists spend more time creating software as part of their research work than five years ago .
* Over 50% of survey respondents reported that they develop their own software.
* Nearly 70% claim that their research directly depends on use of a software &
* Over 90% of scientists say software is important for their research.

The results from of USA survey {nangia2017track} indicate that:

* Over 90% of scientists use software.
* 63% of respondents state that their research is impossible with out using software.
* 31% of scientists say that they could do their work without using a software but more effort would require.
* Only 6% of survey respondents say that there would be no significant difference in their task if they do not use software.

Overall, results from the two surveys clearly indicate that software is pervasive in scientific investigations and many researchers use as well as develop a software for their research.

Even though software plays an important role in a modern research, usually the contributions of software is understated. This can be seen from the poor citation practice of software in research papers across several fields of research {schindler2021somesci,yang2018important, pan2016disciplinary}. In an attempt to promote the recognition of the roles of scientific software in a research, the ReSA has collected literatures that evident the roles software play in a research, at Zetoro group library. The main aim of ReSA is to influences decision makers to acknowledge contributions of a research software and give credits to its developers.

The next section presents more details about the role of software: in general, in specific domains, and in research breakthroughs.

## General roles of software in a research

Software is playing crucial roles in a research and making a shift in a research culture in terms of enabling automation of analysis pipelines, creation of new ways of analysis via computational models, supporting sophisticated analysis of large volume of data, documentation of a research, etc. {jay2020software}.

Some of the most general roles of a software in a research are:

* *A software dictates the quality of a research outcome* {hannay2009scientists}. Outcome of a research becomes unreliable or even useless if there is an error in the software {soergel2014rampant}. For example, several scientists retracted their scientific publications up on a retrospective discovery of a bug in their software {wilson2014best,merali2010computational,miller2006scientist}. A more palpable failure of a research ambition due to an error in the software, for instance, is the failure of *Ariane rocket* in 1996 {enwiki:1054482061}.
* Software helps to explore und understand a research problem {hannay2009scientists}.
* Results from a scientific software is presented as an evidence to support a research conclusion {kanewala2014testing}.
* A software also helps to document a research process and to *validate results of a given research* {jay2020software}. Executable cells in a Jupyter notebook is one real world example where a software can be used to validate a research result.
* Software allows experiments to be made beyond constrains of the physical world. This is because experiments that run on a computer are not limited by processes that occur in nature but only by the laws imbedded in the computer code {wolfram1984computer}.

## Domain specific examples

A software is being extensively used for a research in various areas of science such as physics, chemistry, space science, life science and so on.

The physics research facility, the Large Hydron Collider at CERN, for instance uses a software with more than 5 million lines of code which is used for processing of terabytes of data generated from experiments {storer2017bridging}.

In a nuclear research, a software is being developed increasingly to be used for experiments {yan2017case}. For example, testing a modification in a nuclear weapon can not be done on a field, but instead a software that simulate the impact of modification is usually used {kanewala2014testing}. This is because testing of a nuclear weapon on a field is banned by regulations like nuclear test ban treaties (NTBT) in addition to the potential disaster that testing a nuclear weapon poses to the environment and life {enwiki:1053274189}.

In chemistry research, a software can be used to model and simulate chemical processes that are challenging, too complex or expensive to conduct in reality. Karplus and Levitt used computer simulations for their joint-research “the development of multi-scale models for complex chemical systems” and won a Nobel prize in 2013 for their work {storer2017bridging, andre2014nobel}.

In a climate and environmental studies, software is used to make predictions about climate changes. For example historical temperature data can be integrated to make predictions about future temperature variations {storer2017bridging}.

In a space science, space probes heavily rely on software. In this case a software helps navigate space crafts to other planets, processes and transmits scientific data back to Earth for further processing, helps researchers interpret results, etc.{lutz2011software}.

In medical research and diagnosis, imaging software plays a critical role to assist medical researchers, for instance, for early isolation of cancer cells. The main reason for low chance of survival from cancer is mainly due to late detection of cancer cells in the body. This makes a diagnosis of cancer to be a time critical task and early identification of cancer implies curability of a disease and a higher chance of survival {wagner2004challenges}. Especially on the early stages, it is not straight forward to determine which cells are likely to develop a cancer. For this reason, medical scientists use different types of software to identify cancer cell or to decide weather a tumor is malignant or not. Using a software, they could perform various kinds of analysis and processing on imageries obtained from scans such as MRI or CT Scan {al2012lung}. An example of software that is used for cancer imaging research is DMRI. Such software is extensively used by many researchers, more than 75,000 downloads every year {norton2017slicerdmri}. Therefore, it is clear that software plays a critical role in medicine, to diagnose diseases and ultimately to save life.

Software plays an important role in power system planning and operation as well. One of the major activities in power system operation is contingency analysis. During contingency analysis, engineers determine violations of power grid operation conditions, such as overloading, which might occur when outage of a transmission line or a power generation unit happens. Contingency analysis helps to understand power system behavior after outages and gives an opportunity to take preventative actions {mishra2012contingency}. Power grids are extremely complex and such kind of analysis tasks are unimaginable with out a use of software. An example of software that is used to perform contingency analysis in the power system operation is Power World software {powerworld.com}.

Though it is not possible to mention the role and use of software across all areas of science and research, the above examples serve to be a good sample to see how ubiquitous the impact of software is almost in all research areas.

## The role of software in research breakthroughs

The impact of software is more pronounced and easy to observe when scientists achieve ground breaking results. The use of software enabled scientists to produce better scientific discoveries and achieve research breakthroughs {goble2014better}.

One good example of research breakthrough made, because of use of software in a scientific investigation, is creation of the very first visual representation of a black hole using an open source software NumFOCUS {event2019first}. To observe a black hole that is 55 million light years away, it would have required to build a huge telescope of size of planet earth. But instead of building one giant telescope, which is not possible any way, hundreds of scientists spent decades of years creating a global network of telescopes, known as Event Horizon Telescope (EHT) {enwiki:1052167868}, synchronized precisely using atomic clocks.

The EHT gathered a huge amount of data for years. However there was a lot of noise in the collected data because the EHT was a network non-similar telescopes. In addition, the radio signals were coming through attenuated due to atmospheric effects like water vapor, clouds, turbulence … etc. { <https://numfocus.org/case-studies/first-photograph-black-hole> }.

Therefore the scientists had to use various algorithms and data analysis pipelines. The resulting image from various data processing was compared to ensure the integrity of the result. This huge scientific breakthrough in a space research, can be attributed to the use of powerful data processing software.

## Summary

In this chapter, it has been explained how important and pervasive software in a scientific research is and the impact overall. The next chapter focuses to identify main categories of software usage purposes from the literature and software ontologies.

# Software usage purpose

## Introduction

In scientific investigations broad range of software is being employed for various purposes. In terms of size, software ranges from simple script to extremely complex one with millions of lines of code. In terms task, a software can be used for execution of rudimentary tasks to computation of extremely complex ones. Typical examples of purpose of software use for scientific investigation are simulation, modelling, data analysis, etc. {goble2014better}.

To be able to automatically identify, from context, for what purpose a software is used in a scientific paper, a classifier algorithm has to be trained on a manually annotated dataset that indicate software usage purpose. The SoMeSci data set already has annotations about type of software, type of software mentions, etc. and only require extension with software purpose annotation so that it can be used for training a software usage purpose classifier. However, a comprehensive list of potential software usage purposes has to be identified before hand. To enumerate possible software purposes of usage, three things have been done in this thesis. These are:

1. Analysis of literatures
2. Analysis of software ontologies
3. Analysis of Sci-Crunch repository

After identifying a list of potential software usage purposes, the list has been consolidated further to narrow down the list to a more comprehensive list of software usage purposes for convenience during annotation of data set.

This section elaborates the analysis procedure of a list of resources mentioned above to identify possible software usage purposes.

## Analysis of literature

In a research, scientists follow scientific method to discover knowledge. Typically, scientists begin with a question and attempt to answer questions through a research and propose hypothetical answers for their questions. Then, they test the proposed hypothesis by conducting various experiments. Although all scientists do not follow the exact same procedure, the over all idea remains the same {enwiki:1061107378- Scientific method}. This is where a software use comes into play, aid scientists during their experiment.

Therefore, the analysis of literatures when looking for software usage purpose is aimed at answering, from a given context, “*for what purpose scientists are using a software ?”* in their experiments.

Accordingly, some key words that reflect potential software usage purposes have been identified from the literature and listed on the following table:

|  |  |
| --- | --- |
| * Comparison of experimental groups * Quantification * Measurements * Analysis * Mapping * Correction of mapping * Generate scaffolds * Generate trees * Search sequences * Map * Predict gene structure * Align gene * Filter * Evaluate * Select * Optimise * Classify | * Statistical analysis * Data analysis * Densitometric analysis * Voxel-based Analysis * Cross-sectional ROI analysis * Gene analysis * Gene assembling * Construct contigs * Fill gaps * Generate assembly * Calculate or determine a value * Draw heat map * Validate * Annotation * Fit or train a model * Sketch * Identify |

The list of key words in the above table is used to delineate possible software usage purposes, however, it is intractable to enumerate all possible software purposes by manually reading through unlimited number of publications. To augment results obtained from the analysis of literature, various software ontologies and repositories like, Sci-Crunch, have been analyzed as presented in the following section.

## Analysis of software ontologies

Ontologies are controlled vocabularies that provide formal naming and definition of properties and relation between concepts, entities, data etc. Ontologies are specialized to a specific subject matter and every academic discipline creates ontologies to organize data into useful knowledge {enwiki:1060388948}.

Effective knowledge representation begins with analysis of ontologies with in the domain of interest {chandrasekaran1999ontologies}. Accordingly, analysis of software ontologies have been done to find out possible software usage purposes. The software ontologies, that has been analyzed on this project are: WikiData, SWO (the software ontology), and OntoSoft.

### WikiData

Wikidata is a multilingual knowledge graph that is curated collaboratively by a Wikimedia community and serves as a freely available common source of structured data for everyone {enwiki:1060114687, enwiki:1060408581}.

Wikidata was created by Wikimedia foundation mainly to store meta data that can be used for other Wikimedia projects such as Wikipedia. Interestingly, wikidata is allowed to contain inconsistent and contradicting facts in order to embrace the diversity of knowledge about a given entity {vrandevcic2012wikidata}.

Although wikidata has a tremendous amount of data in it, there was no information that would indicate software usage purposes, rather information about software categories was found. Therefore, an indirect approach has been taken to list down possible software purposes from software categories by assuming each software category has essentially a software purpose associated to it.

Wikidata has a bunch of tools like, SPARQL end point, query builder, data visualization tools, etc. Thus a SPARQL end point has been utilized to query a list software and their potential categories in a format that supports network analysis, with edge and node. The SPARQL query used to retrieve software categories have been listed under *appendix A*. As a result, over 400 software categories have been found from the Query.

To find out potential relation between these categories and to select more general software categories, a network analysis has been done using Gephi software (version 0.9.2) (RRID:SCR\_004293){<http://gephi.org/>}. Using Gephi, clustering of related software categories and filtering has been made to identify a more generalized software categories. The procedure for network analysis has been described as follows:

1. First query result from the SPARQL terminal of wikidata has been downloaded in a csv file format, with a data structure that supports node and edge.
2. Then, the csv file has been opened with Gephi software as “undirected graph”. This renders a network graph with overlapping nodes and edges.
3. To unravel the overlapping nodes for visibility, the lay-out of the graph is then changed to “Fruchterman Reingold”.
4. To find out possible clusters from the network, from the list of statistical tools, “Modularity” has been run. Then nodes and edges has been partitioned using “Modularity class”.
5. Then to adjust size of nodes based on importance, node size ranking has been done with a “Degree” parameter with {minimum, maximum} size of {20, 80} respectively.
6. Then to select the most prominent nodes, among filter tool “Degree range” filter has been used. The Degree range filter estimated prominence of the nodes between values of {1, 60} where the maximum value indicates the most prominent node which corresponds to a more general software category.

Text

Description automatically generated

Text

Description automatically generated

Degree {1, 60}

Degree {3, 60}

Timeline, map

Description automatically generated

Diagram, map

Description automatically generated

Degree {5, 60}

Degree {7, 60}

According to the network analysis, major types of software categories (prominent nodes in the network graph) are:

* Application software
* Utility software
* Computer security software
* System software
* Client
* Programming tool
* Software Library
* Software framework
* Editor
* Science software.
* Graphics software
* Computer aided design software
* Mathematical software
* Communication software

. According to a manual analysis of wikidata, the above software categories are related to each other as well. Mathematical software, for instance, is subclass of science software and science software is subclass of application software.

By further analyzing the relation between the above software categories, overall the three main types of software categories are application software, system software and software component. A simplified version of hierarchical relation between software categories depicted on the graph below.

Diagram

Description automatically generated

#### Identifying software purpose

The main aim of software category analysis of wikidata was to find possible software usage purposes from each software category. It is simple to define a software purpose when the software is dedicated to carry out only a specific task.

One of the three main software categories is application software. According to Wikipedia, an application software is a computer program that is designed to carry out a specific task other than operation of a computer and typically made for end-users {enwiki:1060918552}. Most of research software can be considered as an application software, since they are used for a specific purpose.

However it is also worth mentioning that, there are two types of application software: horizontal (market) application software and vertical (market) application software. A horizontal (market) software is a kind of application software that is more generic, used in wide range of industries, and lack very specific purpose { enwiki:1034388659}. Examples of such types of software are word processors, spreadsheets, calendar applications, etc. On the other hand there are Vertical (market) application software whose purpose is to address needs of a specific niche in a business, research, or even a specific department within an organization {enwiki:879502666}.

Since purpose of software is of interest for this project, emphasis has been given only to application software only which imply a specific purpose of use in research papers. Accordingly, sub-categories of application software with their respective purpose from Wikipedia and internet resources have been gathered and summarized on the following table.

|  |  |  |
| --- | --- | --- |
| Types of Application software  (sample) | Software purpose | Examples |
| Remote sensing software, | * Data collection * information gathering | * Google earth, * OpenEV * ENVI |
| Econometrics software | * Data Analysis | * Stata * R * SATA * SPSS |
| Network simulator | * Simulation | * OPNET * NetSim * GloMoSim |
| IDE , text editors | * Programming * text editing | * Visual Studio, NetBeans, * Atom, Sublime, Vim … etc. |
| Genealogy software | * Record data, * Organize & publish data | * Family Tree builder * Legacy |
| computer-aided design software | * Modelling * Analysis & optimization | * AC3D, SolidWorks * AutoCAD, CATIA, … etc. |
| * Science software, Bioinformatics software, mathematical software, chemistry software,   astronomy software | * Simulation * Modelling * Data Analysis * Visualization * Calculation |  |
| database application | * Retrive, Insert, modify, delate data |  |
| graphics software , animation software | * 3D modelling, * visualization | * 3D computer graphics software, |

In summary, from the analysis of software categories of wikidata ontology, the following list of software usage purposes have been identified:

* Data recording or collection
* Data Analysis
* Visualization
* Simulation
* Modelling
* Programming.

### The software ontology (SWO)

The software ontology (SWO), particularly describes software used, for preparation and maintenance of data, within fields of computational biology and bioinformatics. The SWO was primarily developed to improve reproducibility by providing detailed description about software used for biomedical investigations {malone2014software}.

SWO was found on ontology search (OLS) website and was examined for possible software purposes. Unlike wikidata, a list of possible software purpose were found directly in “browse terms” section of the SWO website. To navigate to the list of key words that suggest potential software purpose one can follow the following steps: “Browse terms”> “entity“>”occurrent”> “planned” >”planned process”. The software usage purpose in the SWO has been presented in to two main groups as “data transformation” and “data visualization”. Under data transformation, 40 sub-types of potential software purposes are listed.

Graphical user interface, website

Description automatically generated

After manual analysis and grouping of purpose of use of software, more general classes of software usage purposes has been summarized on the table below:

|  |  |
| --- | --- |
| * Data transformation * Annotation * Text editing * Modelling * Curve fitting * Simulation * Query and retrieval | * Calculation * Analysis * Data visualization * File rendering * Matrix manipulation * Data mining task * Clustering task |

### OntoSoft

Onosoft is a software registry framework that stores important metadata about software to foster reuse and sharing of software among scientific community. The ontology provides descriptions about a software that would help scientists to identify, understand, execute, and do research with a software. Moreover, it helps scientists get information about update and support for the software.

These descriptions are visualized in a 6 dimensional pie-chart, with each slice indicating the completeness of the description. Particularly, Ontosoft focuses on the geoscience because software resources are not being shared adequately in that field {gil2015ontosoft}.

Graphical user interface

Description automatically generated

The type of information provided in each dimension of description entries are summarized in the table below:

|  |  |
| --- | --- |
| Dimension | Description |
| Identify | * Name of software, abbreviation of the software, etc. |
| Understand | * Creator of the software, publisher of the software, * *domain specific key words* |
| Execute | * URL for downloading the software, license, system requirements …etc. |
| Do Research | * Input / output file formats, preferred citation information, …etc. |
| Get support | * Contact details, possible support included, etc. |
| Update | * Version, developer community, software development process , maintenance, etc. |

From the set of information provided among the 6 dimensions of the Ontosoft, particularly the “understand” dimension has nearly 400 domain specific key words that would potentially indicate software usage purposes. Therefore, those domain specific key words has been retrieved, analyzed and condensed into a more general software purposes.

Sample list of domain specific key-words that would potentially indicate a software usage purpose is listed on the following table.

|  |  |
| --- | --- |
| Domain Key-words | |
| * Data manipulation * Data Mining * Image processing * Machine learning * Simulation- optimization * Network analysis | * Numerical model * Numerical simulation * Thermal model * Integrated modeling * Interactive visualization * Wind wave estimation |

## Analysis of Sci-Crunch repository

The other resource analyzed in addition to software ontologies is to list down software usage purposes is Sci-crunch repository. Sci-crunch is a data portal that searches through hundreds of community databases, aggregates information resources to create a large collection of data and tools available for access at a single spot {grethe2016scicrunch}.

To identify possible software usage purposes, the Sci-crunch repository has been analyzed as follows. On the registry section of the sci-crunch home page, there is a pie chart indicating different types of resources. A software resource, with 7,155 different types of software resources has been selected from the pie chart. From there, top 200 types of software resources have been identified using the site’s built in word-cloud generator.

Text

Description automatically generated

After a manual analysis of the 200 of software types, generated from the word cloud, important software types that indicate possible software usage purpose has been identified. Sample of software types and their corresponding usage purpose is shown on the table below:

|  |  |
| --- | --- |
| Type of software | Purpose |
| * Data acquisition software * Image acquisition software | * Data collection * Data recording |
| * Data Analysis software * Image analysis software * Sequence Analysis software * Network analysis software * text-mining software * signal processing software | * Data Analysis |
| * Data Visualization software * 3D visualization software | * Data visualization |
| * Simulation software | * Simulation |
| * Alignment software , Image reconstruction software | * Data pre-processing or post processing |
| * Rendering software | * Modelling and graphics |
| * Code testing framework | * Programming |

## Types of software usage purposes

Based on a through analysis of scientific literatures in **SoMeSci** dataset, software ontologies and the sci-crunch repository, overall 8 main types of software usage purpose have been identified. These are:

1. Data Collection
2. Data pre-processing
3. Data Analysis
4. Data visualization
5. Simulation
6. Stimulation
7. Modelling
8. Programing

The overview of work flow process followed to identify these software usage purpose is summarized in the picture shown below.

Diagram

Description automatically generated

To establish a clear boundary and avoid ambiguity during the annotation process of software usage statements, in SoMeSci data set, each software usage purpose has been clearly defined based on literature in the next section as follows.

### Data collection

According to Wikipedia, data collection is a process of collecting, recording or measuring information on targeted variables which enables answering of questions. Regardless of the type of data, quantitative or qualitative, data collection is one of the most important steps in a scientific investigation {enwiki:1049936190}.

Scientists collect data for their research using various data collection software tools and gadgets. In one research, for instance, scientists used an Actigraph Reader Interface Unit (RIU-41A) with its software to measure the level of activity of more than 5000 children of age 12 to characterize the relation between physical activity and obesity {ness2007objectively, enwiki:1046731490}.

### Data pre-processing

Data collection processes produce inconsistent data and analysis of such noisy data might yield misleading results because of the “*garbage in, garbage out*” problem {enwiki:1059558941}. To avoid this problem, scientists usually carry out data pre-processing using a software. Data pre-processing generally refers to the addition, deletion, or transformation of raw data into a clean and tidy form to improve performance and reliability of analysis results, especially in data mining applications {kuhn2013data,rinnan2009data}.

Often times data pre-processing involves several steps such as data cleaning, integration, transformation, reduction, etc. {malley2016data}. Data cleaning, for instance, involves dropping of data, replacing, or imputation of missing values in order to improve performance of algorithms and reliability of analysis results, especially in data mining applications {enwiki:1051181443, enwiki:1056727993}.

In a scientific investigation, scientists usually carry out data pre-processing using a custom script or using an existing application software or programming library.

*From here add examples from* ***SoMeSci*** *data*

### Data Analysis

Data analysis refers to processing, transforming, modelling, etc. of data with *a goal of discovering a new insight* that would support conclusions or decision making. Data analysis involves diverse techniques with different names in various domains. {enwiki:1061024140}. In their research, scientists employ various software to carry out various tasks of data analysis such as curve fitting, spectral smoothing using a software {proctor1982data}.

### Data visualization

Data visualization refers to techniques that are used to communicate data or information effectively in the form of visual objects such as points, lines, bars, etc. in a graphic representation {enwiki:1059912747}.

### Simulation

Computer simulations mimic operation of real-world process or system using models that represent key-behaviors of the system. By varying variables of the simulation, predictions about behavior of systems can be made. Simulations have a wide range of application in scientific modelling of natural systems in physics, chemistry and biology {enwiki:1061669086}. Simulations are run to improve understanding of a problem (segal2008developing).

### Stimulation

Stimulation is the act of evoking the development of involuntary activity or response. Living organisms have sensory receptors that generate impulses that travel through nerve to the brain upon a reception of excitation by means of various agents, energy, collectively known as stimuli. Examples of sensory receptors in the human body are photoreceptors in the retina, touch receptors on the skin, chemical receptors in mouth, etc. {enwiki:976395276}.

In the scientific investigations, scientists use various mechanisms to provide a stimulation to their research object. One of the ways to provide stimulation is using a software. In neurological research, for example, scientists use various brain stimulation techniques and software to study neurological disorders. Deep Brain Stimulation (DBS), for instance, is one of brain stimulation techniques used to treat diseases like Parkinson’s, essential tremor, dystonia etc. {schermer2011ethical}.

### Modelling

Modelling refers to scientific activities that aim to facilitate understanding of a particular feature or phenomena in the world. It is a process of identifying and selecting relevant aspects of a situation or phenomenon under consideration. Different types of models with more specific aim exist. For instance, conceptual modelling provide better understanding, mathematical models help to quantify, computational models are used for simulation, etc. {enwiki:1051627717}.

Modelling is a broad term that refers to a wide range of activities. It might refer to 3D modeling and graphical representation of a real world physical objects like vehicles, buildings, …etc. using Computer Aided Design (CAD) software. For instance, some scientists use graphical modelling software, for instance for digitally documenting historical sites such as castles {el2007detailed}. On the other hand modelling can also refer to mathematical representation of a non physical abstract entity. In one research paper, for instance, the researchers were trying to model the occurrence of letters and word’s initials mathematically {pande2010mathematical}.

Regardless of the wide meaning and techniques of modelling, inherently all models serve to represent an object or a system to facilitate the representation, or understanding of particular feature or phenomena {enwiki:1058944086, enwiki:1051627717 }.

### Programing

Programming refers to the process of designing and building executable computer programs that performs a specific task. Computer programs are written in a human readable format mainly to automate execution of complex tasks and for solving problems {enwiki:1062649903}.

## Summary of software usage purposes and examples

|  |  |
| --- | --- |
| Software Usage Purpose | Examples |
| Data Collection | * Surveying * Data acquisition * Text extraction * Measurement * Data recording * Constructing an artificial data set * Importing a file or data of specific format into a software, etc. |
| Data pre-processing | * Data cleaning * Data encoding * Text editing * Error correction * Data normalization, calibration, data type conversion * Missing data handling, removing duplicates * Data transformation, data format conversion * Data reduction * Tabulating, merging data * File formatting * Aligning gene |
| Data Analysis | Sequence analysis  Data manipulation  Testing hypothesis  Data mining , clustering  Prediction  Quantification  Calculation, computation  Comparing, testing, searching,  Assessing / evaluating  Densitometric Analysis  Image analysis /processing  Mathematical analysis  Network Analysis  Numerical Analysis  Regression Analysis |
| Data Visualization | Creating figures  Plotting  Graph generation  Figure generation |
| Simulation | Flight simulation  Event simulation  Flood dynamics simulation  Numerical simulation  Simulation of vehicle schedule |
| Stimulation | * Stimulate behavior |
| Modelling | Scientific modelling  Mathematical modelling  Machine learning / Model fitting  Predicting a behavior  Estimating  Inference |
| Programming | * Implementation * Programming |

# Data set

## Introduction

Training and evaluation of automatic information extraction approaches requires availability of reliable ground truth data of sufficient size. Following a growth of interest for extraction of information about software tools from scientific publications labeled data sets with limited scope such as BioNerDs, SoftCite, SoSciSoSci have came into existence. More recently, SoMeSci data set, a more comprehensive corpus that covers a wide range of information about software tools has also been introduced {schindler2021somesci}.

This section describes the data set used in this project – SoMeSci, the extension process with software usage purpose annotations, issues observed during annotation, pre-processing of the data-set, analysis results of the data and transformation to a suitable format for training purpose.

## SoMeSci data set

SoMeSci data set contains high quality, hand annotated articles collated from PubMed Central (PMC). The articles and annotations included in the data set are summarized below.

### SoMeSci Articles

The corpus is composed of four group of files, namely *PLoS methods*, *PLoS sentences*, *PubMed full text* and *Creation sentences*. Facts about the articles in the SoMeSci corpus is summarized in the table below:

|  |  |
| --- | --- |
| SoMeSci parts | Description |
| *PLoS methods* | * 480 files * Contains only methods sections extracted from PLoS journal |
| *PubMed full text* | * 100 files * Randomly selected 100 full-articles from PMC Open Access |
| *PLoS sentences* | * 677 files * Contains sentences extracted from 677 PLoS articles * sentences contain software names. |
| *Creation sentences* | * 110 files * Out of 110 files, 50 are extracted from PMC OA * Out of 110 files, 60 are extracted from PLoS * Sentences contain statements that indicate creation of software. |
| *Total* | * 1367 files |

### SoMeSci Annotations

SoMeSci corpus has three main types of annotations that correspond to a type of information related with software tools. These annotations indicate the *type of software*, *type of mention* and *additional information* about the software as summarized on the table below:

|  |  |
| --- | --- |
| software information | Description |
| Type of software  (4 - types) | * Application * Plugin, * Operating System and * programming environment |
| Type of mention  (4 - types) | * describes the software’s appearance in the publications. * *Mention* – indicates software was just mentioned in the article * *Usage*- indicates software was used for some reason * *Creation* – indicates novel software is produced or introduced * *Deposition* – indicates deposition of new software in a repository. |
| Additional information  (9 -types) | * *Developer* * *Version* * *URL* * *Citation* * *Extension* * *Release* * *License* * *Abbreviation* * *Alternative name* |

## Annotation tool

The data set has been annotated using BRAT rapid annotation tool, v.1.3 RRID:SCR\_008769, in a Linux 20.4 environment. The annotation tool has been run in a local machine as a CGI application using a browser.

Graphical user interface, text, application

Description automatically generated

### Annotation of SoMeSci with software purpose labels

SoMeSci corpus has been extended with annotations of eight classes of purpose of software usage labels identified in the earlier section. Since using software for a particular purpose only refers to the usage of a software, only usage labels has been further labelled with software purpose. The figure below shows SoMeSci data set before and after software purpose annotations.



A picture containing text

Description automatically generated

### Assumptions in the annotation

For the sake of simplicity, certain types of software usages have been assigned the same class of software purpose annotation. For example, modelling might refer to graphical modelling of an object using CAD software or it might also refer to mathematical representation of a given problem. All such variants of modelling tasks have been assigned “modelling” as a label without differentiating specific variants.

### Challenges during Annotation

Annotations has been carried out in a such way by deciding on each context which software purpose annotation is more important or based on the general goal of the software usage. For example, FlexArray software on the figure below, has been annotated with software purpose analysis even though the same software was used for visualization purpose as well. This is because on this context analysis is more important than visualization and essentially visualization could also be interpreted as one kind of analysis. In addition, specific definition of each of software usage purposes has been also taken into account.



However, annotation of software usage statements was not often straightforward. This is because, in some instances, purpose of software usage appears to be ambiguous. For example a software used for counting or quantification can be assigned a software purpose label of data collection but it is also possible to treat those tasks like analysis. Therefore such cases annotation of purpose label might become subjective, might affect the Inter-rater-reliability agreements and the over all annotation quality.

The other challenge of annotation was difficulty arising from limited domain knowledge.

## Data Pre-processing

Pre-processing of the data set has been carried out to ensure the integrity of our data set before using it in the classifier. The data pre-processing tasks handled annotation errors, merging software-purpose labels with software-usage labels, transforming and splitting of data set.

### Handling annotation errors and missing annotations

As described in the above table, the four types of software mentions in the SoMeSci are *mention, usage, creation* and *deposition*. The main goal of annotating the data set was to assign corresponding *purpose* of software usage label to each instance of software *usage* but not for *mention*, *creation* and *deposition*. However, due to an error there were some instances of software *mention* that has been annotated with software purpose. In addition to this, there were also some instances of *usage,* that has not been annotated and intentionally skipped because of the purpose of software usage did not seem to be clear.

Therefore, all instances of wrong or missing annotations have been identified automatically to ensure the integrity of training data set. After identifying the list of files and instances of annotations with an error or skipped annotations, all errors have been rectified and skipped annotations has been handled. The python code for automatic identification of annotation errors has been listed under *appendix B*.

### Merging annotations

After handing all annotation errors and missing labels, annotations of software usage have been merged with annotations of software purpose mainly for two reasons. The first is to fix annotation error message that is displayed on the BRAT tool. The error message is displayed because more than one annotation per a token is not supported by the annotation tool.

The other reason for merging annotations is to take advantage of legacy code, ariclenizer, which will transform data format from BRAT’s stand-off format into IOB format which is desirable for training purpose. The python code for merging annotations has been listed on the *appendix C*.





### Transformation to IOB format

After merging software usage and purpose labels, transformation of data into IOB format{enwiki:1041803321} has been carried out using articlenizer (link to articlenizer). Picture below shows the data format before and after transformation.





### Data Splitting

After the data has been transformed into the IOB format, it has been further split into training, development and test set in 60:20:20 ratio.

## Analysis of Annotated Data

Analysis of cleaned SoMeSci data set has been carried out to find a deeper insight about the training data.

### Co-reference resolution of software entities

The base line for the analysis of the data set was to carry out disambiguation of software names. This was particularly important because there is large degree of variation in software names. Using list of software name with corresponding [URL](https://github.com/dave-s477/SoMeSci/blob/master/Linking/artifacts.json), software mention instances have been disambiguated from each other and all software name variations that refer to the same entity have been given the same name. Figure below shows name variations for MATLAB software, in which all instances resolve to the same URL i.e. entities referring to the same software. All variations of names has been replaced by the first “Matlab” in this case.

Text

Description automatically generated

### Analysis results

According to the analysis results, the top 3 software by number of software name mention count through out the list of articles in PubMed and PLoS data set are: PASW, GNU-R and STATA.

Chart

Description automatically generated

Data set analysis result from the perspective of purpose of software usage indicates that, the most common *purpose* of software usage are: *Analysis*, *Data pre-processing*, *Data collection* and *modelling* where as the least common are *simulation* and *stimulation*.

Chart, pie chart

Description automatically generated

Chart

Description automatically generated

The other insight form from the *software-type* perspective, is that the most commonly used type of software in the research articles in the data set is *Application* software.

Chart

Description automatically generated

When it comes to share of each purpose of software usage among the 4 types of software, the pattern once again clearly indictaes most of the time a software has been used for the purpose of *analysis* and *data collection* in all of the four software types.

A picture containing chart

Description automatically generated

Lastly, the most interesting insight that is important for the automatic classification task was determining: ” *for how many different purposes a given software have been used for* ?”. The analysis result reveals that from 657 unique software lists, a little over 3 out of 4 software have been used only for a single purpose. Over all almost 98% of software have been used for a purposes maximum of three. This indicates that most of the software have been used only for a specific purpose.

A picture containing shape

Description automatically generated

## Summary

This section has described the training data set, the annotation process and insights about the data. *One of the core results of the analysis result is the fact that most of the time, software has been used for a specific purpose*. The next section presents models that could be suitable for classification of purpose of software usage statements using the data set described above.

# Classifier models

## Introduction

Extraction of information about a software can be done following various approaches. In the past, rudimentary approaches such as searching for a term in paper, manual content analysis using human readers as well as rule based approaches have been employed (kruger2019literature). More recently a deep learning model have also been employed for automatic extraction of information about software such as mention types, software type, etc. {schindler2022role}.

Software is a real world entity and extraction of information about software can be approached as a named entity recognition problem. Named entity recognition tasks are one of classical sequence labeling tasks of NLP among others such as part-of-speech (POS) tagging and text chunking {akhundov2018sequence, he2020survey}.

Therefore, automatic classification of software usage purpose can be approached as a sequence labeling task in which a class label, from a fixed list of class labels, is assigned to each token in a sequence.

Various types of sequence labelling models can be broadly categorized as statistical machine learning based or neural network based models {he2020survey}. Classical machine learning based models for sequence labeling are Hidden Markov Models (HMM) {kupiec1992robust}, Maximum Entropy Markov Models (MEMM) {mccallum2000maximum}, and Conditional Random Fields (CRF) {lafferty2001conditional}. These models are statistical and fundamentally depend on manually crafted features, external resources such as gazetteers and fail to adapt to new domains {ma2016end}.

On the other hand, there are various neural network architectures and models that suit sequence labeling. Recurrent neural network architectures, based on LSTM, had been well established models for sequence labeling. Recently, transformer based neural network architectures, such as BERT, have pushed state-of-the-art performance further not only in sequence labeling tasks but for most NLP tasks {vaswani2017attention}.

This section presents and compares various types of sequence labeling models suitable for sequence labeling task of automatic classification of software usage purposes.

## Machine learning Models

### Hidden Markov models (HMMs)

One of machine learning models that are used for sequence labeling are Hidden Markov Models (HMMs). HMMs are based on Markov processes which describe a sequence of hidden finite states (Yi) in which a given state in a sequence depends only on the state prior to it {aggarwal2018machine, gagniuc2017markov}.

The Markov chain of hidden states Yi generate observations Xi based on its current stateusing a joint probability distribution P(Y,X). The joint probability P(Y,X) is a product of conditional probability P(X|Y) and a prior probability P(Y) which specifies the probability of the observation Yi and the probability of state Xi.

Chart

Description automatically generated with medium confidence

Figure 1: Hidden Markov model structure {aggarwal2018machine}

As the model transitions through hidden states of Yi, it generates an observation Xi which corresponds to a token in a sentence, hence HMMs are also referred to as generative models {aggarwal2018machine }.

The challenge with Hidden Markov Models is that, it could be difficult to define the joint probability for some real world applications where the number of hidden states are infinite {lafferty2001conditional}. In addition, Hidden Markov Models fail to capture long range dependencies because of their Markov assumption which considers only prior states {bulla2006application, wallach2004conditional}.

### Maximum Entropy Markov Models (MEMMs)

Unlike hidden Markov models which generate sequences of tokens based on hidden states, maximum entropy models are discriminative models i.e., they directly model the probability of each label Yi based on the current observation Xi and the prior hidden state Yi-1 {mccallum2000maximum}.

Chart

Description automatically generated

The biggest drawback with MEMMs and other discriminative directed graphical models based on Markov, tend to be biased in favor of states with fewer successor states {lafferty2001conditional}.

### Linear Conditional Random Fields (CRFs)

CRFs are similar with MEMMs in that both are probabilistic and discriminative models that can perform sequence labeling based on conditional probability p(Y|X) unlike joint probability of HMMs {wallach2004conditional}. A sequence of tokens in a sentence corresponds to X and Y refers to a sequence of class labels.

A picture containing chart

Description automatically generated

Figure 2: Linear CRF model {aggarwal2018machine}

CRFs differ from MEMMs in that they are undirected graphs, i.e., the inference of Yi depends on all the labels occurring before it as well as after it. This makes training of CRFs computationally expensive especially when a wider window of tokens is chosen to handle long range dependencies {aggarwal2018machine}.

## Deep Learning Models

Deep learning based models are state-of-the-art for many machine learning and NLP tasks, including sequence labeling. There are various types of deep learning architectures that suit for sequence labeling. Popular neural network architectures for sequence labeling tasks are recurrent neural networks and transformer-encoder.

Recurrent neural network based models such as LSTM, Bi-LSTM, Bi-LSTM-CRF had been state-of-the-art sequence labeling models, before the inception of transformer-encoder based models like BERT which produced a new state-of-the-art performance for many NLP tasks including sequence labeling.

### Long Short-Term Memory (LSTM)

Recurrent Neural Networks (RNNs) are specific type of neural network architectures that take a sequence of inputs features to yield a sequence of labels. However, plain RNNs fail to capture long-range dependencies because of vanishing and exploding gradient problems {pascanu2013difficulty, fischer2018deep}.

Fortunately, other variants of RNNs such as LSTM networks are capable overcoming unstable gradient problems {akhundov2018sequence, lample2016neural, ma2016end}. This is because, unlike RNNs, LSTM networks have memory cells that are composed of three gateways which determine what amount of information to forget at a given instant of time {ma2016end}.

LSTM networks are typically composed of input layer, hidden layer(s), and output layer. Number of inputs to the input layer corresponds to input features and the number of outputs on the output layer corresponds to number of classes of the classification task. The memory cells of LSTM are found in the hidden layers of the network {fischer2018deep}. A memory cell structure is depicted on the figure below.

Diagram

Description automatically generated with medium confidence

img source { fischer2018deep }

### Bi-Long Short-Term Memory (BLSTM)

For sequence labeling tasks, it is desirable to access left as well as right input features to capture context information. However, LSTM models are capable of remembering only past (left) sequence of features.

To solve this problem, Bi-LSTM combines two LSTM networks where the first LSTM unit captures input features in a forward direction while the other captures the information in the reverse direction. Finally the two input features get concatenated to create a feature representation that encapsulates context information in both directions {ma2016end}.

### Bi-LSTM-CRF

In sequence labeling, it is also important to capture correlations among adjacent ***input features at a sentence level***. Since CRFs are capable of incorporating correlations among neighboring input ***features in a seqence***, outputs of Bi-LSTM can be fed into CRF to create Bi-LSTM-CRF model which is aware of correlation between neighboring features in addition to context in both directions {ma2016end}.

The use of past and future inputs using Bi-LSTM combined with CRF to capture correlations among features at a sentence level boosts sequence labeling accuracy. Because of this the Bi-LSTM-CRF model produce state-of-the-art results among LSTM based models. Evidently, an F1 score of 97.55%, 94.46% and 88.83% was achieved on classical sequence labeling tasks of POS, chunking (CoNLL 2000) and NER (CoNLL-2003) respectively {huang2015bidirectional}.

Diagram, schematic

Description automatically generated

## Transformer Encoder Models

Although Bi-LSTM-CRF models have superior performance for sequence labeling over other LSTM based models, they have inherent weaknesses. Firstly, Bi-LSTM-CRFs are not truly bi-directional context readers, rather pseudo-directionality is achieved learning right and left contexts separately and concatenating them. Because of this the true context of words could be lost slightly. Secondly, Bi-LSTM-CRFs are very slow to train because of their sequential nature which does not allow to take advantage of parallelization using modern GPUs.

Deep learning models based on transformer architectures, like BERT, have the ability to read past and future context simultaneously. In addition, such models can train faster because of their ability for parallelization using modern GPUs because they are not sequential unlike LSTM networks.

### BERT

BERT is a model based on *transfer-learning*, using a pre-trained neural network on known task and then fine-tuning the pre-trained model for down-stream purpose specific applications.

BERT:

Designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers.

BERT model can be finetuned with just one additional output layer to create state-of-the-art models.

BERT obtains new state-of-the-art results on eleven natural language processing tasks.

BERT is pre-trained general language representation (language model)

Language model ***pre-training*** has been shown to be effective for improving many natural language processing tasks such as:

* Sentence level tasks
  + natural language inference
  + paraphrasing
* token level tasks
  + named entity recognition
  + question answering

How do pre-train language models? Ans: 2 Strategies

* *feature-based* 
  + example ELMO-using pre-trained representations
  + task specific architecture
* *fine-tuning like GPT*
  + *minimally task specific*
  + *trained on downstream tasks, simply fine tuning pre-trained parameters*

what makes BERT different from ELMO, GPT ?

* ELMO and GPT -are unidirectional, could perform bad for token level tasks like question answering.
* BERT solves unidirectionality problem using **MASKED LANGUAGE MODEL**
* masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based on context.

Why is BERT cool ?

* pre-trained representations reduce the need for many heavily-engineered task specific architectures.
* BERT advances the state of the art for eleven NLP tasks

Deep learning models based on transformers learn from data automatically using self-attention technique. An example of transformer model that accomplished state-of-the-art performance in many NLP tasks including sequence labeling is BERT. BERT is pre-trained on a corpus containing more than 3.3 billion words from books and Wikipedia {devlin2018bert}. One of the great advantages of BERT is that

, such as BERT, achieved a new state-of-the-art performance for NLP tasks including sequence labeling {ezen2020comparison}. Transformers use self-attention technique to focus on a small part of data at a time.

State-of-the-art deep learning model labeling are based on language representation models such as BERT.

In addition, there are also transformer based models such as BERT, Sci-BERT and Bio-BERT.

state-of-the-art Bi-LSTM-CRF {huang2015bidirectional}. There are also transformer based models such as BERT, Sci-BERT and Bio-BERT.

Due to its performance, BERT model has been employed for classifying software usage purpose from a sequence of tokens in a sentence. The classifier model implemented as a multi-classifier with 4-layered network where each classifier layer is dedicated to classifying software, software-type, mention-type and software-purpose in respective order. Fully connected model is shown in the figure below.

Graphical user interface

Description automatically generated

### Language models

To generate features for each word in a sentence, word embeddings like Sci-BERT and Bio-BERT have been used.

Sci-BERT is pretrained on multi-domain random collection, over 1.14M, of scientific publications and capable of generating contextualized embeddings for each token in a sequence. Most of the scientific papers used to train Sci-BERT are from biomedical domain (82%) and the rest from computer science domain {beltagy2019scibert}.

Bio-BERT is as well a contextualized word embedding trained on biomedical corpora. The scientific publications in the Bio-BERT corpora are made of abstract section of PubMed journal and full text articles from Pub Med Central (PMC){ li2019fine}.

# Model Training and Optimization

## Introduction

The Bi-LSTM-CRF multi-class classifier model has been trained and tested in a various training scenarios to determine conditions that leads to improved performance of classifier.

The training scenarios considered include the impact of data-split, inclusion/exclusion of some parts of SoMeSci data sets, and context information.

Furthermore, the model’s performance has also been investigated by truncating the 4-layred multi-level Bi-LSTM-CRF classifier into 2 layers after removing mention-type and software-type classifiers.

In addition, the impact of word embedding on classifier’s performance has also been investigated using different variants of pre-trained models of embeddings.

Finally, results from all training scenarios have been discussed and best performing model has been selected based on the results of investigation.

## Data set split optimization

The training data set, SoMeSci, has been split into train, test and development set with 60, 20, 20 ratio respectively. To ensure the distribution of enough samples of class labels, for all 8 types software usage purpose class labels, the data set has been iteratively split until each of software usage purpose class label lies within +/- 5% range for all train, test and development datasets.

Initially the iterative data splitting was carried out separately for PLoS methods and PubMed-full text. However, due to lack of enough class labels in PubMed-full text, the iterative splitting did not converge. For this reason PLoS and PubMed articles have been combined to make the splitting converge.

The most optimal data split obtained from the iterative splitting is shown on the figure below, where there is still class imbalance for stimulation in the development data set. Finally, few articles from the test set have been manually moved into the development set to make the data split comply 100% within +/- 5%.

Text

Description automatically generated

Part of a python code for optimizing the data split has been listed on appendix D. The notebook for data set splitting and optimization can be found on a git-hub repository.

{ https://github.com/BeTKH/SoMeNLP/tree/2layerClassfier/bin}

## Model training with inclusion/exclusion part of data

As described in section 4.2.1 before, the SoMeSci dataset is composed of 4 different sets of articles: *PLoS-methods*, *PubMed-full text*, *PLoS-sentences* and *creation-sentences*. Since only articles in the *PLoS-methods* and *PubMed-full* are annotated with software usage purpose labels, it was desired to evaluate weather including *PLoS* and *creation* would result in improved performance of the software usage purpose classifier.

The results of evaluation indicates that including *PLoS-sentences* and *creation-sentences* in the training dataset, definitely improved model performance for classification of software. However, the software purpose classifier’s performance is not improved, rather deteriorated. This makes sense given articles in those sentences do not have software purpose labels.

Evidence: software purpose classification degrades with including sentences (pic-1) and software classification improves with inclusion of sentences data set(pic-2).

* SciBERT---contxt2Sentcs\_wo---WithSent---2,2\08-03-2022\_10-49-47 --- grey
* SciBERT---contxt2Sentcs\_wo---2,2\05-03-2022\_02-17-25 --- blue

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Since the main goal of this project is to maximize software usage purpose classifer’s performance, for subsequent steps of analysis, the software usage purpose classifier has been trained only with datasets of PLoS-methods and PubMed-full text by excluding creation /PLoS sentences.

## The impact of context on classifier’s performance

Scientific papers like any other well written documents, have a sequential structure that form abstraction at various levels such as sentence, paragraphs, sections, chapters, etc. These levels of abstractions often determine the meaning of words because each level of abstraction or context conveys a valuable information {ghosh2016contextual}.

Likewise, contextual information helps to determine the correct purpose of use of a software. Therefore for each sentence in a text of scientific articles of training data set, various range of neighboring sentences have been incorporated to provide a contextual information. Typically 2 adjacent sentences before a sentence, after a sentence, and before & after a sentence have been considered. Further more, a context as broad as the whole paragraph has also been considered. The python code for reading neighboring sentences for context has been listed on the *appendix E*.

### Left context vs right context sentence

Left context refers to sentences prior to a given sentence with in a paragraph and right context refers to sentences that lie right after a given sentence.

Classifier’s performance evaluation indicate that there is a slight improvement in model’s performance for right context as compared to the left context. However, when 2 sentences to left as well as right are taken into account for context, there is no significant gain on the model’s performance for both software and purpose of usage.

However, due to the nature of randomness in the training of the classifier model, each instance of training round tends to give slightly different result. Because of this, both left and right context of two sentences have been considered for subsequent steps of analysis of the model despite the apparent significance of right context over the left.

SciBERT---contxt2Sentcs\_wo---0,2\06-03-2022\_22-11-29 --blue

SciBERT---contxt2Sentcs\_wo---2,0\06-03-2022\_14-07-03-- orange

SciBERT---contxt2Sentcs\_wo---2,2\05-03-2022\_12-17-17--red

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

Theoretically consideration of broader context, more than 2 adjacent sentences, is supposed to improve the classifier’s performance. However, the result of the analysis reveals that a context as big as the whole paragraph did not contribute to the model’s classification performance. Rather, a smaller window of context such as 2 sentences left and right appears to have a better performance.

pictures: SciBERT---contxt2Sentcs\_wo---2,2\05-03-2022\_12-17-17 –red

SciBERT---contxt2Sentcs\_wo---paragrapgh\05-03-2022\_16-32-32 -- blue

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

### Context outside a paragraph

The other scenario considered was evaluation of model’s performance when context is not limited to a paragraph. The results indicate that, classifier’s performance degraded when a context is not limited within a paragraph. This agrees with the fact that each paragraph of a scientific publication conveys a specific information and a contextual information outside a paragraph might not be useful for the classifier. The classifier model which considers classifier context outside a paragraph has been listed on “contxt\_out\_parg” branch of SoMeNLP project.

Pictures: SciBERT---contxt\_out\_parg----2,2\07-03-2022\_17-35-19 --- blue

SciBERT---contxt2Sentcs\_wo---2,2\05-03-2022\_12-17-17--- red

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

## Classification with 2-layers

Overall the 4-layred Bi-LSTM-CRF multi-class classifier’s performance has been poor regardless of consideration of various factors discussed above. Because of this, the model has been simplified into a two layered Bi-LSTM-CRF model by removing software-type and mention-type classifiers.

Truncating the model into two layers helped to evaluate the software purpose classifiers performance exclusively by removing the effects of intermediate layers of mention-type and software-tape classifiers.

Graphical user interface

Description automatically generated

Figure 3: The 2-layred Bi-LSTM-CRF classifier model

Evaluation of the 2-layered classifier model reveals that overall software purpose classification performance has shown small improvement compared to the original 4-layered Bi-LSTM-CRF model, where as the software classifier’s performance does not indicate any performance improvement.

Pic: SciBERT---contxt2Sentcs\_wo---2,2\05-03-2022\_12-17-17--green

SciBERT----2layerClassfier---2,2\14-03-2022\_23-48-46 ---blue

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

## Model training Bio-BERT vs Sci-BERT

Even though both Bio-BERT and Sci-BERT give a contextualized representation of a word in a sentence, representations of a word would differ due to the inherent difference of corpora used for pre-trained models of Bio-BERT and Sci-BERT { beltagy2019scibert , li2019fine}. For this reason, classifier’s performance has been evaluated for both Sci-BERT and Bio-BERT models.

The results of evaluation indicate that the classifier models performed slightly better when using Bio-BERT large embedding compared to Sci-BERT. However, the model performed better with Sci-BERT compared to the Bio-BERT small embedding.

Pics: BioBERT---2layerClassfier---2,2\12-03-2022\_00-26-04 (large model --orange)

BioBERT-base---2layerClassfier---2,2\12-03-2022\_11-01-55 (dark-blue)

SciBERT---2layerClassfier---2,2\10-03-2022\_22-14-39 (light-blue)

Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

## Observations and summary of results

Performance evaluation of the Bi-LSTM-CRF multi-class classifier model, in various training scenarios, reveals some interesting insights.

Firstly, the analysis of the model performance results indicate that, the composition of the data set has significant impact on the models performance. As stated before, when the classifier model is fed with data set that lacks software purpose annotations in creation and plos sentences, the models performance decreased.

Secondly, it has been observed that various levels of context can affect models performance. As presented earlier, when neighboring sentences in a scientific articles of SoMeSci data set is considered it has been clearly observed that models performance has shown improvement. However, the analysis also reveals that unbounded context information such as a context outside a given paragraph is not useful to the model’s performance, rather it will degrade the models performance. Further more, it has been observed that a context of sentences on the left and right might not be equally important. The analysis results indicated that a context of sentences from the right were observed to be more useful to the models performance that the left.

Thirdly, evaluation of the classifier’s performance by truncating the intermediate layers of classifiers indicated a marginal improvement of software purpose classifier which lies at the end of the 4-layered chain of classifiers but there was no tangible performance gain was observed for the software classifier. This might indicate that, probably there flawed classifications of intermediate layers might have an impact on the software purpose classifier.

Further, it was also observed that the use of various types of embeddings like Bio-BERT and Sci-BERT impacted the classifier’s performance. The use of larger variants of embedding models was observed to improve classifiers performance.

Ultimately, it is reasonable to conclude that the study of all training scenarios and consideration of factors does not show a very large scale performance gain or loss. This might be because a constraint on the amount of labeled data set and unavailability of larger size of training data set.

### Summary of results for classification performance

Best performing software purpose classifier model, after consideration of different training scenarios, is a 2-layered software and software purpose classifier model which is trained with Bio-BERT (large) model. The classifier’s performance has been summarized for software purpose as well as software on the following table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Software Usage  Purposes | F-Score | | Precision | | Recall | |
| Dev. | Test | Dev | Test | Dev. | Test |
| Analysis | 0.72 | 0.73 | 0.72 | 0.69 | 0.72 | 0.76 |
| Data Collection | 0.30 | 0.47 | 0.29 | 0.49 | 0.31 | 0.45 |
| Data Pre-processing | 0.55 | 0.64 | 0.54 | 0.60 | 0.57 | 0.69 |
| Modelling | 0.61 | 0.50 | 0.50 | 0.52 | 0.77 | 0.48 |
| Programming | 0.41 | 0.42 | 0.46 | 0.47 | 0.37 | 0.37 |
| Simulation | 0.53 | 0.35 | 0.87 | 0.40 | 0.38 | 0.32 |
| Stimulation | 0.57 | 0.51 | 0.54 | 0.41 | 0.60 | 0.68 |
| Visualization | 0.66 | 0.66 | 0.72 | 0.63 | 0.61 | 0.70 |
| Total | 0.62 | 0.64 | 0.62 | 0.62 | 0.63 | 0.67 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Software  Usage | F-Score | | Precision | | Recall | |
| Dev. | Test | Dev | Test | Dev. | Test |
| Abbreviation | 0.86 | 0.87 | 0.88 | 0.89 | 0.85 | 0.84 |
| Alternative name | 0.76 | 0.44 | 0.72 | 0.35 | 0.80 | 0.61 |
| Application | 0.86 | 0.88 | 0.85 | 0.85 | 0.87 | 0.91 |
| Citation | 0.89 | 0.88 | 0.88 | 0.87 | 0.90 | 0.89 |
| Developer | 0.84 | 0.93 | 0.78 | 0.90 | 0.90 | 0.96 |
| Extension | 0.74 | 0.86 | 0.65 | 0.99 | 0.85 | 0.76 |
| License | 0.86 | 0.96 | 0.84 | 0.93 | 0.89 | 1.0 |
| Release | 0.80 | 0.68 | 0.71 | 0.58 | 0.93 | 0.83 |
| URL | 0.89 | 0.92 | 0.83 | 0.88 | 0.96 | 0.97 |
| Version | 0.97 | 0.95 | 0.96 | 0.92 | 0.98 | 0.97 |
| Total | 0.88 | 0.89 | 0.86 | 0.86 | 0.90 | 0.92 |

# Conclusion and Future work

## Introduction

From the review of literature and analysis of related works, it has been observed that software tools have been playing a critical role in science and research and will continue to be an important asset. Following this importance, it has been observed that there was a lot of effort undertaken to identify software tools from scientific papers, to give credits to the software developers and so on.

To assist the goal of mining information about a software various approaches ranging form manual analysis to automatic have been applied in the past. In addition various data sets has also been put forward to assist automatic extraction of information about software tools.

This project has also contributed to that effort in a matter one drop into the ocean by annotating purpose of software usages in the SoMeSci data set.

## Conclusion

## Future work and limitations