Contents

[1 Introduction 3](#_Toc95681466)

[1.1 Scope 4](#_Toc95681467)

[1.2 Objectives of the research 4](#_Toc95681468)

[1.3 Overview of the report 4](#_Toc95681469)

[1.4 Summary 5](#_Toc95681470)

[2 The role of Software in Scientific research 6](#_Toc95681471)

[2.1 Introduction 6](#_Toc95681472)

[2.2 General roles of software in a research 7](#_Toc95681473)

[2.3 Domain specific examples 8](#_Toc95681474)

[2.4 The role of software in research breakthroughs 9](#_Toc95681475)

[2.5 Summary 9](#_Toc95681476)

[3 Software usage purpose 10](#_Toc95681477)

[3.1 Introduction 10](#_Toc95681478)

[3.2 Analysis of literatures 10](#_Toc95681479)

[3.3 Analysis of software ontologies 11](#_Toc95681480)

[3.3.1 WikiData 11](#_Toc95681481)

[3.3.2 The software ontology (SWO) 15](#_Toc95681482)

[3.3.3 OntoSoft 16](#_Toc95681483)

[3.4 Analysis of Sci-Crunch repository 18](#_Toc95681484)

[3.5 Types of software usage purposes 19](#_Toc95681485)

[3.5.1 Data collection 19](#_Toc95681486)

[3.5.2 Data pre-processing 20](#_Toc95681487)

[3.5.3 Data Analysis 20](#_Toc95681488)

[3.5.4 Data visualization 20](#_Toc95681489)

[3.5.5 Simulation 20](#_Toc95681490)

[3.5.6 Stimulation 21](#_Toc95681491)

[3.5.7 Modelling 21](#_Toc95681492)

[3.5.8 Programing 21](#_Toc95681493)

[3.6 Summary of scope definition for software purposes 22](#_Toc95681494)

[4 Data set 23](#_Toc95681495)

[4.1 Introduction 23](#_Toc95681496)

[4.2 SoMeSci data set 23](#_Toc95681497)

[4.2.1 SoMeSci Articles 23](#_Toc95681498)

[4.2.2 SoMeSci Annotations 24](#_Toc95681499)

[4.2.3 Annotation tool 24](#_Toc95681500)

[4.2.4 Extension of SoMeSci 25](#_Toc95681501)

[4.2.5 Challenges during Annotation 25](#_Toc95681502)

[4.3 Data Pre-processing 26](#_Toc95681503)

[4.3.1 Handling missing annotations and annotation errors 26](#_Toc95681504)

[4.3.2 Merging annotations 26](#_Toc95681505)

[4.3.3 Transformation of Data 26](#_Toc95681506)

[4.3.4 Data Splitting 27](#_Toc95681507)

[4.4 Analysis of Annotated Data 27](#_Toc95681508)

[4.4.1 Top software by mention count 27](#_Toc95681509)

[4.4.2 Top software usage purposes 28](#_Toc95681510)

[4.4.3 Types of software 28](#_Toc95681511)

[4.4.4 Types of software and share of purpose of use 29](#_Toc95681512)

[5 Sequence Tagging models 30](#_Toc95681513)

[5.1 Introduction 30](#_Toc95681514)

[5.2 Information extraction approaches 30](#_Toc95681515)

[5.3 Sequence classification approaches 31](#_Toc95681516)

[5.3.1 Feature based classification 32](#_Toc95681517)

[5.3.2 Distance based classification 32](#_Toc95681518)

[5.3.3 Model based classification 32](#_Toc95681519)

[6 Feature selection 40](#_Toc95681520)

[6.1 Phrase 40](#_Toc95681521)

[6.2 Sentence 40](#_Toc95681522)

[6.3 Paragraph 40](#_Toc95681523)

[6.4 Dependency tree 40](#_Toc95681524)

[7 Classification and Evaluation 42](#_Toc95681525)

[8 Summary of Results and Conclusion 43](#_Toc95681526)

[8.1 Summary of Results 43](#_Toc95681527)

[8.2 Conclusion 43](#_Toc95681528)

[8.3 Limitations and future work 43](#_Toc95681529)

# 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 is non-obvious and 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 a 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 mention type or 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, disambiguate one software from another, etc.

Various manual and rule based techniques has been attempted in the past to extract information about software. However, machine learning based techniques have not been exploited despite being more powerful. 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 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 extract information about software mentions, particularly, to identify for what purpose a given software is used in a given literature.

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 annotation.
* To perform analysis of SoMeSci data set to drive interesting facts about the data set.
* To select a suitable feature, classifier, and to train a classifier model.
* To evaluate and optimize results.

## 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 used in a research 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 on 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 on selection of feature for the classifier.

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 knowledge 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 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 of science.

## The role of software in research breakthroughs

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

One good example of research breakthrough made possible, 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 text, 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 literatures

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 comes with 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. {Take a look at the SPARQL query used on the Appendix}. 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 {3, 60}

Degree {1, 60}

Timeline, map

Description automatically generated

Diagram, map

Description automatically generated

Degree {7, 60}

Degree {5, 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

The above software categories are related to each other as well. According to a manual analysis of wikidata, 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 three main types of software categories have been identified as follows:

1. Application software
2. System software
3. Software Component

A simplified version of relation between software catefories is depicted on the grapgh below.

#### 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 scope definition for software purposes

|  |  |
| --- | --- |
| Software Usage Purpose | Scope |
| 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 presents descriptions about the SoMeSci data set, the extension process of data set 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 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

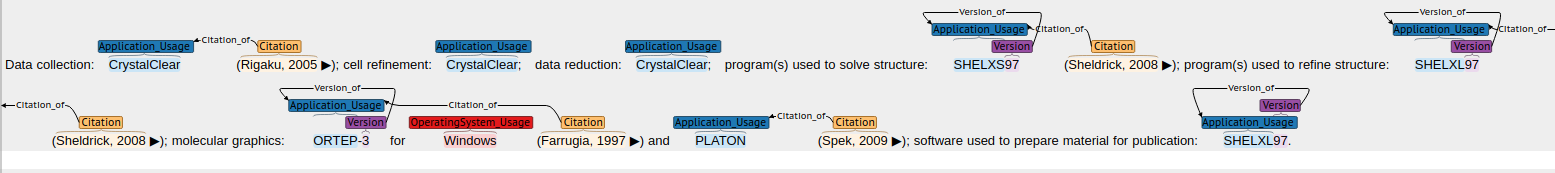
The data set has been annotated using BRAT rapid annotation tool, v.1.3 RRID:SCR\_008769, in a Linux 20.4 environment. The annotation tool has been run in a local machine as a CGI application using a browser.

Graphical user interface, text, application

Description automatically generated

### Extension of SoMeSci

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.

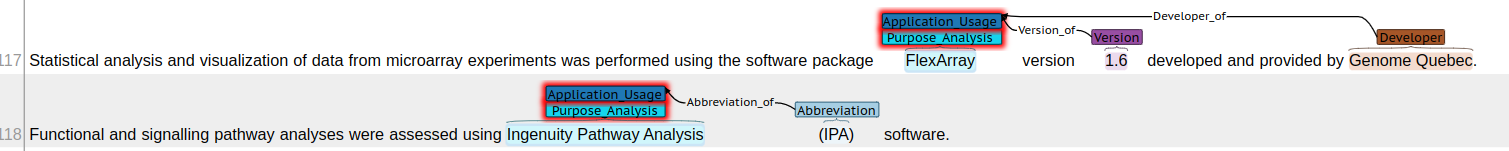


A picture containing text

Description automatically generated

### 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 in essence that visualization could 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 as shown in the figure below, purpose of software usage might not be clear from the context.

In addition in some cases possibility of multiple candidate labels for a given software usage label was also observed. For example in an example below

{PICTURE}

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

## Data Pre-processing

### Handling missing annotations and annotation errors

During the annotation of SoMeSci usage mentions, few instances where the original software usage annotation seemed like just mention of s software has been skipped during the first phase of annotation.

Hence, pre-processing of the data set has been carried out to ensure the integrity of the overall annotation by automatically identifying:

* Software usage annotations that are not annotated yet.
* Erroneous annotation of other types of software mentions such as software mention and creation.

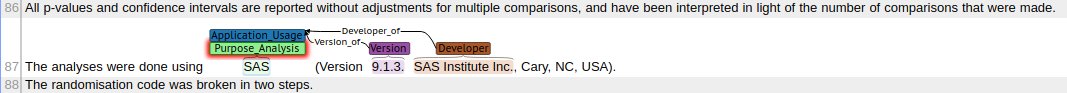
After identifying the list of files and instances of annotations with an error or skipped annotations, all errors have been rectified and skipped annotations has been handled.

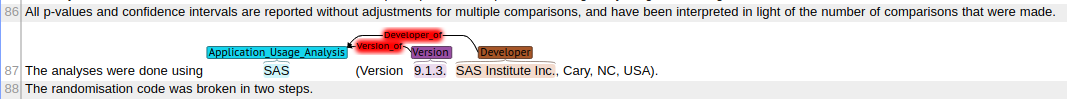
### Merging annotations

After handing all annotation errors and missing labels, annotations of software usage has been merged with annotations of software purpose. Merging of the annotations solves two problems.

First it will 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.

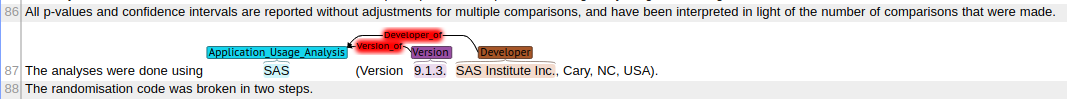
The other reason for merging annotations is to take advantage of legacy code, ariclenizer, which will transform data from BRAT too in a stand-off format into IOB format which is desirable for training purpose.





### Transformation of Data

After merging software usage and purpose labels, transformation of data into IOB format 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

To find further insights about the data, analysis of cleaned SoMeSci data set that contains software\_usage\_purpose has been carried out. Analysis results has been discussed as follows.

### Top software by mention count

Chart

Description automatically generated

### Top software usage purposes

A picture containing circle

Description automatically generated

Chart

Description automatically generated

### Types of software

Table

Description automatically generated

### Types of software and share of purpose of use

A picture containing chart

Description automatically generated

# Sequence Tagging models

## Introduction

## Information extraction approaches

To extract information about software, various approaches have been attempted in the past. From rudimentary approaches such as searching for a term and manual extraction to a more rigorous rule-based approaches and machine learning techniques (kruger2019literature).

Term search approaches are the most basic way to extract information about a software as long as each software has a unique name and if there is no ambiguity. However, in reality a single software can have a different name in different languages as well as spelling variation. For instance the software “Statistical Package for the Social Sciences” can be referred to as SPSS, PASW Statistics, or PASW which makes it difficult to extract all instances of software mentions.

Approaches based on manual content analysis rely on humans reading contents of a research to manually identify and extract information about software. The problems with this approach are:

* It requires a lot of effort to extract information this way.
* It is not scalable for large scale analysis and
* It is difficult to extract information when it is hidden in the textual description.

On the other hand, rule based approaches use rules, such as regular expressions, to match a well defined patterns of information about a software such as Research resource Identifiers (RRIDs). One advantage of rule based approaches is that it can be designed to a various degree of complexity and could perform well if targets follow a regular pattern. However, rule based methods once again fail mainly because of lack of regularity in software citations among researchers who do not follow standardized software citation practices.

All the three approaches: term search, manual based as well as rule based approaches fail to extract information about a software especially when information about software is not explicitly mentioned but concealed in textual description. A natural solution to this problem seems using machine learning approach.

*Information extraction about a software using machine learning technique is apparently the least exploited approach despite being more powerful which can handle even extraction of information about software hidden in textual description. The main reason why this approach not employed, to its potential, could be because of lack of training data which can support training of a classifier for software information extraction.* 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 knowledge [SciBERT]. Fortunately, identification of software mentions from scientific articles has drawn more attention recently over the past years and now various labelled data sets, such as BioNerDs [4], SoSciSoSci [25], SoftCite [3] and SoMeSci [], are available {SoMeSci-A 5 star}.

Sequence tagging models are one of machine learning techniques used for information extraction.

can be modelled as a sequence labelling task, or more specifically Named Entity Recognition problem. Various classical machine learning approaches, supervised as well as unsupervised has been employed in the past (Nev eol et al.,11, ).

Fortunately, SoMeSci, a new set of gold standard labelled data set has been published recently and can be used for automatic extraction of information about software. The data set has been annotated with information about a software such as various types of software and type of mention, relational labels such as version, developer, publisher, license,…etc. Such additional information about software helps to uniquely identify and disambiguate software from one another.

Despite state of the art performance of deep learning approaches for sequence labelling and NER, no pervious work has employed them.

Automatic classification of software usage purpose can be modeled as a sequence classification task because usually a software name is a sequence of tokens. There are various approaches for handling sequence classification problems. This section presents literature review of various classical sequence classification methods as well as the state of the art.

There are various types of sequences according to {xing2010brief}. These are simple symbolic sequence as a sequence of amino acids in DNA, a complex symbolic sequence as an ordered list of items, a simple or multivariate time series.

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7360207> (Rule based)

SEQUENTIAL data is often encountered in a number of important settings, such as texts, videos, speech signals, biological structures and web usage logs, where a sequence is generally an ordered list of singletons. Because of a wide range of applications, sequence classification has been an important problem in statistical machine learning and data mining. The sequence classification task can be defined as assigning class labels to new sequences based on the knowledge gained in the training stage.

## Sequence classification approaches

Sequence classification problems can be overall approached in two ways. One approach is using classical machine learning approaches and the other is using modern deep learning techniques. What is common for both approaches is that both work by transforming a sequence into a vector of features.

### Feature based classification

Feature based classifications work by and performing the classification using a conventional classification algorithms such as decision trees or neural networks.

#### Conventional classification algorithms

* decision trees

#### Neural networks for sequence classification

* bidirectional LSTM (BI-LSTM) networks
* LSTM with a Conditional Random Field (CRF) layer (LSTM-CRF)
* bidirectional LSTM with a CRF layer (BI-LSTM-CRF)

### Distance based classification

Distance based classifications work using a distance function to determine the similarity between a pair of sequences. Hence, performance of classifiers is directly impacted by the choice of distance function. The most commonly used distance function is Euclidian distance.

A picture containing text

Description automatically generated

The problem with the Euclidian distance is that it requires two sequences under comparison to have the same length. However, there are other distance functions that work regardless of sequence length such as Dynamic time warping distance (DTW). Once the distance measure is obtained, classification can be done using algorithms such as K-nearest neighbor classifier (KNN), Support Vector Machine(SVM), etc.

### Model based classification

#### Generative models

* Naive Bayes sequence classifier
* Markov model
  + Hidden Markov Models
  + Visible Markov model (n-grams )

#### Discriminative models

* Maximum entropy Markov Model (MEMM)
* Conditional Random Fields (CRF)

[ <https://www.cs.sfu.ca/~jpei/publications/Sequence%20Classification.pdf> ]

* feature based classification
  + transform a sequence into a vector of features
  + for Conventional classification methods like decision trees and neural networks
* distance based classification
  + define a distance function to measure the similarity between a pair of sequences
  + K nearest neighbor classifier (KNN) and SVM with local alignment kernel
  + Euclidean distance is a widely adopted
* model based classification (e.g HMM)
  + based on generative models,
  + assume sequences in a class are generated by an underlying model M
* Some methods may ride on multiple categories

<https://www.cs.sfu.ca/~jpei/publications/Sequence%20Classification.pdf>

**Sequence classification applications:**

* *genomic research*, classifying protein sequences into existing categories is used to learn the functions of a new protein.
* In *health-informatics*, classifying ECG time series (the time series of heart rates) tells if the data comes from a healthy person or comes from a patient with heart disease.
* In *anomaly detection/intrusion detection*, the sequence of a user’s system access activities on Unix is monitored to detect abnormal behaviors.
* In *information retrieval*, classifying documents into different topic categories has attracted a lot of attentions.
* *classifying query log sequences* to distinguish web-robots from human users.
* classifying *transaction sequence data* in a bank for the purpose of combating money laundering

**Types of sequences:**

* *simple symbolic sequence. E.g.* DNA sequence of four amino-acid A, C, G, T and a DNA segment, such as ACCCCCGT
* *A complex symbolic sequence* - an ordered list of vectors. For example, for a sequence of items bought by a customer over one year… <(milk, bread)(milk, egg)· · · (potatos, cheese, coke)>.
* *A simple time series* : is a sequence of real values ordered in timestamp ascending order. For example, {(t1, 0.1)(t2, 0.3) · · · (tn, 0.3)}
* A *multivariate time series* is a sequence of numerical vectors. For example, {(t1,{0.1, 0.3, 05})(t2,{0.3, 0.9, 0.8})···(tn,{0.3, 0.9, 0.4})}

**Sequence classification challenges:**

1. Most classifiers, such as decision trees and neural networks, can only take input data as a vector of features. However, there *are no explicit features in sequence data*.
2. Feature selection is far from trivial. Even with sophisticated feature selection techniques, the *dimensionality of potential features may still be very high* and the sequential nature of features is difficult to capture.
3. besides accurate classification results, in some applications, we may also want to get an interpretable classifier. Building an interpretable sequence classifier is difficult since there are no explicit features.

**Sequence classification methods – 3 types**

1. *feature based classification (Conventional classification methods)*, which transforms a sequence into a feature vector and then apply conventional classification methods like ***decision trees and neural networks***. Feature selection plays an important role in this kind of methods.

*Feature selection methods:*

* + - * + *k-gram based – for simple symbolic sequence*
        + *pattern-based feature selection – select feature based on some criteria*
        + *Time series shapelets – for time series sequence*

1. *sequence distance based classification* - define a distance function to measure the similarity between a pair of sequences. The distance function which measures the similarity between sequences determines the quality of the classification significantly. Once such a distance function is obtained, we can use some existing classification methods, such as ***K-nearest neighbor classifier (KNN)*** and ***SVM*** with local alignment kernel.

The choice of distance measures is critical to the performance of KNN classifiers.

* ***Euclidean distance*** is a widely adopted for *simple time series* classification. But it usually requires two time series to have the *same length* andsensitive *to distortions in time dimension.*
* ***Dynamic time warping distance (DTW)*** - does not require two time series to be of the same length. DTW is computed has the *quadratic time complexity* and it is costly on a large data set.
* ***alignment based distances*** are popular for symbolic sequences, such as protein sequences and DNA.

**Support Vector Machine (SVM)**

* an effective method for sequence classification.
* basic idea of applying SVM on sequence data is to map a sequence into a feature space and find the maximum-margin hyperplane to separate two classes.
* Sometimes, we do not need to explicitly conduct feature selection. A kernel function corresponds to a high dimension feature space. One of the widely used kernels for sequence classification is *k-spectrum kernel* or *string kernel*, which transforms a sequence into a feature vector.
* Other kernels used for sequence classification include polynomial like kernels , kernels derived from probabilistic model (Fisher’s kernel), and diffusion kernels.
* One disadvantage of kernel based methods is that it is hard to be interpreted and hard for users to gain knowledge besides a classification result.
* The challenges of applying SVM to sequence classification include how to define feature spaces or kernel functions, and how to speed up the computation of kernel matrixes.

1. *Model based classification*

* Is category of sequence classification technique based on generative models, such as using Hidden Markov Model (HMM) and other statistical models to classify sequences, which assume sequences in a class are generated by an underlying model M.
* Given a class of sequences, M models the probability distribution of the sequences in the class.
* Usually, a model is defined based on some assumptions, and the probability distributions are described by a set of parameters.
* In the training step, the parameters of M are learned. In the classification step, a new sequence is assigned to the class with the highest likelihood.

***Naive Bayes sequence classifier***

* The simplest generative model is the ***Naive Bayes sequence classifier.*** The naïve assumption is: *“given a class, the features in the sequences are independent of each other”.*
* The conditional probabilities of the features in a class are learned in the training step.
* However, the independence (*the naïve*)assumption of Bayes is often violated in practice. Therefore *Markov Model* and *Hidden Markov Model* can model the dependence among elements in sequences.

***Markov model***

* a *k-order Markov model* to classify protein and text sequence data.
* In the training step, the model is trained in a discriminative setting instead of the conventional generative setting to increase the classification power of the generative model based methods.

***Hidden Markov Model***

* Different from Markov Model, Hidden Markov Model assumes that the system being modeled is a Markov process with unobserved states.
* a ***profile HMM*** has been used to classify biological sequences. A profile HMM usually has three types of states, inserting, matching and deleting.

**Sequence Classification with A Sequence of Labels**

* labeling sequences is to label each element in a sequence. For example, given a sentence, where each word is treated as an element, sequence labeling is to assign each word to a category, such as name identity, noun phrase, verb phrase etc.
* *straightforward solution*: to label each element independently.
* *advanced solution:*  to consider the labels of the elements in a sequence related to each other.
* Sequence labeling problem has been solved by using ***conditional random fields***.

<https://arxiv.org/pdf/1508.01991.pdf>

**Sequence Tagging**

* Can be done via a variety of LSTM methods, which are an artificial [recurrent neural network](https://en.wikipedia.org/wiki/Recurrent_neural_network) (RNN) architecture used in the field of [deep learning](https://en.wikipedia.org/wiki/Deep_learning).
* Other sequence tagging models are:
  + Hidden Markov Models (HMM),
  + Maximum entropy Markov models (MEMMs) &
  + Conditional Random Fields (CRF)
* In addition Convolutional network based models like ***Convolutional-CRF*** that has a ***convolutional network*** and a ***CRF layer*** on the output.
* Variants of LSTM are:
  + bidirectional LSTM (BI-LSTM) networks
  + LSTM with a Conditional Random Field (CRF) layer (LSTM-CRF)
  + bidirectional LSTM with a CRF layer (BI-LSTM-CRF)
    1. can efficiently use both past and future input features because of a bidirectional *LSTM component*. Aka context encoding … encodes the past and future.
    2. can also use sentence level tag information because of a *CRF layer.*

**What are sequence tagging tasks ?**

Classic NLP tasks like:

1. **Part of speech tagging (POS**) - aims at assigning a correct part-of-speech tag to each lexical item.
2. **Text Chunking** - The goal of the text chunking task is to divide text into syntactically related non-overlapping groups of words, i.e., phrase, such as noun phrase, verb phrase, etc.
3. **Named entity recognition (NER)** - identify named entities from text belonging to pre-defined categories. Generally consists of three major categories (i.e., **entity**, time, and numeric) and seven sub-categories (i.e., **person name, organization, location**, time, date, currency, and percentage)

These are often a base line up on which other NLP applications like machine translation, question answering, information retrieval, etc.…are built.

<https://arxiv.org/pdf/2011.06727.pdf> - **Summary of DL techniques**

**Sequence labeling approaches**

**Classical machine learning approach** – statistical techniques, based on the carefully designed features to represent each training data, the machine learning algorithms are utilized to train the model from example inputs and their expected outputs, learning to make predictions for unseen samples.

* Examples:
  + HMM
  + SVMs
  + Maximum Entropy Models
  + CRFs – heavily rely on hand crafted features
* Pros:
  + superior performance
* Cons:
  + considerable amount of domain knowledge required and
  + efforts on feature engineering make them extremely difficult to extend to new areas

**Deep learning approach**

* State of the art performance

<http://www.cs.cornell.edu/~nhnguyen/icml07structured.pdf> **Comparisons of Sequence Labeling Algorithms**

**Algorithms :**

* + Conditional Random Fields (CRF),
  + Averaged Perceptron (AP),
  + Structured SVMs (SV Mstruct),
  + Max Margin Markov Networks (M3N), and
  + an integration of search and learning algorithm (SEARN)
  + HMM

C. Aggrawal

**NER**

* is the most fundamental problem in Information Extraction because it provides the basic building block on the top of which many useful Information Extraction (IR) methods are built. For ex, It is not possible to do Relation extraction before NER.
* Information Extraction pipeline:
  + 1. Tokenization
    2. POS Tagging
    3. Parsing
    4. Dependency Extraction
    5. NER
    6. Relation Ex

**NER approaches**:

* Using gazetteers – a dictionary of all known entities
  + Cons:
* The set of known entities are not constant and evolves with time. New entities emerge as well as some entities cease to exist . e.g. company names.
* Ambiguity – depending on a context an entity might refer to something else. A context in which a particular term is used is important to consider.

**Two approaches for Information Extraction**:

* Rule Based approach
  + Each token converted to a set of features.

Features are properties of tokens or their context that could assist for information extraction.

Therefore, feature engineering is an important aspect of rule-based approaches.

common features:

***surface value*** – string representation of the token

***Orthography of token (aka shape feature)*** – e.g Caps, Punctuation, spelling…

***Pos Tags***

***Syntactic base phrases***

***Dictionaries*** – for name prefixes, company endings like Inc.

**Token Level Classification (a.k.a. Sequence Labelling/Tagging)**

* Unlike rule-based methods, using feature for a token is not helpful for sequence tagging tasks. This is because, classifiers that treat each token independently are not useful because in sequence tagging tokens before and after a token are informative.
* A natural approach for handling sequence tagging is using ***recurrent neural networks***! Other approaches for Sequence Labeling include:
  + Hidden Markov Models (HMM)
  + Maximum Entropy Markov Models (MEMM) &
  + Conditional Random Fields (CRFs)

**HMM**

* Generative model, because HMMs generate sequences using transition between states.
* HMMs transition through a sequence of hidden states and each state produces a *token*.
* States in the HMM are *dependent on each other*.
* Types of HMMs:
  + Visible HMMs ( n-grams )
* States of the model are directly associated with the generated symbols.
* Are n-gram models are visible models in which the state is defined by the last n-1 words.
* The number of states = the number of tokens
* n-grams are special case of HMMs.
  + Hidden HMMs

**Maximum Entropy Models**

* Discriminative model
* Directly model probability of the tag based on the states.

**Conditional Random Fields (CRFs)**

* Closer to maximum entropy models
* In maximum entropy models, restrictions are placed on the probabilistic modelling on the labels occurring before it but not after it. In CRFs inference of the label depends on both the labels occurring before and after.

# Feature selection

## Phrase

## Sentence

## Paragraph

## Dependency tree

# Classification and Evaluation

# Summary of Results and Conclusion

## Summary of Results

## Conclusion

## Limitations and future work