
Final Report

Submitted as part of the requirements for:

CE903 Group Project

Topic: Fact-checking (based on Wikipedia)

Group Number: 6

Registration Number of Group Members:

1802772

1806447

1804523

1806152

1807600

1807846

Date of Submission: 21 March, 2019

Table of Contents

1	Introduction	1
1.1	Description of project domain and project objectives	1
1.2	Project planning and employed methodology	3
1.2.1	Software Engineering Methodologies	3
1.2.2	Agile model	6
1.2.3	Waterfall model	8
1.2.4	Gantt chart	9
1.2.5	Change Management	10
1.3	Description of software platform and used tools	11
2	System Design	17
2.1	Requirements Specification	18
2.2	System Architecture	19
3	Implementation	21
3.1	Implementation methodology of first version	21
3.2	Implementation methodology of final version	23
3.3	Evaluation the performance of final version	25
4	Testing	28
4.1	Questions related to testing	28
4.2	Strategy and level of testing	29
4.2.1	Unit testing	32
4.2.2	Regression Testing	32
4.2.3	Integration Testing	33
4.2.4	Alpha Testing	34
4.2.5	Acceptance Testing	34
4.3	Testing detail of the project.	35
5	Conclusions	37
5.1	System related limitations and possible way of improvement	37
5.2	Effectiveness of the implementation	38
5.3	Project management	38
	References	40
	Appendix	41

List of Figures

Figure 1: Initial ideas to implement the fact-checking process	2
Figure 2: Work flow of agile model	7
Figure 3: Work flow of Waterfall model	8
Figure 4: Gantt chart of our project	10
Figure 5: Common tools for the development of the software	11
Figure 6: Scikit-learn for Machine Learning Algorithm	13
Figure 7: Data Processing Model of deployed Fact Checking System	19
Figure 8: Data Flow Diagram of deployed Fact Checking System based on Wikipedia	20
Figure 9: Implementation methodology of first version	22
Figure 10: The implementation methodology of final version of the software	23
Figure 11: Activity sequence of the final version of the software	24
Figure 12: Type of our software testing	31
Figure 13: Different level of testing	31
Figure 14: Unit Testing (black / White box) at pre-processing stage	32
Figure 15: Regression testing (along with white box)	33
Figure 16: Integration testing (along with white box/black) after stage-1 processing	33
Figure 17: Alpha testing (along with white box/black) after stage-1 processing	34
Figure 18: Overall process flow of our project	39

List of Tables

Table 1: Output of Maximum similarities score	27
---	----

1 Introduction

At present, fake news that consists of deliberate disinformation is considered as one of the greatest challenges to handle. The fake news usually spread via news media, traditional print media, and online social media. One possible way to combat fake news is fact-checking. Although fact-checking poses various kind of challenges, the growing movement of fact-checking plays an important role in improving the political and social discourse [1], [2].

One of the biggest challenge related to the fact-checking process is the rapid spread of misinformation. Through the different medium of technology, social media, and journalism it is very easy to disseminate falsehood that frequently outplaces the truth. Some challenges that associated with the fact-checking process are lack of proper resources for necessary investigation, advance and robust method of fact-checking, and time-consuming process [3], [4], [5]. Although several fact-checking platforms have been developed by many organizations, fact-checking is an intellectually demanding process which requires more research.

Through our project, we have tried to develop a simple system to provide a comprehensive solution to perform the fact-checking against online Wikipedia pages for a given user query. Our system can be expanded further to provide better solutions in terms of authenticity and identification of facts by using Wikipedia and other possible authentic sources. As the fact-checking is a vast area and required more research, our deployed software is the beginning of a noble approach, which can be improved with further analysis and appropriate research.

1.1 Description of project domain and project objectives

The main objective of our deployed system is to identify facts based on Wikipedia for a given user query. Wikipedia is a free, enormous, cross-language NLP repository, and a good source for basic level fact-checking. Initially, the domain for the project was restricted to the person. Later it has been changed to a domain that allows multiple entities called people, place, and organization. Our project is consists of several stages and changed with the necessary feedback from the client.

Initially, we have considered three ideas to implement the fact-checking process. Below-mentioned diagrams summarize those ideas.

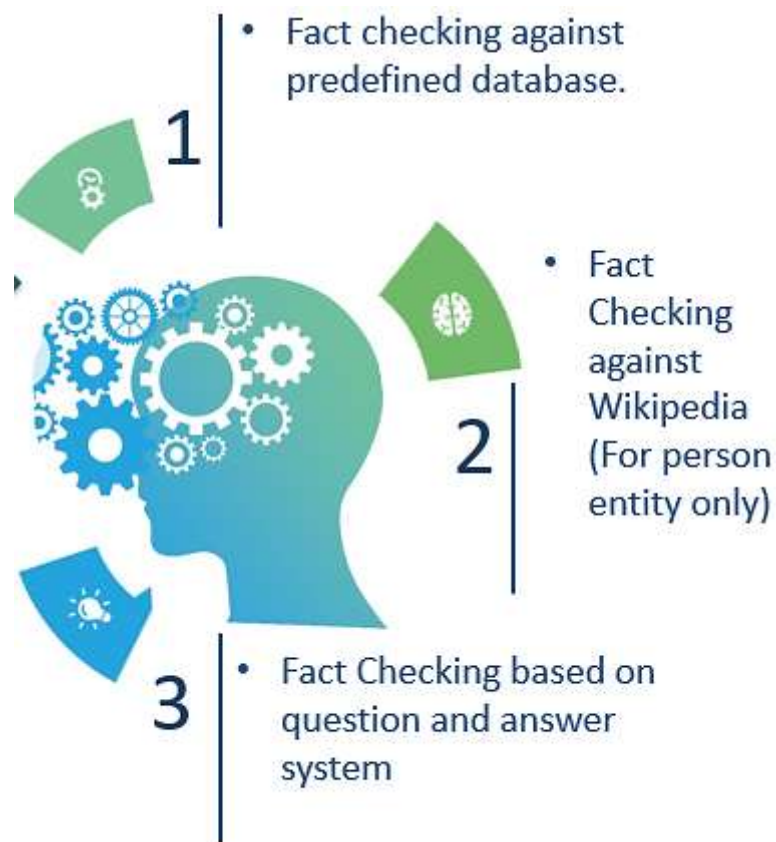


Figure 1: Initial ideas to implement the fact-checking process.

After the initial feedback from the client (Professor), fact-checking by using Wikipedia has been selected. Similarly, the person entity has been elected as the domain. However, due to the simple approach and person-specific domain, the client has asked to modify the domain. The client also asked to change the domain and make it more effective and capable to handle other entities. For instance, people, place, and organization. Necessary modifications have been done as per client suggestion. Therefore, we finalized the domain for multiple entities based on client feedback. Our final development is based on a domain that supports multiple entities which eventually broaden the scope of the implementation.

1.2 Project planning and employed methodology

In order to determine the suitable methodology for our software development project, we have gone through different software methodologies that have been used in the current age of the information era. All of the current software engineering methodologies have some advantages. The next subsections will describe why some methodologies are more relevant to our projects. Same time it also describes why a particular model did not pertain to our project specifications. At the beginning of the project, we opted for the agile model. However, as we proceeded further into the project, we had to shorten the plan to a waterfall model. Through this subsections, we will discuss further the employed project management methodology and what led us to proceed to a change in the Software Engineering model. The latter part of this section has our Gantt chart and a simple work breakdown structure of our model, and we have also given a short insight on how changes are managed through the implementation of our system.

1.2.1 Software Engineering Methodologies

When someone, has a plan to build some kind of Software Engineering project, the first thing that comes into their mind is what kind of methodology should be used to tackle this project. Similarly, when we look into Software Engineering Models, to tackle our problem statement of trying to identify, Fake News, through a fact-checking software, we looked into models, like the Waterfall Model, the V-Shaped Model, the evolutionary prototyping model, the Spiral Model, the iterative, and the Agile Model.

As we indulged deeper, into studying the literature, we had a deeper analysis of what our system would uphold, so we began to rule out particular models.

- *The V-Model*

When we looked in upon the V-model, we first, got to identify, that this model, also called, the verification and validation model, is quite similar to the Waterfall model. Like the Waterfall model, this model, also worked, through means of phases, wherein each phase was required to be completed before one could move on to the next phase. When we looked into this model, we saw that each phase was divided into 2 parts, a

verification phase, and something to test that verified Stage, or rather the validation phase.

So when the user requirements are being gathered, the User acceptance testing stage is being considered, the system requirements are being gathered, the system testing phase is taking place, when the high-level design is taking place, the system integration testing phase is going on in parallel, when the low level or detailed design process is taking place, the Unit Testing phase is simultaneously considered and so on.

However, when, we started out this plan, we initially made a number of test cases, our field of area of fact-checking originally considered to be a broad area of study, we felt, a simple parallel phase consideration would be inappropriate, as our testing standards, could change with time, and our system was overly flexible.

Also, we were particularly interested, in identifying at an early stage, what our coding, or code, would reveal to us, and how efficient would the code that we apply, successfully be able to identify, fact from fake news. Hence, seeing that this model relies on coding only at the later stage and that our project was meant to work on a broader scale, we decided to drop considering this modelled approach for carrying out our project.

- ***Incremental Model:***

When we tried to analyse the incremental model, we saw that this model also quite similar to the Waterfall model, wherein, the entire process was divided into phases, but here, there was a catch. Making use of this model would mean that the software is finished very early during the process and that each stage, new development would occur to the previously completed model. So basically, we would divide the requirements into batches, and work on each batch and integrate at the final stage.

Although we intended to make our system behave in such a similar fashion, this would be suitable for time-independent projects. However, we did have an initial idea, about we expect to carry out during the output, but implementing the idea, at the initial stage was still vague, hence we dropped the idea of creating an Incremental model.

Iterative Model: The Iterative Model works just like the incremental model. However, in this model, the requirements itself isn't considered as a whole in the first stage. So unlike the incremental model, where all the requirements are already predefined, in this model, the requirements are carried out partly, and then after a peer review, we find out if the further portion of the requirements is needed. In our system, although we did have our requirements initially different, wherein, we had intended to implement our Fact checking model with only information from a Wikipedia source pertaining to a person, but later on, we increased the odds, to a broader category of Wikipedia sites. Our initial focus, did not have a clear display of the implementation plus, making use of designing this model is more costly than making use of any other simpler model like the waterfall model, hence, we dropped the idea of making use of an iterative model.

- ***Spiral Model***

As of this model, this model is more similar to the incremental model, but here, a greater emphasis is provided on risk assessment and risk analysis. In this model, there are 4 phases, Planning, Risk Analysis, Engineering, and Evaluation. In this model, each step or phase repeats itself to provide more functionality, like the arms of a Spiral. Each future spiral works on the primary or baseline spiral in an incremental fashion.

In the planning stage, the requirements and specifications are gathered, like the Business requirements and Software specification. In the risk analysis stage, a risk analysis session takes place, and a risk plan is created, to identify risks, and a risk report is created to identify the risks and the solutions pertaining to those risks.

In the Engineering phase, Software is developed and tested, so the coding and testing part is carried out in this stage. The evaluation stage is important for this model, because, it is at this stage, that the software designer has to decide, whether to continue with the spiral. In this stage, the business owners, test the software against the requirements and see if the requirements are satisfied. If not, continue on with the spiral.

Although this model, was very efficient in identifying risks, as well, help in dealing with mission critical. And large scope projects, this model, was like the incremental model, a very costly model to apply, and there was a higher emphasis on risk analysis, which was not the major agenda of our model.

In the end, we shortlisted two models, that we believed, would work, best on our project, the agile model and the waterfall model.

1.2.2 Agile model

In today's era of the information age, a lot of emphases has been catered on the rise of technological advancements, it is considered normal, if you have your client approach, you saying he needs to have a variation, in his original set of requirements. Furthermore, such is the complexity of today's projects, that a normal project manager, would insist on considering making use of a hierarchical model, rather than use simply a single model like a V-Shaped Model. Hence it is of utmost importance that at the very earliest Project planning stage, we should have a model that has a high confidence level to base with the constant ever-demanding nature of requirements, code changes, code optimization, design and model evolutions, and model evaluations, test upgrades and changes in deployment specifications.

Most of the traditional software engineering projects that are undertaken in organizations make use of a sequential stepwise manner of procedure, commonly called the waterfall model but what differentiates the waterfall model from some of the more advanced methodologies is its lack in adapting to the ever-demanding changes. Hence an approach, we initially focused on in fulfilling for our project was the Agile Methodology.

One would call the Agile Methodology, an increment to the traditional waterfall model. In this model, the approach to changes in any stage of the Software development Life Cycle is maintained by carrying out by constantly updating the prior stages in tandem with the current stage in the form of an iterative and incremental approach. Agile as such is more of a communication dependent model, and the agile process does not necessarily have a fixed predefined process, rather it is

a Software Engineering process that brings together the software development principle members and stakeholders to act responsively in the time of change rather than following a fixed and dedicated plan. Hence, the agile method works more on a real-time basis, wherein, this model has a strong capability to meet the ever demanding requirement changes of clients and stakeholders.

An example of the agile method process can be seen in the below figure:

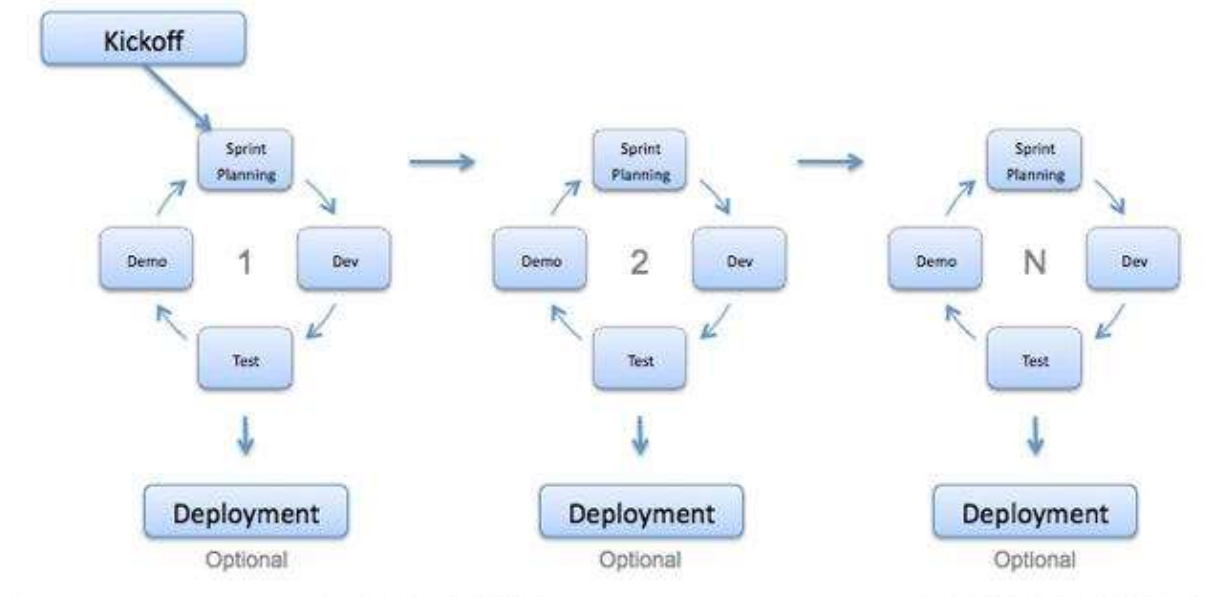


Figure 2: Work flow of agile model

Each cycle of the agile model is called a sprint, we have a Sprint master who manages the entirety of the Sprint structure, we have the project owner who specifies the task to the project engineers, who perform their tasks and peer review their completed tasks to get the necessary feedback.

In our case of identifying facts from a source like Wikipedia or any other website. The source like Wikipedia is not predefined, and it is constantly staying abreast with the facts, and furthermore, a future requirement might involve having the system work on other sources. Our project may also adapt to run on more advanced techniques of data processing or machine learning, or furthermore to fix any defects that may arise of such future versions, hence we initially employed the use of an agile approach of methodology.

However as and, when we looked into the literature concerning our approach towards the project, a lot of questions rose in our mind. How efficient will our fact-checking algorithm be, and how can this be employed in more than one source. In taking these questions into consideration, we came into realization that the fact-checking domain, is much broader in scope, and although this project should be modelled using the Agile approach, should there be no time constraints, modelling our system to meet with a narrower approach to single website source, and a more specific, fact detection algorithm, allowed us to look for a simplistic approach like the Waterfall model.

1.2.3 Waterfall model

The waterfall model is the most commonly used model used to design smaller projects with limited specifications. In a waterfall model, the requirements are already known at the earliest stage of the model, and hence this kind of model is generally used in most contract related projects. In the waterfall model, each stage has its individualistic importance. In this model, each stage is attended to in a sequential manner, right from requirements gathering, planning, designing, implementing, testing and deploying stages. In other words, the waterfall model is called the linear sequential lifecycle model. An example of the waterfall model is shown below.

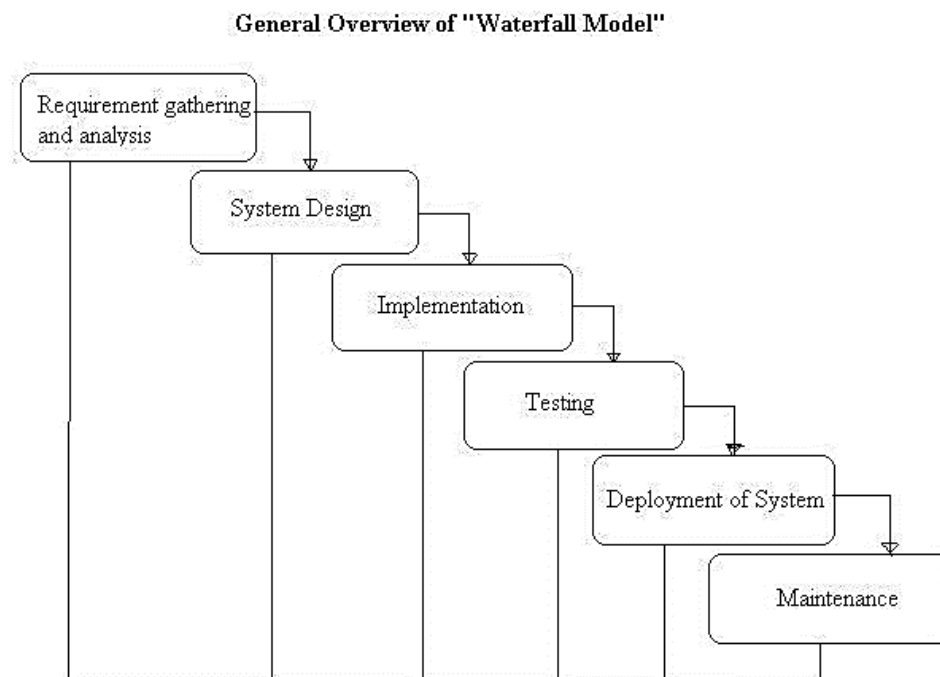


Figure 3: Work flow of Waterfall model

As can be seen from the diagram, the Waterfall model is divided into several stages: In the requirements gathering stage, much of the elicitation, and requirements are gathered, it is one of the most important stages of this model, because, a proper requirement gathering would mean a proper project outcome to the primary stakeholders. In the design analysis stage, more emphasis will be put on the type of model to be created, like as in the Use cases, and in our cases, the dataflow diagrams relevant in the fact-checking process. Our system requirement analysis is carried out in this stage.

In the implementation stage, our model and code are generated, and we perform unit testing at this stage. In the testing phase, more of the detailed testing techniques are carried out, test cases are built and a test plan is created.

In the deployment stage, the project is deployed to the stakeholders and future maintenance scopes are carried about.

We designed our system with the waterfall model, because of its simplistic approach in use, and having our requirements exactly specified at the earliest stage made it plausible to choose this model.

Aside from these features, as we decided to narrowly bring down the specifications of fact-checking to a single source like Wikipedia, this model was most apt for our requirements.

1.2.4 Gantt chart

Below is a Gantt chart specifying which of the steps of our project was carried out, and when each of the steps was carried out.

Task/Time	Jan 23rd 2019- Feb 1st 2019	Feb 2nd 2019- Feb 10th 2019	Feb 11th 2019- Feb 20th 2019	Feb 21st 2019- Mar 2nd 2019	Mar 3rd 2019- Mar 12th 2019	Mar 13th 2019- Mar 21st 2019
Perform Literature survey and come up with a solution.						
Build an SRS Report, and Design Document						
Testing of Development Environment and components						
Identifying Source datasets, and data collection						
Web crawling coding and testing						
Nltk preprocessing and Data parsing coding and testing.						
Data structure conversion and data structure testing completion						
Information retrieval coding and output testing.						
Output Fact Checking from query. And deployment						
Building on Final report of our findings						

Figure 4: Gantt chart of our project

1.2.5 Change Management

One of the most important aspects of the life of any good software is its adaptability to change. This adaptability to change must properly ensure that the given software is stable in the market. To make sure that this proves of change happens smoothly, one must focus more on how one's software can learn to respond in terms of a disaster or a crisis, in terms of adaptability to newer technology, and scope of the project.

Our case primarily focuses on extracting information from a single source like Wikipedia. Given the simplicity and ease of use of our model, is providing an approach towards web scraping for the relevant information from a website. It will be quite easy to adapt our system to handle say another organization's websites, simply through small modification to the code. Another focus on change should be in the concept of defect management and crisis management. Having our system work with just one source of a website will nullify most defects. Furthermore, it is also possible to run our code by web scraping through multiple sources, hence there are possibilities of us coming across some of these defects which could occur through fetching information, data pre-processing, or setting up a poor threshold in the identification of facts. These defects can be handled through efficient coding and testing strategies.

Some of the more uncontrolled defects can arise in terms of power failure, or an error with loading server. As far as a power failure is concerned we will be making use of

storing a backup database of our processed information per each query. Further enhancements in our code in the future can include management of concurrent query processing. As far as server loading errors are concerned, the websites used in our case, Wikipedia are publicly well-renowned websites working in an integrated cloud environment; hence this eliminates the need for concern for server.

1.3 Description of software platform and used tools

Our implemented system is mainly based on user input (a query provided by the user) and online Wikipedia. Therefore, our implemented system is fully based on online. In order to implement the fact-checking process we have used **python 3+** as our development language. Our developed code could be run by using any operating system (*Windows, Linux, Mac-OS, and Ubuntu*). It is flexible and based on some common python libraries. Moreover, our implemented system has two versions. To develop, implement and test the two versions we have used below mentioned tools (Python Libraries).

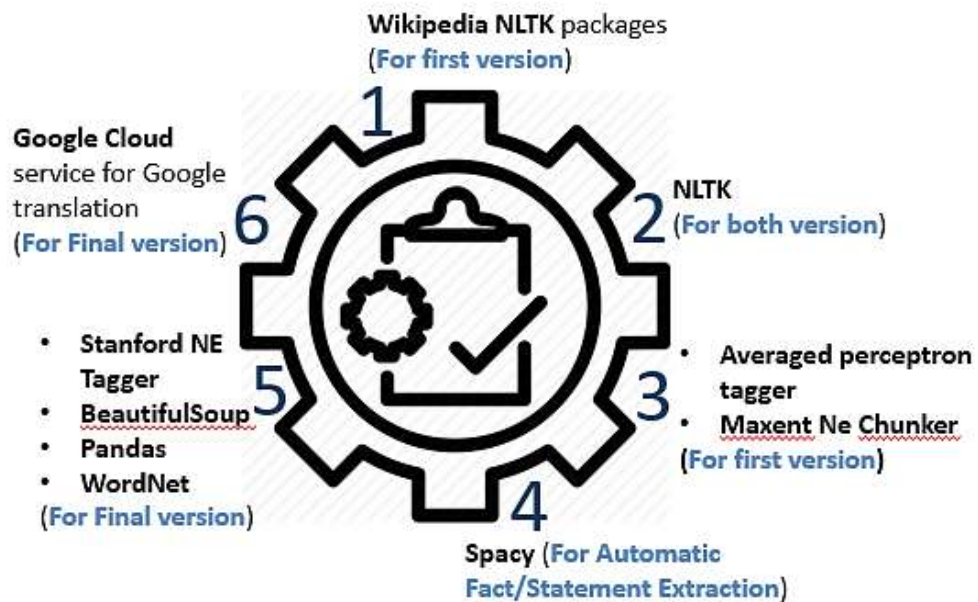


Figure 5: Common tools for the development of the software.

The description of necessary tools that have been used to develop the software is as follows.

NLTK:

NLTK is a natural language processing toolkit, one of the most commonly used Python libraries in the NLP world. NLTK is an open source project that includes: Python modules, datasets, and tutorials for NLP research and development. NLTK includes graphical presentations and sample data. It provides tutorials that explain the basic concepts behind the language processing tasks supported by the toolkit. Some common function that has been used for our deployment is as follows.

```
Nltk.sent_tokenize(text) # Segment the text according to the sentence
Nltk.word_tokenize(sent) # Segmentation of sentences
NLTK for part-of-speech tagging.
Nltk.pos_tag(tokens) #tokens is the result of a sentence segmentation, also a
sentence-level annotation
NLTK for Named Entity Recognition (NER)
Nltk.ne_chunk(tags)#tags is the result of the phrase part-of-speech tagging,
also the sentence level
NLTK for Syntax analysis
```

Nltk does not have a good parser. Therefore, it is recommended to use Stanford parser. However, nltk has a nice tree class, which is implemented with a list. We can build a python syntax tree using the output of Stanford parser.

```
>>> tparse = nltk.tree.Tree.parse
>>> tree = tparse("(NP (DT the) (JJ fat) (NN man))")
>>> for subtree in tree:
    print subtree, '---', subtree.node

(DT the) --- DT
(JJ fat) --- JJ
(NN man) --- NN
```

NLTK for Stem extraction (stemming)

Explain that Stemming is the stem or root form of a word (not necessarily capable of expressing complete semantics). Three of the most commonly used stemmer interfaces are available in NLTK, namely Porter stemmer, Lancaster Stemmer and Snowball Stemmer.

NLTK for Lemmaization

Lemmatization restores a vocabulary of any kind to a general form (which expresses complete semantics). Relatively speaking, stemming is a simple and lightweight method

of morphological merging. The final result is stem, which does not necessarily have practical significance. The morphological reduction process is relatively complicated, and the result is the prototype of the word, which can carry a certain meaning. Compared with the stem extraction, it has more research and application value.

NLTK for Maximum matching algorithm (MaxMatch)

The algorithm gradually reduces the length of the string from the right side to find the maximum length of the string that may match. Considering that the vocabulary we get may contain some kind of morphological changes, we use Lemmatisation and then perform matching lookups in the thesaurus.

Scikit-learn

Scikit-learn is also based on the python library, which includes many advantages. Some advantages are as follows.

- Built on existing [NumPy](#) (basic n-dimensional array package), [SciPy](#) (scientific computing base package), [matplotlib](#) (full 2D/3D drawing), [IPython](#) (Enhanced interactive interpreter), [SymPy](#) (Symbolic mathematics), [Pandas](#) (data structure and analysis), made an easy-to-use package.
- Simple and efficient tools for data mining and data analysis.
- Open to everyone and easy to reuse in many scenarios.
- This is the algorithm map of Scikit-learn:

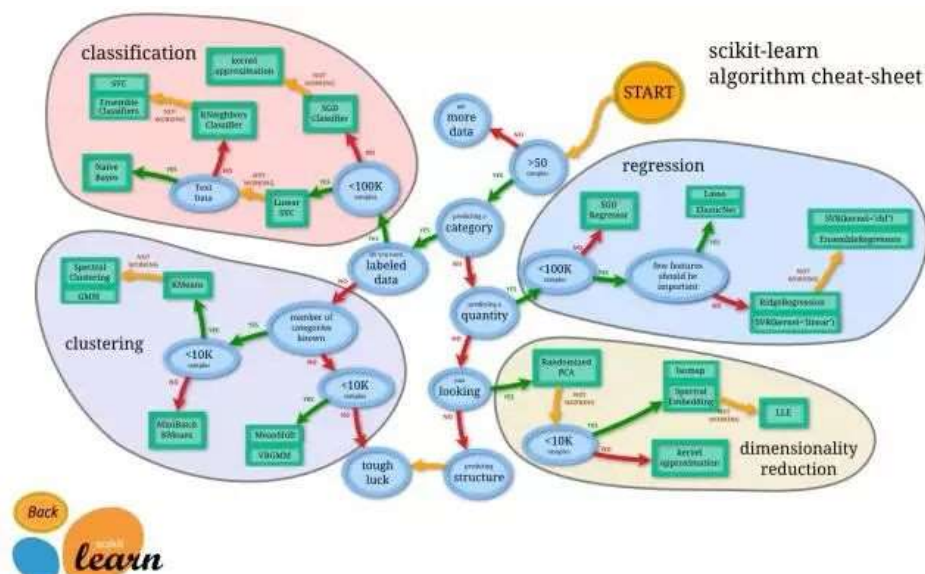


Figure 6: Scikit-learn for Machine Learning Algorithm.

According to the above figure, the main functions provided by scikit-learn are mainly concerned with data modeling, rather than loading, manipulating, and summarizing data. These tasks may be sufficient for [NumPy](#) and [Pandas](#). To this end, scikit-learn mainly provides the following functions:

- Test data set, sklearn. Datasets module provides many open source data sets for breast cancer, kddcup 99, iris, California house prices, etc.
- Dimensionality Reduction: Reduce the number of attributes for feature screening and statistical visualization.
- Feature extraction: Define properties in a file or image.
- Feature selection: To identify the attributes that have real relationships in order to establish a supervised learning model.
- Classification by algorithm function, divided into supervised learning: classification and regression, and unsupervised learning: clustering.
- Clustering: Use an algorithm such as K-Means to classify unlabeled data.
- Cross Validation: To evaluate the performance of the supervised learning model.
- Parameter Tuning: Adjust the parameters of the supervised learning model to get the maximum effect.
- Manifold Learning: To count and delineate multi-dimensional data.

Beautiful Soup

Beautiful Soup provides some simple, python-like functions for handling navigation, searching, modifying parse trees, and more. It's a toolbox that provides users with the data they need to crawl by parsing the document. Because it's simple, you don't need much code to write a complete application. Beautiful Soup automatically converts the input document to Unicode encoding and the output document to UTF-8 encoding. You don't need to consider the encoding method, unless the document does not specify an encoding method, then Beautiful Soup can not automatically identify the encoding. Then, you just need to explain the original encoding. Beautiful Soup has become a great python interpreter like lxml and html6lib, giving users the flexibility to provide different parsing strategies or powerful speeds.

Four object types of Beautiful Soup

Beautiful Soup transforms complex HTML documents into a complex tree structure, each node is a Python object, and all objects can be grouped into four types:

- Tag
- NavigableString
- BeautifulSoup
- Comment

The BeautifulSoup object represents the entire contents of a document. Most of the time, it can be treated as a Tag object, which is a special Tag. We can get its type, name, and properties to feel it.

```
Print type(soup.name)
#<type 'unicode'>

Print soup.name
# [document]

Print soup.attrs
```

WordNet

WordNet, as a linguistic ontology library, is also a semantic dictionary, which is widely used in natural language processing research. The most striking difference between WordNet and other standard dictionaries is that it divides the vocabulary into five broad categories: nouns, verbs, adjectives, adverbs, and function words. In fact, WordNet only contains nouns, verbs, adjectives, and adverbs. Words are usually part of the linguistic syntactic component, and WordNet ignores the smaller set of imaginary words in English. The most distinctive feature of WordNet is the organization of vocabulary information based on word meaning rather than word form. It can be said that WordNet is a semantic dictionary. But unlike the alphabetical semantic dictionary and the semantic dictionary arranged by topic, it is organized according to the matrix model of vocabulary.

spaCy

spaCy is written by **cython** (Python's C language extension, designed to make Python programs perform as well as C programs), so it runs very efficiently. **spaCy** provides a streamlined set of APIs that are easy to use and based on well-trained machine learning and deep learning models.

The use of **spaCy**, as well as its various properties, is achieved by creating pipes. When the model is loaded, **spaCy** will create the pipeline. In the **spaCy** package, a variety of

modules are provided that contain various information about language processing such as vocabulary, training vectors, syntax, and entities.

The `spaCy` document can be split into single sentences during the tokenized process, and these single sentences can be further split into words.

```
# document's first word
document[0]

>> Nice

# document's last word
document[len(document)-5]

>> boston

# List the sentences in document
list(document.sents)

>> [ Nice place Better than some reviews give it credit for.,
Overall, the rooms were a bit small but nice.,
Everything was clean, the view was wonderful and it is very well
located (the Prudential Center makes shopping and eating easy and the
T is nearby for jaunts out and about the city).]
```

Part of speech tagging (POS Tag)

Part-of-speech tagging refers to the part of speech of a word in a grammatically correct sentence. These annotations can be used for information filtering, statistical models, or

```
# Get all annotations
All_tags = {w.pos: w.pos_ for w in document}

>> {97: u'SYM', 98: u'VERB', 99: u'X', 101: u'SPACE', 82: u'ADJ', 83:
u'ADP', 84: u'ADV ', 87: u'CCONJ', 88: u'DET', 89: u'INTJ', 90:
u'NOUN', 91: u'NUM', 92: u'PART', 93: u'PRON' , 94: u'PROPN', 95:
u'PUNCT'}
```

The part of the first sentence in # document

```
For word in list(document.sents)[0]:
    Print word, word.tag_

>> ( Nice, u'JJ') (place, u'NN') (Better, u'NNP') (than, u'IN') (some,
u'DT') (reviews, u'NNS') (give, u'VBP') (it, u'PRP') (creit, u'NN')
(for, u'IN') (., u'.')
```

Entity recognition

spaCy has a fast entity recognition model that finds entity phrases from document. It recognizes various types of entities such as person names, locations, institutions, dates, numbers, and more. You can read these entities with the ".ents" attribute.

```
# Get all types of named entities in document:
Labels = set([w.label_ for w in document.ents])
For label in labels:
    Entities = [cleanup(e.string, lower=False) for e in
document.ents if label==e.label_]
    Entities = list(set(entities))
    Print label,entities
```

Noun Phrase (NP)

Dependency trees can also be used to generate noun phrases:

```
# Generate noun phrases
Doc = nlp(u'I love data science on analytics vidhya')
For np in doc.noun_chunks:
    Print np.text, np.root.dep_, np.root.head.text
>> I nsubj love
    Data science dobj love
    Analytics analytics pobj on
```

2 System Design

The system design for the fact-checking process based on Wikipedia is consists of different levels. In order to implement an effective system design, we need to prepare well-defined requirements with a clear idea about the implementation of the requirements specification. Next four subsections describe the necessary requirements to design an effective system.

2.1 Requirements Specification

The fact-checking process mainly identifies whether a specific claim is true or false or in between them based on some methodology defined by specific organization/ fact-checking community [2], [4], and [5]. Our developed fact-checking system mainly based on Wikipedia and it verdicts the user claim as true or false based on available information in Wikipedia for that particular user claim [6], [7], [8], and [9]. Our system is capable to handle any facts related to a person, organizations, and places as far as the fact is present in Wikipedia pages. Initially, our system was based on a person entity only. Later it has been changed and the final system is able to identify the facts for a person, organizations, and places.

The functional requirements of the implemented fact-checking system are based on given input by the user and the online Wikipedia pages. Therefore, the deployed system process the user query based on Wikipedia pages to identify the facts and to meet the user information needs. The functional requirements of the deployed fact-checking process can be described as follows:

- **Input:** Our system accepts different facts related to person, organizations, and places and passes it for further processing.
- **Pre-Processing of Wikipedia pages and user inputs:** An execution of pre-processing pipeline with several methods of NLTK converts the inputs into a suitable format for further matching to execute the fact-checking process.
- **Fact-Checking:** By using Natural Language Processing techniques and Wikipedia pages, deployed system identifies necessary facts relevant to the user query.
- **Output:** Identify the fact as true or false and display the result to the user.

For non-functional requirements, our deployed system mainly based on the process-oriented approaches as it has given more emphasis on the design process.

Our developed system is mainly based on Python programming language which is easy to implement and can be easily integrated with other systems. Therefore, it is a user-friendly interface. Moreover, the response time of our deployed system is reasonable as it is able to process 100 queries by using a workstation with a very basic level of configuration. However, the response time will increase with the amount of input query.

Our deployed system is easily portable and can be accessed by using any web interfaces. Therefore, the deployed system can be easily handled by a user with minimum to a standard level of skills

2.2 System Architecture

In order to understand the functionality of the deployed system, the system architecture can be represented in two ways. One way is by describing the overall data process model and another way is by using a data flow model [10].

Data processing model:

The deployed fact-checking process mainly focus on the user queries and Wikipedia as a fact-checking medium. The system accepts the user queries, process it by using several NLP and IR methods and returns the result after necessary cross-checking with Wikipedia pages. The overall data process model can be represented as follows.

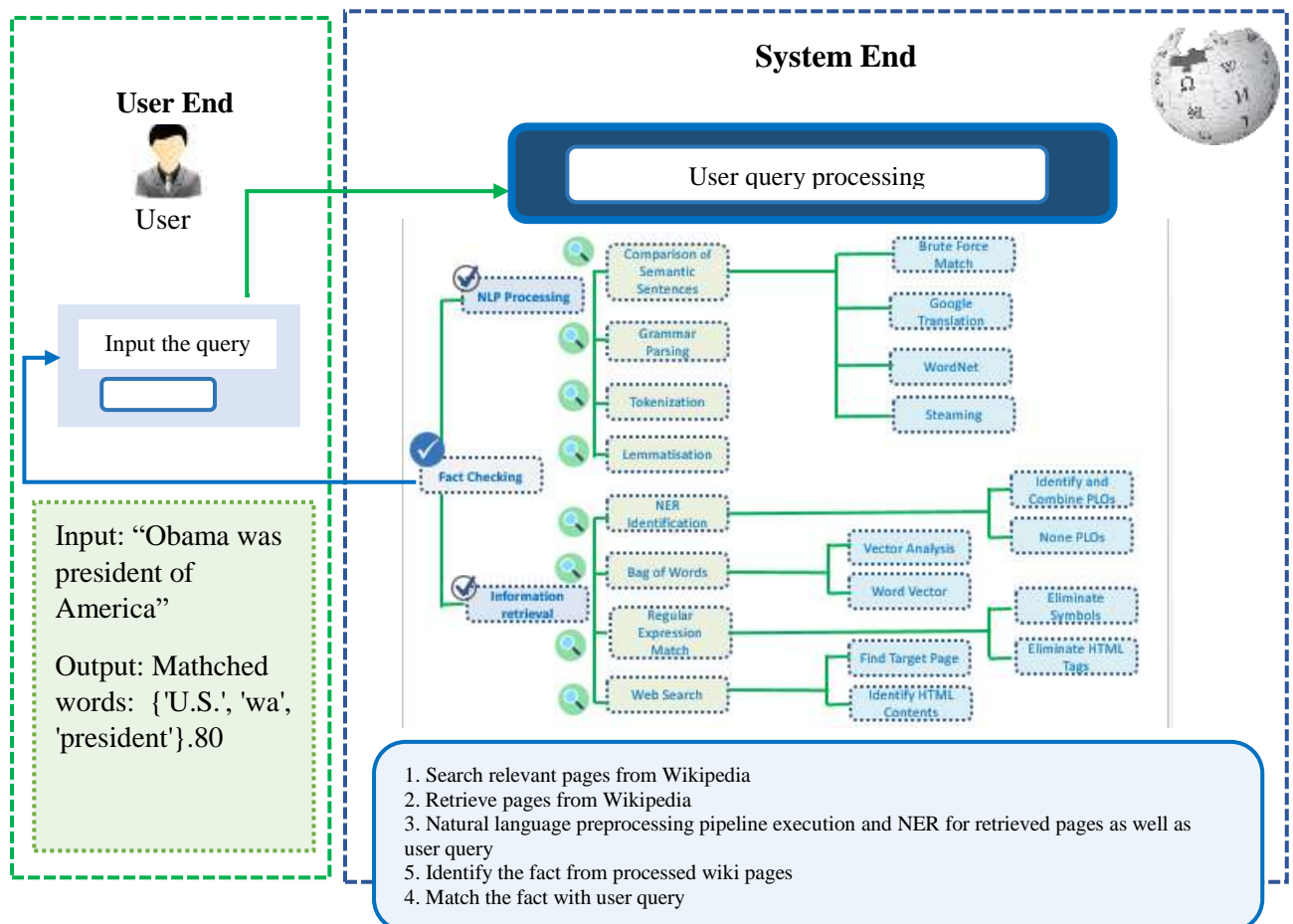


Figure 7: Overall Data Processing Model of deployed Fact Checking System based on Wikipedia.

The overall data flow (up to level 1) of deployed fact-checking software can be illustrated as follows.

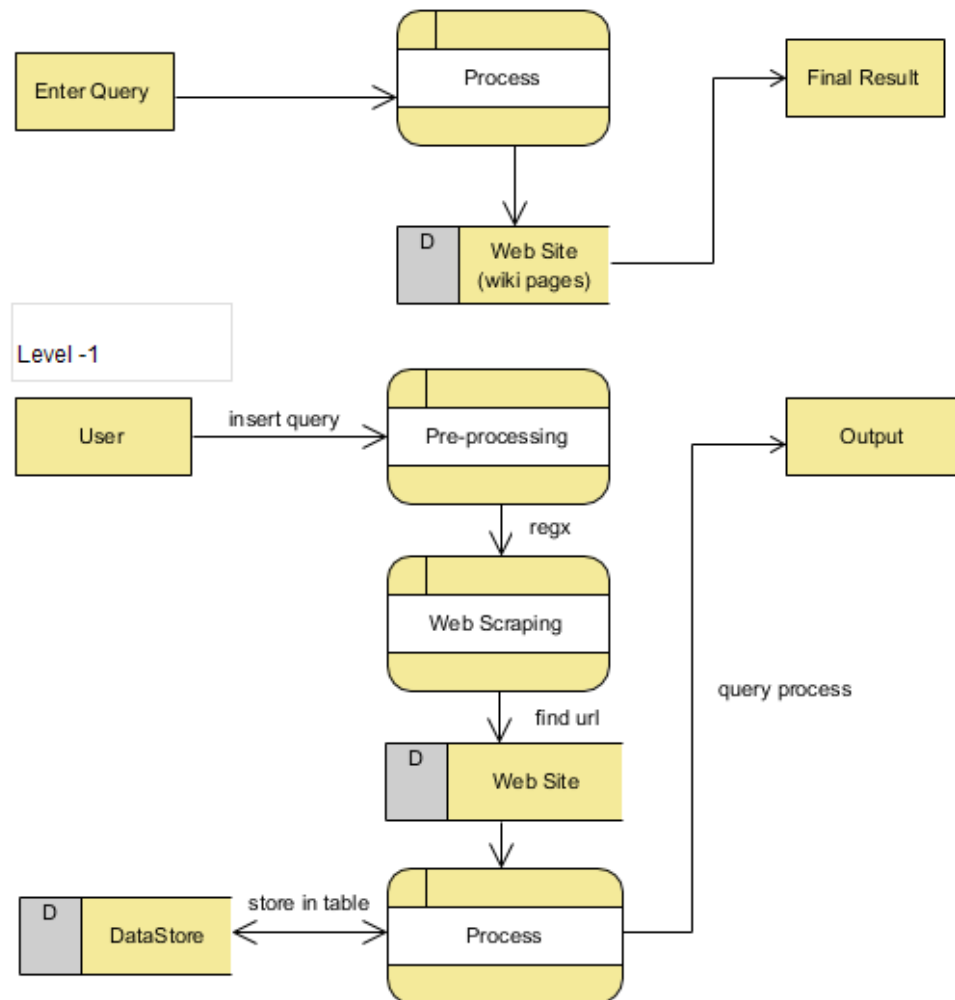


Figure 8: Level 1 Data Flow Diagram of deployed Fact Checking System based on Wikipedia [11]

Note: Above-mentioned diagram is based on the previously submitted “Software Requirements Specification” by Group 6.

3 Implementation

The implementation of the project has been done into two phases. After the initial feedback from the client software development has been initiated. Once the development of the first is over, the first version has been released for the testing as well as for the performance evaluation. After the necessary evaluation of the first version, several limitations have been identified. Based on the identified limitations, a second approach has been implemented to overcome the limitations. Moreover, through second implementation new implementation methodology has been adopted which eventually resulted in a better solution. Before getting the feedback from the client about the final solution, we also did different testing to ensure the effectiveness of the system. After the necessary feedback from the client, the final version has been released. Next three subsections will briefly describe the implementation methodology of two versions of the software as well as the evaluation result of the implementation.

3.1 Implementation methodology of first version

The first version of implementation is divided into two parts. First one is Named Entity Extraction from the user given input (fact) and the second one is the verification of the fact by using the online Wikipedia.

The execution of the program starts with the processing of fact given by the user. The fact provided by the user is passed to the named entity extraction method of fact class. The processing of the given fact begins at this stage. After that, the tokenization of the fact starts by using `nltk.word_tokenize`. Moreover, the tokenized fact pass to the `nltk.pos_tag` for necessary POS tagging. Once the pre-processing of user entered fact is done, the system will use the `nltk_ne_chunk` method to return a tree of named entity. Later it will pass to continuous chunk to extract the Named Entity from the tree. The Named Entity Recognition for the given facts ends at this stage.

The second execution part begins once the named entity extraction from the user provides fact is done and the program fetches the Wikipedia pages for every entity of the fact. It also stores all the relevant Wikipedia pages in the memory. When the necessary fetching is completed, the program will check whether the named entities of the fact occur on the same page or not. If it is on the same pages it will calculate

the ratio between common occurrences of both Named Entities and all comparisons for Named Entities. If the ratio is above the defined threshold (For the first implementation the threshold has been set at .80), the program will return a positive result which is 1 otherwise it will return 0 which indicates a negative result. The whole process can be summarized as follows.

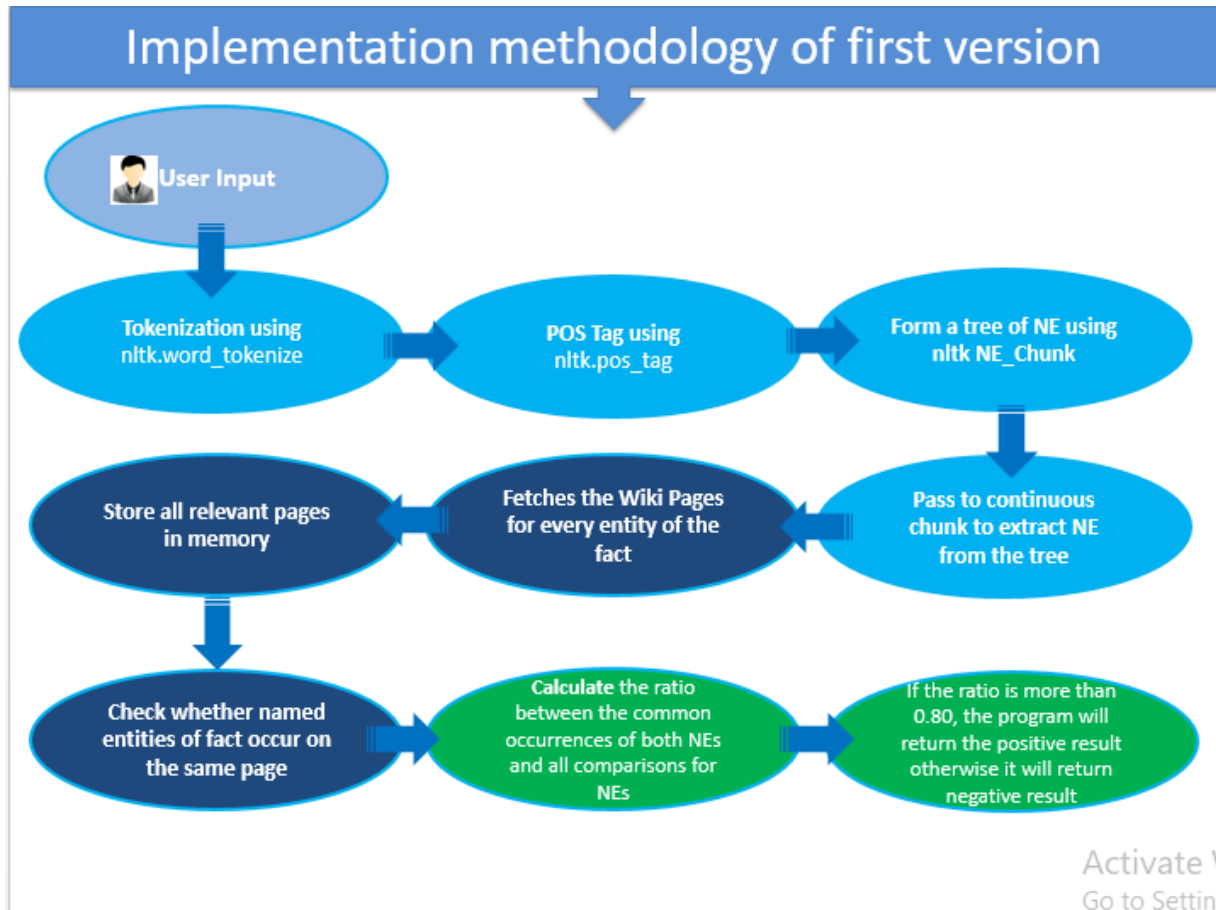


Figure 9: Implementation methodology of first version

The implemented first version has some limitation which is as follows:

- If the NEs present in more than one Wikipedia pages, the program will return 0 as it will not be able to decide which page needs to be fetched from the cache.
- For NEs of same pages, it will always return 1 regardless of the actual realization between them.
- The first version does not support any statistical method to identify the maximum similarities between multiple matches.
- It is unable to handle negative queries and synonyms.
- It also badly handles the date-related facts.

In order to overcome the above-mentioned problem, a better approach has been implanted which increases the test accuracy around 10%. For the first version, the accuracy rate for the fact-checking process is around 70%. The fact-checking accuracy is based on 100 test case data which has been extracted from Wikipedia Pages manually as well as automatically. A simple code has been developed to extract statements/facts from the Wikipedia pages.

3.2 Implementation methodology of final version

The final version of the software successfully overcomes most of the limitations of first implemented version. The second version also implement similar kind of approach with some additional methods which improves the performance of the software significantly. The overall implementation methodology for the final version can be summarized as follows.

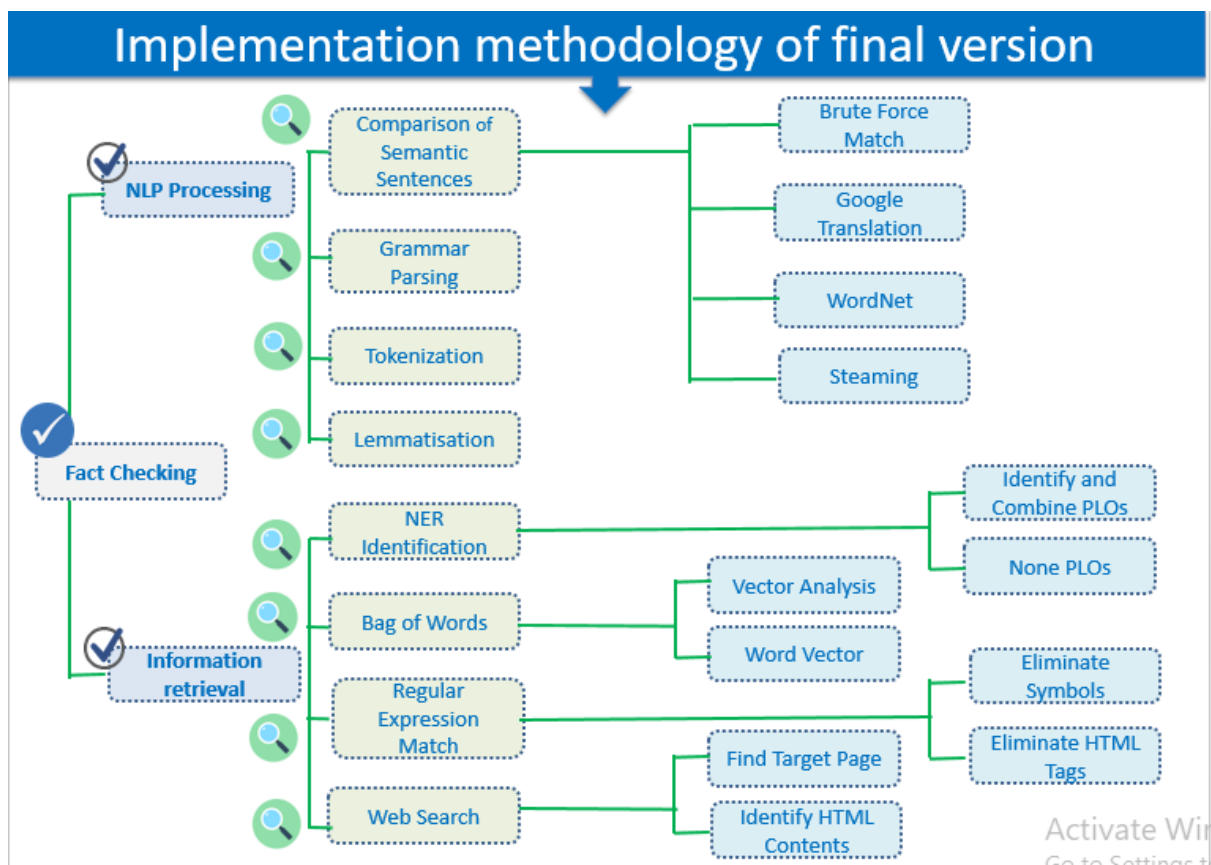


Figure 10: The implementation methodology of final version of the software.

The activity sequencing of the final version of the software can be described as follows.

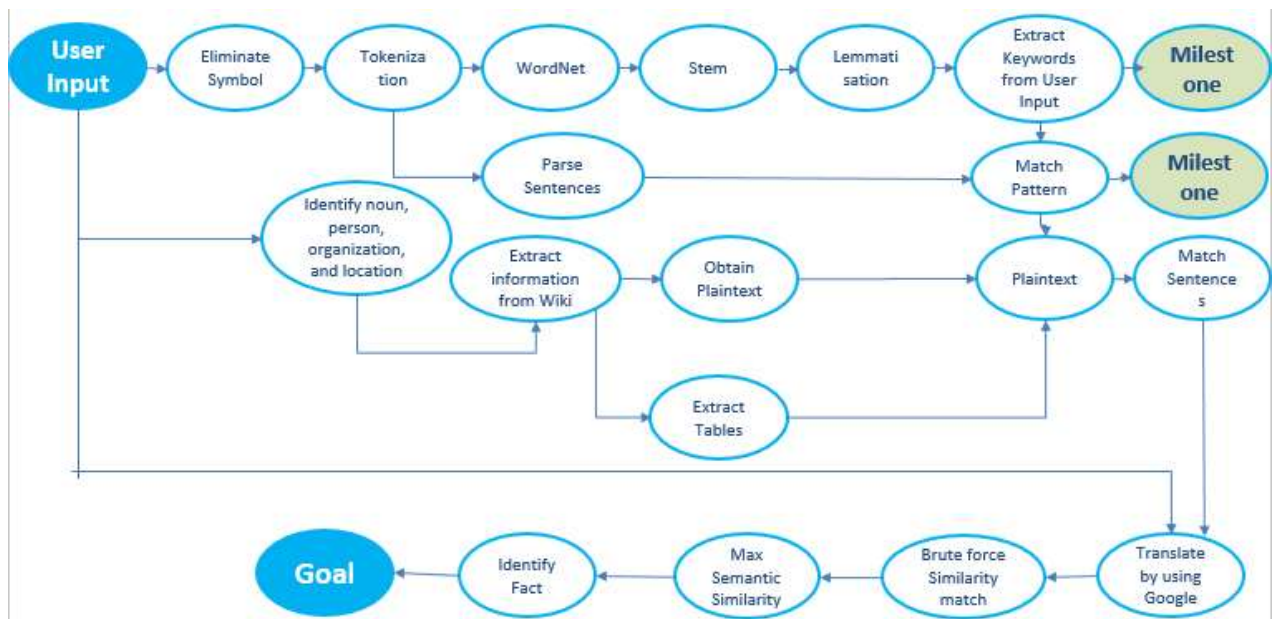


Figure 11: Activity sequence of the final version of the software.

The necessary execution steps of the final version can be described as follows.

The software starts working with the input of the user. First, it eliminates all the unnecessary symbols from the user input. After that, the tokenization process starts working. Tokenization is followed by the WordNet Synonyms checking, stemming, and lemmatisation methods. The whole pre-processing pipeline extract keywords from the user input. The system also parses the tokens into sentences. Simultaneously, the system identifies the person, organization, and locations from the user query and extracting information from Wikipedia based on that. The next step is to obtain the plaintext from the Wikipedia pages. After that, plaintexts extracted from Wikipedia matches against the user query (at this stage user query also represents as plaintext). Matches sentences (both Wikipedia sentences and user inputs) then forward to the Google Translate for necessary translation into another language. The final step is the brute force matching and calculation of the maximum semantic similarity. If the Max Semantic Similarity value is more than the threshold value the system will return a positive result otherwise it will return a negative result.

After implementing the above-mentioned methods the accuracy rate for the fact-checking process of final implementation for 100 test case is 0.79.

Average precision is: 0.788235294117647

3.3 Evaluation the performance of final version

In order to improve the performance of the final implementation, several methods have been implemented. One of the methods is the use of the regular expression. However, the regular expression also has some limitations. The limitations associated with a regular expression can be improved by using another method called a bag of words solutions. Below is the necessary example of how regular expression and bag of words increases the performance of the final software.

1



Fact given by a user

- Donald Trump performed a song with Megan Mullally at the 57th Primetime Emmy Awards in 2005.

Solutions provide by the systems

- **Regular Expression solution:**

'?:Donald Trump(?: .*performed) (?: .*song)'

- **Disadvantage:**

- ✓ The length of regular expression also extend with the length of sentence.
- ✓ Regular expression modification is required with the position change of a word.

- **Bag of words solutions**

In order to overcome previously mentioned problem with long sentences.

Input sentence by our system:

At 57th Emmy Awards, Donald Trump performed a song with Megan.



Output:

When all the input keywords are matched, the system will return it as a perfect match.




Another method to improve the performance of the final implementation is the use of Stemming or Lemmatizing. Moreover, the use of WordNet also significantly increased the performance of the fact-identification. Below is the example of how

Stemming/Lemmatizing and WordNet improves the performance of the final version of the software.

- 2  **Input :** After graduating from Columbia University in 1983, he worked as a community organizer in Chicago.
Question: graduate vs graduating
Solution: Stemming or Lemmatizing.
- 3  **Input :** David Bowie's death place is London.
However, our input sentence is: David Bowie was dead in London.
Solution: WordNet to paraphrase the sentence.

One unique approach to identify the similarities between sentences is the use of the translation method by using google translator. This approach is helpful to identify the more relevant fact in terms of similarity between translate meaning and original meaning. Use of the translate method significantly increases the performance of the final version of the implementation. The whole idea could be illustrated as follows.

- 4  **Input : (wrong input – Grammar related error)**
Wladyslaw Reymont's office is Nobel Prize in Literature.
Solution: Google translation.
- 1. Translation input to Chinese:**
Wladyslaw Reymont 的办公室是诺贝尔文学奖。
 - 2. Match for translated sentence (to Chinese):**
Wladyslaw Remont 在1924年的时候引人注目的获得了诺贝尔文学奖。
◦
 - 3. KMP/Brute Force Algorithm** to obtain the maximum matched length
Output: Our software will return one matched sentence
Władysław Reymont is Notable awards of Nobel Prize in Literature
1924.
If the rank is under the defined threshold, the system will refer the query as not matched.

After the necessary translation into another language with the Maximum similarities score for the user query “After graduating from Columbia University in 1983, he worked as a community organizer in Chicago” will return the following output.

No.	Sentence	Translation (ja)	Rank (ja)
1	After graduating from Columbia University in 1983, he worked as a community organizer in Chicago.	1983年にコロンビア大学を卒業した後、彼はシカゴでコミュニティオーガナイザーとして働いていました。	0.44
2	Later in 1981, he transferred as a junior to Columbia University in New York City, where he majored in political science with a specialty in international relations and in English literature and lived off-campus on West 109th Street.	1981年の後半に、彼はニューヨークのコロンビア大学に入学し、そこで国際関係と英語文学を専門とする政治科学を専攻し、西109丁目の学外に住んでいました。	0.28
3	Two years after graduating from Columbia, Obama was back in Chicago when he was hired as director of the Developing Communities Project, a church-based community organization originally comprising eight Catholic parishes in Roseland, West Pullman, and Riverdale on Chicago's South Side.	コロンビアを卒業して2年後、オバマ氏はシカゴのサウスサイドにあるローズランド、ウェストプルマン、リバーデールの8つのカトリック教区で構成される教会ベースのコミュニティ組織、Developing Communities Projectのディレクターとして雇われたときシカゴに戻りました。	0.2
4	He then left to attend graduate school on a scholarship at Harvard University, where he earned an M.A.	それから彼はハーバード大学の奨学金で大学院に通っていましたが、そこで彼は修士号を取得しました。	0.12

Table 1: Output of Maximum similarities score with the necessary translation into another language by using Google Translate.

4 Testing

Software testing is explained as a method to detect the error in our system and check whether the definite outcomes match the predictable results. In other words, testing is the development of assessing a system or its element(s) with the committed to discovery whether it pleases the stated requirements or not. Moreover, testing is performing a model in a direction to recognise any gaps, faults, or absent necessities in complication to the real requirements.

4.1 Questions related to testing

Few major questions related to testing are as follows.



Who does the testing?

Testing is depending on the development and the linked stake-holders of the project(s). In the Information technology (IT) business, big corporations have a team with responsibilities to assess the advanced software in the context of the agreed necessities. Moreover, creators also conduct testing which is called Unit Testing. In maximum cases, the following experts are involved in testing a system within their respective capacities.

- Software Tester
- Software Developer
- Project Lead/Manager
- End User



When to start the testing?

There is an important question regarding the starting time of testing. The right time to start the testing as early as possible when we start the project because the early start reduces the time to revise and cost and create error-free software that is sent to the

client. However, in (SDLC) Software Development Life Cycle, testing can be started since the Requirements Gathering stage and sustained until the end of the software.

Testing is also dependent upon the methods used to develop the software. For instance, if we used Waterfall strategies the testing is performed in the testing phase. On the other hand, if we use the incremental approach then testing is performed at each iteration phase and the final system can be tested at last.

Testing is performed in diverse forms at each stage of SDLC.

- The analysis or requirement and validation is also part of testing.
- Revising the plan in the design phase to resolve the error and to progress the design is likewise measured as testing.
- After completion of the code, the developer is testing their system is also considered as testing.



When to stop the testing?

When we stop testing is also a very difficult task because the testing is a never-ending process, and no one can create software which claim that it is 100% tested. For stopping testing, some facets are to be considered which are mentioned below.

- Deadlines of the project.
- Achievement of functional and code attention to a particular point.
- Bug frequency reached a minimum level and no high-priority bugs are recognised.
- Management decision.

4.2 Strategy and level of testing

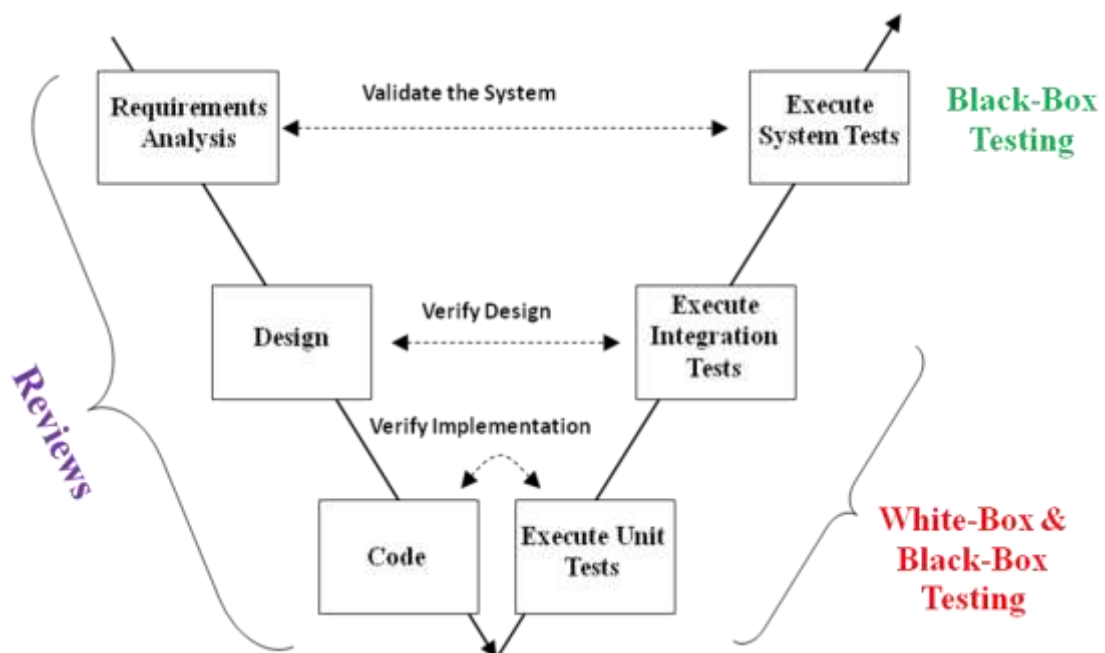
In order to implement an appropriate strategy to test out software, we have tried to address the below questions.



Verification – Are we building the product right?



Validation – Are we building the product right?



As we applied the V model in our project, the project development, testing, and coding work as parallel. In this model, we did the testing at each stage. The below-mentioned figure shows important categories and types of testing based on what we have done for our testing. Moreover, it mainly divided into two main categories called functional and non-functional testing.

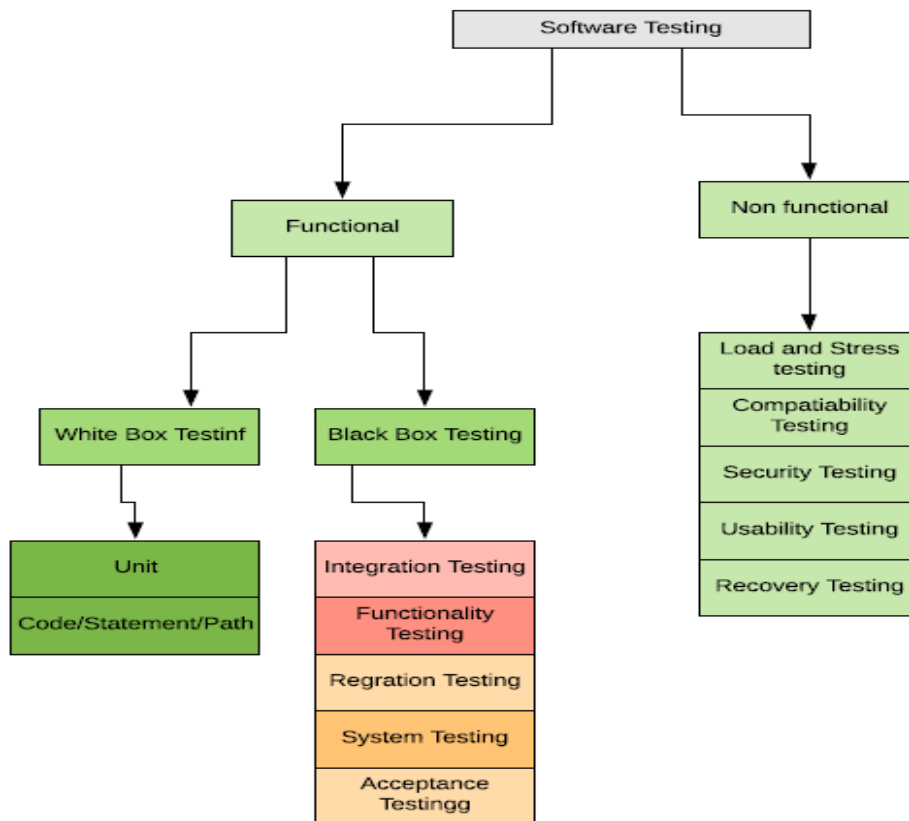


Figure 12: Type of our software testing.

Furthermore, based on the levels we have divided the testing into the following categories.

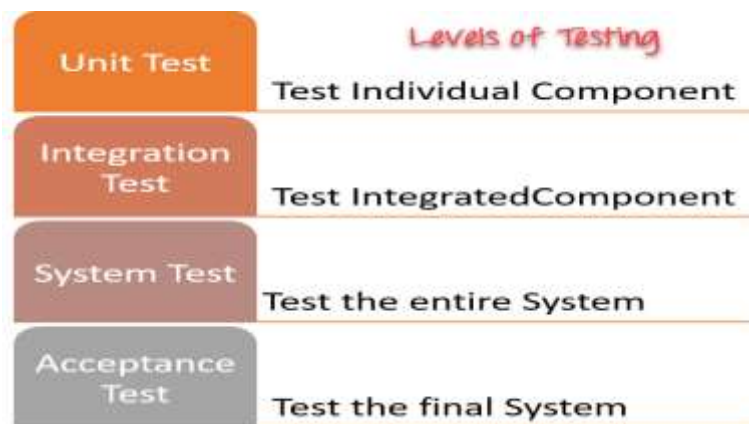


Figure 13: Different level of testing.

Each level of testing has some specific purpose. Next few subsections will illustrate different level-wise testing.

4.2.1 Unit testing

A Unit is an initial level of testing performs with the smallest unit. In this unit, we perform both types of testing (Black and white). In unit testing, the software is tested unit by unit to ensure that each part is functioning as expected.

In our project, we have done the unit testing on all individual modules. For instance, at the first phase of pre-processing, we performed unit testing and checked that the data we got from the web is clean data (without unnecessary HTML Tag, JavaScript, and other kinds of the script). To perform this unit test we have used both types of testing called black and white box testing. As the programmer is a tester at this stage, we did this process at the individual level and select the one which provided the best result. After applying the unit testing we got the clean data. Moreover, the test is performed by a programmer or occasionally it is performed with the help of a person who does not know about the project.

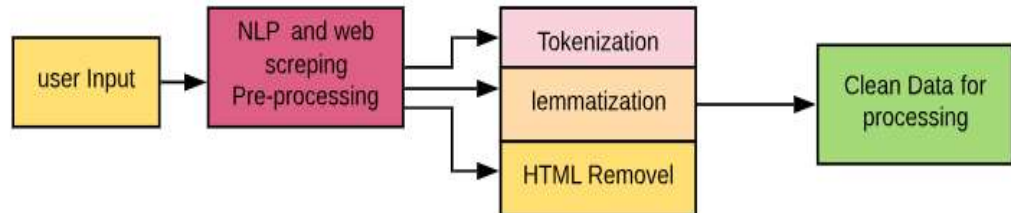


Figure 14: Unit Testing (black / White box) at pre-processing stage

Here in user input user entered the person name and whose information he wants.

4.2.2 Regression Testing

In regression testing, if we modify our system as per user/client, the functionality of the system has been checked by this testing.

After this stage, we apply the regression testing. According to the suggestion of our professor (client), we have changed our code to handle general facts instead of person-specific facts. Therefore, we have performed regression

testing and checked whether our system works well with pre-processing or not.

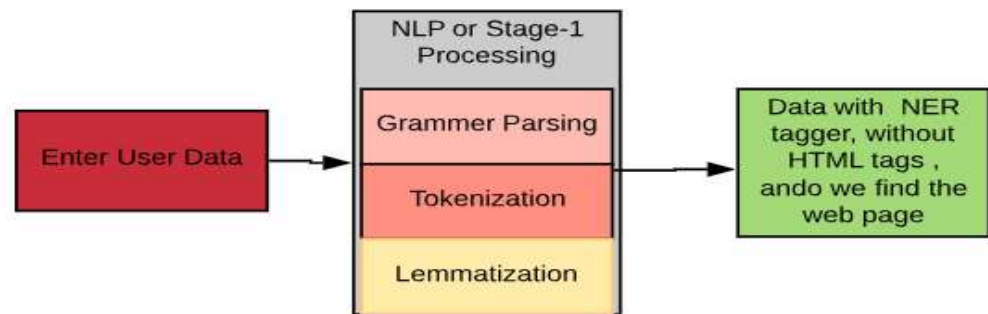


Figure 15: Regression testing (along with white box)

4.2.3 Integration Testing

In integration testing, components are joint and tested together. Integration testing is compulsory to expose any problems between merging units. Moreover, it is necessary to identify future difficulties regarding the merging units.

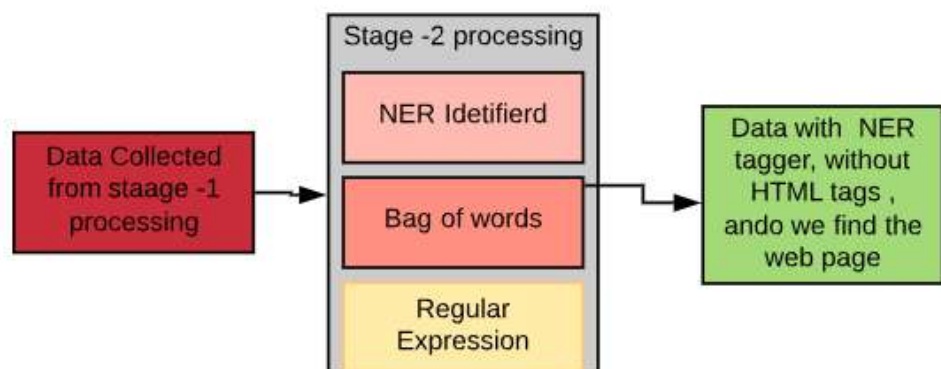


Figure 16: Integration testing (along with white box/black) after stage-1 processing

At the final stage, first of all, we have done the unit testing to verify that this final module is working properly. After that, we have performed the integration. At this stage, we have combined all of our units (pre-processing, data collect from stage-1; processing from stage-2) and have checked whether the system is working properly or not. In order to do this, we have used some examples, white box and black-box testing with the help of necessary people.

Now our system is created and ready for final release. Before the final demonstration in front of the client (In our case professor) we have done the alpha testing.

4.2.4 Alpha Testing

It is the most common kind of testing used in the industry. The main purpose of this testing is to recognize all likely issues or faults before delivering the product to the user. Alpha testing is performed after the software developed but before the Beta Testing.

Alpha testing is led by the developer at his own site. In-house virtual user situation can be formed for this type of testing. After this testing, our product is ready for the acceptance testing.

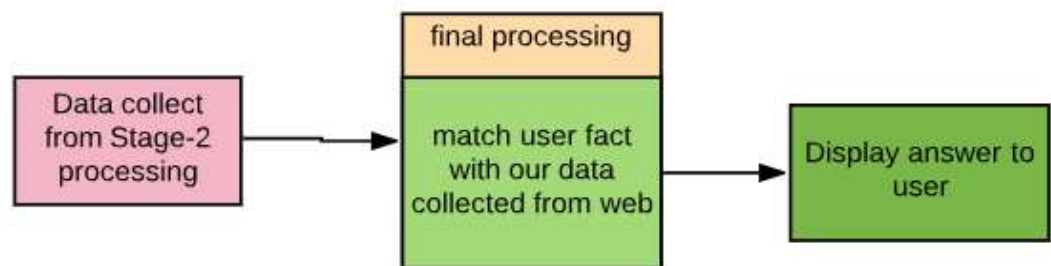


Figure 17: Alpha testing (along with white box/black) after stage-1 processing

4.2.5 Acceptance Testing

Acceptance Testing is done by the customer and confirms whether the movement of the product is as per the occupational necessities or not and if it is as per the requirements of the end client. The client receives the product (software) only when all the features and functionalities work as expected.

It is the final stage of the testing, afterward, the software is handover to the client. This is also called User Acceptance Testing (UAT). We have shown our final product to our client (Professor) and then he has checked our system with our provided test tools. Based on the satisfaction level of our client, we have decided to release our final version of the software.

4.3 Testing detail of the project

Some of the test details which we have done in our project are as follows.

S.no	Test case id	Testing Type	Condition	Result	Date
1	TC_NLTK_00	Unit / White Box	NLTK System checking	Pass	Feb 8,2019
2	TC_INPUT_001	Unit/White Box	Url Accessing checking	Fail	Feb,10, 2019
3	TC_INPUT_001	Unit/White Box along with Regression testing because in first attempt nit get required URL because user input wrong data	Url Accessing checking	pass	Feb,10, 2019
4	TC_WEB_002	Unit/White Box	Access data from URL	Partially Pass , because we get some noise data	Feb,14, 2019
5	TC_WEB_002	Unit/White Box and regression, because at this point professor advise us to go generic not for particular Person name	Access clean data from URL	Pass	Feb 15, 2019
6	TC_DATA_003	Integration/ White Box	Apply NER and other normalization Tools	Pass	Feb, 21 ,2019

7	TC_DATABASE_004	Unit/White Box	Upload first set of test cases	Pass	Feb 28, 2019
8	TC_IRC_0005	Integration/ White Box/ Black Box	Perform fact checking with test cases as well as with the help of other user (Friends)	Pass	Mar,03, 2019
9	TC_INTER_006	Integration/White Box Integrate all units ant test our system performance	After seeing the performance of our system all team member thinks to do something different and we introduce one intermediate language (Chines)	Pass	Mar, 06 2019
10	TC_INTER_007	System/ White Box and Black Box also (All Team Members along with Friends who help in Black box testing)	Check performance of new complete system	Pass	Mar, 08 2019
11	TC_ACCEOT_008	Acceptance Testing	All team members along with Professor	Pass	Mar,10 2019
12	TC_LOAD_000	Load Testing	Try With More cases up to 100 test cases	Pass	Mar,10 2019
13	TC_PORT_000	Portability	Test performance on other operating systems (Ubuntu, Mac)	Pass	Mar,13 2019
14	TC_PERF_000	Performance/ Beta testing	Test performance under of system with different OS and final testing	Pass with some limitati on	Mar,14 2019

5 Conclusions

Our developed system can quickly match sentences with matching score in ways that aid in the identification of important factual input provided by a user. Although our system has some limitations, better implementation can be done with the implementation of different approaches of fact-checking, further analysis, and sufficient time for the development. Next subsections will briefly address these issues.

5.1 System related limitations and possible way of improvement

Our developed system has some limitations which are illustrated as follows.

- A limited time period to develop the software.
- Our testing accuracy is less than **80%**. Therefore, it is needed to be improved.
- The problem of handling negative words.

For instance: If a user inputs the following query, our system will not be able to handle the negative words (“not”) properly,

Input: Barak Obama is not the president of America

Output: Fact is true

- Our final version only checks the fact against a single Wikipedia page.
- As we are using online Wikipedia, our system is not able to execute an offline request.

In order to overcome the above-mentioned limitations, we can implement the following things as a part of the future scope of the project.

- Offline Wikipedia (Wikipedia Dump) can be used to check a fact without any online bindings.
- With a proper time limit, a better solution can be implemented. Instead of online fact checking, we can combine online and offline together.
- We can build a database which stores extracted facts (facts as a sentence) at the local database. Moreover, we can use the database to implement some extra checking to develop a more robust solution.

- Along with Wikipedia other authentic sources, (For instance, BBC News, CNN news, different government related pages and many more) of information can be used to boost the performance of the implantation.
- The machine learning algorithm can be used to handle future user queries with a minimum level of dependency of online fact-checking basis like Wikipedia pages. In order to implement that, we can create a train and test dataset to implement suitable supervised algorithm with the manually annotated gold standard training set. Also, it will require a significant amount of human effort, a learned algorithm might able to identify the fact.

5.2 Effectiveness of the implementation

Although our implemented software has several shortfalls, it successfully handles user queries to identify the fact by using online Wikipedia. With proper threshold value setting our implemented software able to identify a fact around 80% accuracy. Same time, our system also able to find other information related to a user queries as far as it is present in the Wikipedia pages. However, multiple Wikipedia pages and negative words (in a user query) might reduce the performance significantly, use of google translation method increases the possibilities of matching facts. Our developed system might not perfect in all aspects but it is a reasonable solution which can be developed further to make a more effective fact-checking system.

5.3 Project Management

In order to produce a workable solution, to maintain all the task timely, and to make our project more successful we have tried to strictly follow the project management methodology. Initially, we have adopted Agile Methodology using the Scrum Method to implement the initial coding and test case preparation, due to time constraints we have switched to Waterfall to maximize the effectiveness of the implementation.

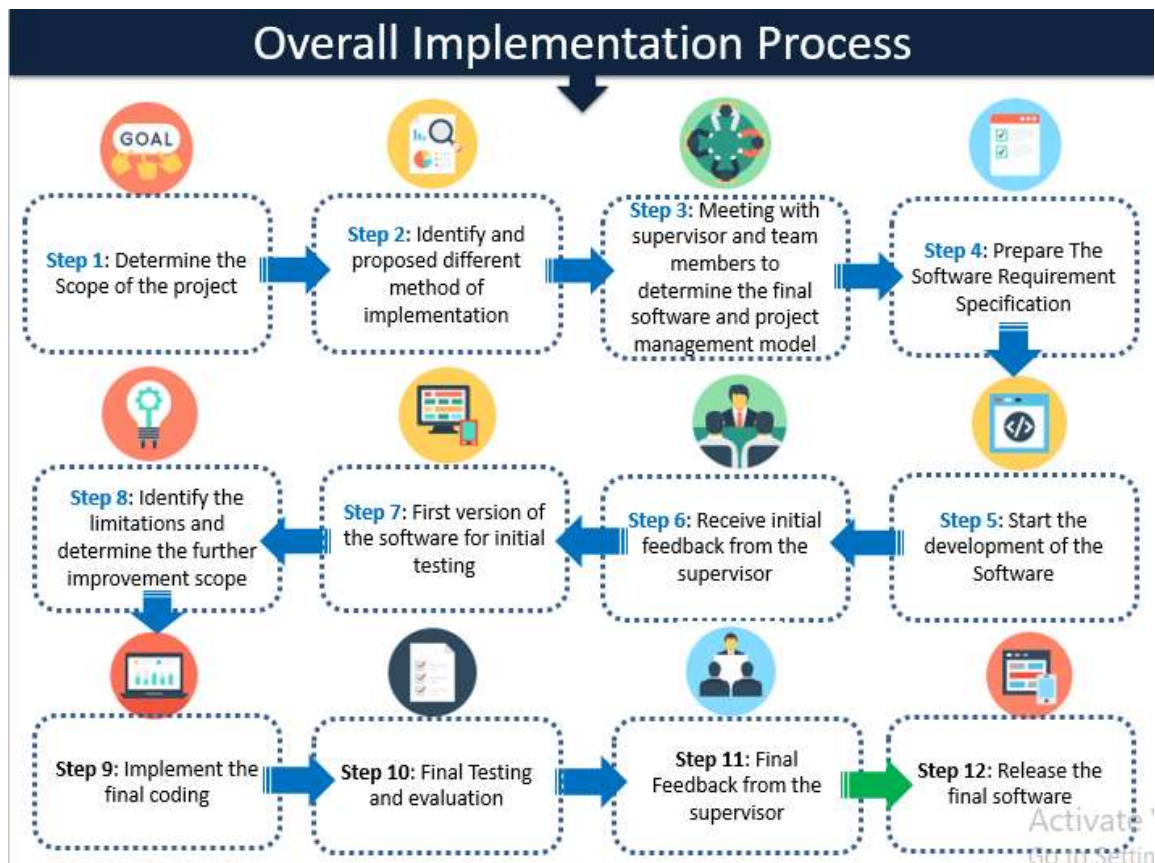


Figure 18: Overall process flow of our project.

While developing the software we have faced some difficulties in terms of time management, gathering authentic data for test cases, and effective software development within a short period of time. In order to overcome the challenges related to the project, we tried to maintain a strict schedule despite time constraint. We regularly organized the meetings to find out the possible solutions to overcome challenges. Moreover, we maintained the meeting minutes and agenda and circulated it within the group. We also contacted our client (Professor) to ensure the client level satisfaction as well as necessary improvement based on the received feedback.

Although there are several shortfalls related to the project, we tried to maximize the project scope, with all possible engagement of team members, regular discussion within the team as well as the client, and by strictly maintain the schedule of software development and testing. Our approach was to maximize the project scope as well as develop a workable solution by maintaining the appropriate project management method. In conclusion, it could be said that this project is a good approach towards development of software by ensuring the methodology of project management.

References

- [1] P. B. Brandtzaeg, and A. Folstad, "Trust and distrust in online fact-checking services", *Communications of the ACM*, vol. 60, no. 9, pp. 65-71, 2017. [Online]. Available: <https://dl.acm.org/citation.cfm?id=3122803>. [Accessed: February 2, 2019].
- [2] C. Conforti, M.T. Pilehvar, and N Collier, "Towards Automatic Fake News Detection: Cross-Level Stance Detection in News Articles", *Proceedings of the First Workshop on Fact Extraction and VERification (FEVER)*, pp. 40-49, 2018. [Online]. Available: <http://www.aclweb.org/anthology/W18-5507>. [Accessed: February 2, 2019].
- [3] V. Huynh, and P. Papotti, "Towards a Benchmark for Fact Checking with Knowledge Bases", *Companion Proceedings of the The Web Conference 2018*, pp. 1595-1598, 2018. [Online]. Available: <https://dl.acm.org/citation.cfm?doid=3184558>. [Accessed: February 2, 2019].
- [4] N. Hassan, F Arslan, C li, and M. Tremayne, "Toward Automated Fact-Checking: Detecting Check-worthy Factual Claims by ClaimBuster", *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1803-1812, 2017. [Online]. Available: <https://dl.acm.org/citation.cfm?id=3098131>. [Accessed: February 2, 2019].
- [5] "The State of Automated Factchecking", *fullfact.org*, 2016. [Online]. Available: <https://fullfact.org/blog/2016/aug/automated-factchecking/>. [Accessed: February 2, 2019].
- [6] D. Jimenez, "Towards Building an Automated Fact-Checking System", *ACM International Conference on Management of Data*, pp. 7-9, 2017. [Online]. Available: <https://dl.acm.org/citation.cfm?id=3055176>. [Accessed: February 2, 2019].
- [7] F. Wu, and D. S. Weld, "Autonomously Semantifying Wikipedia", *ACM conference on Conference on information and knowledge management*, pp. 41-50, 2017. [Online]. Available: <https://dl.acm.org/citation.cfm?id=1321449>. [Accessed: February 2, 2019].
- [8] F. Wu, and D. S. Weld, "Automatically Refining the Wikipedia Infobox Ontology", *7th international conference on World Wide Web*, pp. 635-644, 2008. [Online]. Available: <https://dl.acm.org/citation.cfm?id=1367583>. [Accessed: February 2, 2019].
- [9] J. Nothman, J. R. Curran, and T. Murphy, "Transforming Wikipedia into Named Entity Training Data", *Proceedings of the Australasian Language Technology Workshop*, vol. 6, pp. 124-132, 2008. [Online]. Available: <http://www.aclweb.org/anthology/U08-1016>. [Accessed: February 2, 2019].
- [10] T. A. Pender, "UML Weekend Crash Course", New York, Wiley Publishing, 2002.
- [11] Group 6, "Fact-checking based on Wikipedia", *Soft Requirement Specification*, 2019.

Appendix

- Contains detail code and output with necessary description of the code

Group Project - CE903

Fact Checking by using Wikipedia

Submitted by Group 6

First version and Final Version

1 Final Version

In [14]:

```
import re
import os
import nltk
import numpy as np
import pandas as pd
import csv, xlrd
from urllib import request
from nltk import sent_tokenize
from nltk import word_tokenize
from nltk.corpus import stopwords
from bs4 import BeautifulSoup
from bs4.element import Tag
from nltk.corpus import wordnet as wn
from itertools import chain, groupby
from nltk.tag import StanfordNERTagger
from nltk.stem import WordNetLemmatizer

wikipedia_base = 'https://en.wikipedia.org/wiki/'
#pronoun = ['he', 'she', 'his', 'her', 'their', 'He', 'She', 'His', 'Her',
'Their']
negative_words = ['no', 'not']

def open_target_url(subject):
    return wikipedia_base + subject

def obtain_target_text(url):
    try:
        html = request.urlopen(url).read().decode('utf-8')
        bs = BeautifulSoup(html, features = 'html.parser')
        attrs = {"class": 'mw-parser-output'}
        need = bs.find(name = 'div', attrs = attrs)
        raw_text = ''
        for tag in need.contents:
            if tag.name == 'p' :
                raw_text += tag.text.strip() + '\n'
        header = ''
        stat = []
        counter = 0
        for tag in need.contents:
            if tag.name == 'table':
```

```

        for tbody in tag.contents:
            if tbody.name == 'tbody':
                for tr in tbody:
                    features = ''
                    contents = ''
                    if tr.name == 'tr':
                        for th in tr:
                            if th.name == 'th':
                                if counter == 0:
                                    header = th.text
                                    counter += 1
                                else:
                                    features = re.sub('\n', ' ',
th.text)

                            if th.name == 'td':
                                contents = re.sub('\n', ' ',
th.text)

                        stat.append(header + ' is ' + features + '
of ' + contents + '.\n')
                    raw_text += ''.join(stat)
                    return raw_text
        except Exception as e:
            print('Can not find the target page in Wikipeddia.',url)
            return

def nerInNLTK(text):
    text1 = re.sub('(\.|\'s|\'')', '', text)
    try:
        st =
StanfordNERTagger('./NER_//english.all.3class.distsim.crf.ser.gz',
'./NER_//stanford-ner.jar')
        result = st.tag(text1.split())
        return result
    except Exception as e:
        print('did not detect any person, organization or location, please
check.')
```

```

        return re.sub(r'^\w', ' ', inpt)
def eliminate_symbols_1(inpt):
    return re.sub(r'^\w', '', inpt)
def eliminate_annotations(inpt):
    return re.sub(r'\[\S*\]', '', inpt)

def pre_process(sent):
    key_facts = []
    sent_token = nltk.word_tokenize(sent)
    pos_taged = nltk.pos_tag(sent_token)
    for word, tag in pos_taged:
        if tag.startswith('N') or tag.startswith('J') or
tag.startswith('V') or tag.startswith('R') or tag.startswith('C'):
            key_facts.append(word)
    return key_facts

def find_similar_word(pure_key_word):
    simis_word = []
    for word in pure_key_word:
        synsets = wn.synsets(word)
        for syn in synsets:
            for sy in syn.lemmas():
                simis_word.append(sy.name())
    simis_word += pure_key_word
    return set(simis_word)

# natural language process . paral: the sequence of the html store in the
url.txt

def pre_process1(words):
    sent_token = nltk.word_tokenize(words)
    patterns=
[(r'.*ing$', 'VBG'), (r'.*ed$', 'VBD'), (r'.*es$', 'VBZ'), (r'.*ould$', 'MD'), \
(r'.*\'s$', 'NN$'), (r'.*ly$', 'RB'), (r'.*s$', 'NNS'), (r'^-?[0-
9]+(.[0-9]+)?$', 'CD'), (r'.*', 'NN')]
    regexp_tagger = nltk.RegexpTagger(patterns)
    pos_taged = regexp_tagger.tag(sent_token)
    wnl = WordNetLemmatizer()
    after = []
    for lemma in pos_taged:
        if lemma[1].startswith('N'):
            after.append(wnl.lemmatize(lemma[0], 'n'))
            continue
        if lemma[1].startswith('J'):
            after.append(wnl.lemmatize(lemma[0], 'a'))
            continue
        if lemma[1].startswith('V'):
            after.append(wnl.lemmatize(lemma[0], 'v'))
            continue
        if lemma[1].startswith('R'):
            after.append(wnl.lemmatize(lemma[0], 'r'))
            continue
        else:
            after.append(wnl.lemmatize(lemma[0]))
    return after

def key_word_match(needed_text, entities, facts, confident_threshold):
    matched_sentences = []

```

```

sentences = sent_tokenize(needed_text)
pre_suf = entities.split()
pre_suf.append(entities)
pure_key_word = [fact for fact in facts if fact not in pre_suf]
simis_word = find_similar_word(pure_key_word)
for sentence in sentences:
    had_matched_word = []
    words = pre_process1(sentence)
    for word in words:
        if word in simis_word:
            had_matched_word.append(word)
    matched_lenth = len(set(had_matched_word))
    confidence = matched_lenth - confident_threshold
    neg_matched = [neg for neg in negative_words if neg in
had_matched_word]
    neg_input = [neg for neg in negative_words if neg in facts]
    if len(pure_key_word) <= confidence and neg_input == neg_matched:
        matched_sentences.append(sentence)
        print('Mathched ',matched_lenth, 'keywords on:', '
'.join(words))
        print('Mathched words: ', set(had_matched_word))
    return matched_sentences

# Imports the Google Cloud client library

from google.cloud import translate

def translation(traget_lan, sentence):
    # Instantiates a client
    translate_client = translate.Client()
    # The text to translate
    text = sentence
    # Translates some text into Chinese
    translation = translate_client.translate(text,
target_language=traget_lan)
    print(u'Translation: {}'.format(translation['translatedText']))
    return translation['translatedText']

def compare_translation_sent(sent1, sent2):
    sents1 = eliminate_symbols_1(sent1)
    sents2 = eliminate_symbols_1(sent2)
    sents_array_1 = [i for i in sents1]
    sents_array_2 = [j for j in sents2]
    counter = 0
    chunk_match_length = []
    for i in range(len(sents_array_2)):
        for j in range(len(sents_array_1)):
            if i <= (len(sents_array_2) - 1):
                if sents_array_2[i] == sents_array_1[j]:
                    counter += 1
                    chunk_match_length.append(counter)
                    i += 1
                    continue
            else:
                counter = 0
                continue
    if len(chunk_match_length) > 0:
        match_accuracy = sorted(chunk_match_length, reverse = True)[0] /
len(sents_array_2)
    else:

```

```

        match_accuracy = 0
    return match_accuracy

# please use pip install xlrd here.
def read_csv_files(file_name):
    df = pd.read_excel(os.getcwd() + '\\'+ file_name)
    stat = ['Fact_Statement']
    target = df[stat].values.astype(str).tolist()
    target1 = np.array(target).flatten()
    return target1
stat_lines = read_csv_files('Fact_Checking.xlsx')

for line in stat_lines:
    print(line)
    #inputs_trans = translation('zh-CN', line)
    key_facts = pre_process(line)
    key_facts1 = pre_process1(' '.join(key_facts))
    # recognize person, location and org
    ners = nerInNLTK(line)
    entities = combo_ners(ners)[0]
    sent_mean = {}
    counter = 0;
    for entitie in entities:
        target_url = open_target_url(entitie[1])
        target_text = obtain_target_text(target_url)
        if target_text == None:
            break
        target_text_1 = eliminate_annotations(target_text)
        matched_sentences = key_word_match(target_text_1, entitie[1],
key_facts1, 0)
        if len(matched_sentences) > 0:
            for sent in range(len(matched_sentences)):
                counter += 1
                print('Matched ', counter, ': ', matched_sentences[sent])
        else:
            print("No Matches", 'in Wikipedia page of', '\\'+
entitie[1], '\\'+)

    print()

#for sent in matched_sentences:
    #matched_trans = translation('zh-CN', sent)
    #mean_similarity = compare_translation_sent(matched_trans,
inputs_trans)
    #sent_mean[sent] = mean_similarity
    #max_mean = sorted(sent_mean.items(), key=lambda item: item[1],
reverse=True)[0]
    #if float(max_mean[1]) > 0.21 and counter == 0:
        #counter += 1
        #print('Matched the sentence with max semantic similarity of :',
max_mean, end='\\n\\n')
    #else:
        #print('You maybe worng!', end='\\n\\n')

```


- **First Version**
- **Required packages**

In order to run the below-mentioned code below package is mandatory

- wikipedia
- To install wikipedia nltk packages use below command
 - pip install wikipedia
- For more detail please go through the below link
 - <https://pypi.org/project/wikipedia/>
- For more information about nltk ne_chunk go through the below links
 - <https://www.nltk.org/api/nltk.chunk.html>
 - <https://www.kaggle.com/nltkdata/maxent-ne-chunker>
- To understand how perception tagger works please go through the below link.
 - <http://www.nltk.org/modules/nltk/tag/perceptron.html>

- **How the program works**

- The program is divided into two parts which are as follows.
 - Named Entity Extraction from the fact
 - Verify the fact using the on-line Wikipedia

- **Named Entity Extraction from the fact**

- The program starts working with the fact (provided by the user).
- The fact provided by the user is passed to the named entity extraction method of Fact class. The processing of given fact begins in this step.
- Tokenized the fact using nltk.word_tokenize. After that tokenized fact passed to nltk.pos_tag for POS Tag.
- After pre-processing (tokenization with POS Tag) of fact, it will use nltk ne_chunk method to return a tree of named entity.
- Later it will pass to continuous chunk to extract the NE from the tree (for more detail how it works please go through the below link https://www.programcreek.com/python/example/91258/nltk.ne_chunk)
- NER for given facts end here.

- **Verify the fact using the on-line Wikipedia**

- Once the named entity extraction from the facts (provided by the user) is done, the program fetches the Wikipedia pages for every entity of the fact (based on the user input) and stores all relevant pages in the memory.
- When the necessary fetching is completed, the program will check whether the named entities of the fact occur on the same page or not.
- After that it calculates the ratio between the common occurrences of both NEs and all comparisons for NEs. If the ratio is more than 0.80, the program will return the positive result otherwise it will return negative result.

Whole Working process can be summarized as follows

- Drawbacks of the program Below are some major drawbacks related to the program
 - If the NEs present in more than one Wikipedia pages, the program will return 0 as it will not be able to decide which page needs to be fetched from the cache.
 - For NEs of same pages it will always return 1 regardless of the actual realization between them.

In [10]:

```
import wikipedia
import nltk
import csv
from nltk.tree import Tree
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('punkt')

class NamedEntityExtractionOfFact:
    def __init__(self):
        pass

    def extract_named_entities_fact(self, sentence):
        sentence_processing = self.sentences_pre_processing(sentence)

        ne_chunks = nltk.ne_chunk(sentence_processing, binary=True)
        list_of_named_entities = self.get_continuous_chunks(ne_chunks)

        #print(ne_chunks)
        #print(list_of_named_entities)

        return list_of_named_entities

    # Tokenize and POS for the Facts
    def sentences_pre_processing(self, sentence):
        sentence = sentence.replace('\n', ' ')
        sentence = nltk.word_tokenize(sentence)
        sentence = nltk.pos_tag(sentence)

        return sentence

    # Identify the NEs and form a tree of NEs for the preprocessed facts
    @staticmethod
    def get_continuous_chunks(chunk):
        continuous_chunk = []
        current_chunk = []

        for c in chunk:
            # Create a tree for NEs
            if type(c) == Tree:
                current_chunk.append(" ".join([token for token, pos in
c.leaves()]))
            elif current_chunk:
                named_entity = " ".join(current_chunk)
                if named_entity not in continuous_chunk:
                    continuous_chunk.append(named_entity)
                    current_chunk = []
```

```

        else:
            continue

    if continuous_chunk:
        named_entity = " ".join(current_chunk)
        if named_entity not in continuous_chunk:
            continuous_chunk.append(named_entity)

    return continuous_chunk    # Extract the NEs from the tree

def process_fact(self, fact):
    self.extract_named_entities_fact(fact)

```

Identify and store the relevant wikipedia pages in the memory for every NEs of the fact

```
class WikipediaPageExtraction:
```

```

    def __init__(self):
        self.cache_wiki = {}
        pass

    def cached_wikipedia_entity(self, entity):
        wikipages = None
        if entity in self.cache_wiki:
            wikipages = self.cache_wiki[entity]
        else:
            try:
                wikipages = wikipedia.page(entity)
            except wikipedia.exceptions.DisambiguationError:
                print("Disambiguation error: " + entity)
                wikipages = None
            except:
                print("Exception Occured : " + entity)
                wikipages = None

            self.cache_wiki[entity] = wikipages

    return wikipages

```

```
class FactCheckingUsingWikiPages:
```

```

    def __init__(self):
        self.fact_named_entity_extraction = NamedEntityExtractionOfFact()

        self.wikipedia_page_extractor = WikipediaPageExtraction()
        pass

    def fact_checking(self, fact):

        # get named entities
        named_entities = self.fact_named_entity_extraction.extract_named_entities_fact(fact)

        named_entities_with_pages = {}

        for entity in named_entities:

```

```

        named_entities_with_pages[entity] = self.wikipedia_page_extractor.cached_wikipedia_entity(entity)

        count_of_common_occurrence = 0
        total_num_of_occurrences = 0
        entities_comparison = set()

        # Identify whether the NEs are from same pages
        for wiki_entity in named_entities:
            for fact_entity in named_entities:

                wiki_page = named_entities_with_pages[wiki_entity]

                if wiki_page is None:
                    continue

                wiki_and_fact_entities = wiki_entity + ',' + fact_entity
                wiki_and_fact_entities_reverse = fact_entity + ',' + wiki_entity

                if wiki_entity != fact_entity and not wiki_and_fact_entities in entities_comparison:

                    # check if entities exist together
                    if fact_entity in wiki_page.content:
                        count_of_common_occurrence += 1

                    total_num_of_occurrences += 1
                    entities_comparison.add(wiki_and_fact_entities)
                    entities_comparison.add(wiki_and_fact_entities_reverse)

            if total_num_of_occurrences == 0:
                return 0.0

            #Calculate the ratio
            percentage_of_similarity = count_of_common_occurrence / total_num_of_occurrences

            if percentage_of_similarity > 0.8:
                return 1.0

            else:
                return 0.0

# Read the input files (Facts provided by the user)

def main():
    inputfile = "fact.tsv"
    fact_checking_wiki = FactCheckingUsingWikiPages()

    with open(inputfile, encoding="latin-1") as tsvfile:
        next(tsvfile)

```

```

reader = csv.reader(tsvfile, delimiter='\t')

i = 0

for row in reader:
    if len(row) < 2:
        continue
    id = int(row[0])
    fact = row[1]

    estimated_val = float(fact_checking_wiki.fact_checking(fact
))
    i += 1
    print(str(i) + ": " + str(estimated_val))

if __name__ == "__main__":
    main()

```

2 Automatically Identification and Extraction of Facts (Statements) from a Wikipedia Page.

3 Required package

- In order to run the below-mentioned code below packages are mandatory
 - spacy
 - textacy
- To install spacy please go through the below page
 - <https://spacy.io/usage/>
- For linux/ubuntu
 - \$pip install -U spacy
- To install textacy go through the below-mentioned links
 - <https://pypi.org/project/textacy/>
 - <https://media.readthedocs.org/pdf/textacy/stable/textacy.pdf>

In [1]:

```

import nltk, re, requests
import spacy
import textacy.extract
from bs4 import BeautifulSoup
from nltk import pos_tag, ne_chunk
from spacy import displacy

```

```

spacy_eng_nlp_model = spacy.load('en_core_web_lg')

```

In [2]:

```

# Grab a perticular page from online wikipedia.

```

```

wiki_page = "Brexit"

```

```
extract_link = f"https://en.wikipedia.org/api/rest_v1/page/html/{wiki_page}"
#extract_link = f"https://en.wikipedia.org/wiki/{wiki_page}"
```

In [3]:

```
raw_data = requests.get(extract_link)
```

In [4]:

```
bsoup = BeautifulSoup(raw_data.text, "lxml")
extracted_text = bsoup.select("body")[0].get_text().strip()
```

In [5]:

```
# Identification of extraction of statement (can be treat as a fact) by using textacy.
```

```
texts = spacy_eng_nlp_model(extracted_text)
statements = textacy.extract.semistructured_statements(texts, wiki_page)
statements = list(statements)
```

```
print (statements)
```

```
[(Brexit, be, a factor for the slowdown in EU immigration), (Brexit, was, a warning for the EU), (Brexit, may be, good for terrorists and the Kremlin and bad for European security), (Brexit, is, only alternative to Chequers plan), (Brexit, are, the options), (Brexit, are, the options)]
```

In [6]:

```
#Process the text and strip out the [] from the text
```

```
def processing(txt):
    reg_exp = re.compile("\.?\[\d+\]?")
    txt = reg_exp.sub("", txt).strip()

    if txt[-1] == ".":
        txt = txt[0:-1]

    return txt
```

In [7]:

```
# Extract the fact
for fact_wiki in statements:
    subject, verb, fact = fact_wiki
    fact = processing(str(fact))
    print(f"{subject} {verb} {fact}.")
```

Brexit be a factor for the slowdown in EU immigration.

Brexit was a warning for the EU.

Brexit may be good for terrorists and the Kremlin and bad for European security.

Brexit is only alternative to Chequers plan.

Brexit are the options.

Brexit are the options.

Output of final version (without Google Translation)

Ivana Trump is the first wife of Donald Trump.
Mathched 4 keywords on: Ivana Marie Trump (née Zelníčková ; Czech : ,
born February 20 , 1949) is a Czech former model and businesswoman , w
ho wa the first wife of Donald Trump .
Mathched words: {'wife', 'Donald', 'is', 'first'}
Matched 1 :
Ivana Marie Trump (née Zelníčková; Czech: , born February 20, 1949) is
a Czech former model and businesswoman, who was the first wife of Donal
d Trump.
No Matches in Wikipedia page of " Donald Trump ."

Ivana Trump is the second wife of Barak Obama.
No Matches in Wikipedia page of " Ivana Trump ."
No Matches in Wikipedia page of " Barak Obama ."

The UK has a pertially regulated market economy.
No Matches in Wikipedia page of " UK ."

Barak Obama born August 4, 1961.
Mathched 4 keywords on: Barack Hussein Obama II (/bəˈrɑ:k huːˈseɪn ou
'bɑ:mə/ (listen) ; born August 4 , 1961) is an American attorney and
politician who serve a the 44th president of the United States from 200
9 to 2017 .
Mathched words: {'August', '4', '1961', 'born'}
Mathched 4 keywords on: Obama wa born on August 4 , 1961 , at Kapiolan
i Medical Center for Women and Children in Honolulu , Hawaii .
Mathched words: {'August', '4', '1961', 'born'}
Mathched 4 keywords on: Barack Obama is Born of Barack Hussein Obama I
I (1961-08-04) August 4 , 1961 (age 57) Honolulu , Hawaii , U.S.. B
arack Obama is Political party of Democratic .
Mathched words: {'August', '4', 'Born', '1961'}
Matched 1 :

Barack Hussein Obama II (/bəˈrɑ:k huːˈseɪn ouˈbɑ:mə/ (listen); born Aug
ust 4, 1961) is an American attorney and politician who served as the 4
4th president of the United States from 2009 to 2017.
Matched 2 : Obama was born on August 4, 1961, at Kapiolani Medical Ce
nter for Women and Children in Honolulu, Hawaii.
Matched 3 : Barack Obama is Born of Barack Hussein Obama II (1961-08-
04) August 4, 1961 (age 57)Honolulu, Hawaii, U.S..
Barack Obama is Political party of Democratic.

Barak Obama graduate from Columbia University.
Mathched 3 keywords on: After graduate from Columbia University in 198
3 , he work a a community organizer in Chicago .
Mathched words: {'Columbia', 'University', 'graduate'}
Matched 1 : After graduating from Columbia University in 1983, he wor
ked as a community organizer in Chicago.

Barak Obama graduate from University of Essex.
No Matches in Wikipedia page of " Barak Obama ."

Ivana Trump is the second wife of Donald Trump.
No Matches in Wikipedia page of " Ivana Trump ."
No Matches in Wikipedia page of " Donald Trump ."

Barac Oobama born August 4, 1961.

Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/Barac_Oobama

Melbourne is the most populous city and metropolitan area of the European Union and the second most populous in Europe.

No Matches in Wikipedia page of " Melbourne ."

No Matches in Wikipedia page of " European Union ."

No Matches in Wikipedia page of " Europe ."

Computer is made of Copper.

Obama was president of America.

Matched 3 keywords on: Obama left office and retire in January 2017 and currently reside in Washington , D.C. A December 2018 Gallup poll found Obama to be the most admired man in America for an unprecedented 11th consecutive year , although Dwight D. Eisenhower was selected most admired in twelve non-consecutive years .

Matched words: {'Washington', 'America', 'wa'}

Matched 3 keywords on: The Council was chaired by Senior Advisor to the President Valerie Jarrett .

Matched words: {'President', 'wa', 'chair'}

Matched 3 keywords on: During his July 2015 trip , Obama also was the first U.S. president ever to visit Kenya , which is the homeland of his father .

Matched words: {'U.S.', 'wa', 'president'}

Matched 1 : Obama left office and retired in January 2017 and currently resides in Washington, D.C. A December 2018 Gallup poll found Obama to be the most admired man in America for an unprecedented 11th consecutive year, although Dwight D. Eisenhower was selected most admired in twelve non-consecutive years.

Matched 2 : The Council was chaired by Senior Advisor to the President Valerie Jarrett.

Matched 3 : During his July 2015 trip, Obama also was the first U.S. president ever to visit Kenya, which is the homeland of his father.

Matched 3 keywords on: George Washington , who had led the revolutionary army to victory , was the first president elected under the new constitution .

Matched words: {'Washington', 'wa', 'president'}

Matched 3 keywords on: Barack Obama , the first African-American and multiracial president , was elected in 2008 amid the crisis , and subsequently passed stimulus measure and the Dodd-Frank Wall Street Reform and Consumer Protection Act in an attempt to mitigate its negative effect and ensure there would not be a repeat of the crisis .

Matched words: {'wa', 'president', 'Obama'}

Matched 4 : George Washington, who had led the revolutionary army to victory, was the first president elected under the new constitution.

Matched 5 : Barack Obama, the first African-American and multiracial president, was elected in 2008 amid the crisis, and subsequently passed stimulus measures and the Dodd-Frank Wall Street Reform and Consumer Protection Act in an attempt to mitigate its negative effects and ensure there would not be a repeat of the crisis.

Abdus Salam won the Nobel Prize.

Matched 3 keywords on: Salam 's notable achievement include the Pati-Salam model , magnetic photon , vector meson , Grand Unified Theory , work on supersymmetry and , most importantly , electroweak theory , for which he was awarded the Nobel Prize .

Mathched words: {'Nobel', 'award', 'Prize'}

Mathched 3 keywords on: For this achievement , Salam , Glashow , and Weinberg were award the Nobel Prize in Physics in 1979 .

Mathched words: {'Nobel', 'award', 'Prize'}

Mathched 3 keywords on: In 1979 , Salam wa award the 1979 Nobel Prize in Physics , along with Glashow and Weinberg , For their contribution t o the theory of the unify weak and electromagnetic interaction between elementary particles , include , inter alia , the prediction of the weak neutral current .

Mathched words: {'Nobel', 'award', 'Prize'}

Matched 1 : Salam's notable achievements include the Pati-Salam model , magnetic photon, vector meson, Grand Unified Theory, work on supersymmetry and, most importantly, electroweak theory, for which he was awarded the Nobel Prize.

Matched 2 : For this achievement, Salam, Glashow, and Weinberg were awarded the Nobel Prize in Physics in 1979.

Matched 3 : In 1979, Salam was awarded the 1979 Nobel Prize in Physics, along with Glashow and Weinberg, For their contributions to the theory of the unified weak and electromagnetic interaction between elementary particles, including, inter alia, the prediction of the weak neutral current.

London has been a major settlement for two millennia.

Mathched 6 keywords on: Standing on the River Thames in the south-east of England , at the head of it 50-mile (80 km) estuary lead to the North Sea , London ha been a major settlement for two millennium .

Mathched words: {'millennium', 'ha', 'been', 'two', 'major', 'settlement'}

Matched 1 : Standing on the River Thames in the south-east of England , at the head of its 50-mile (80 km) estuary leading to the North Sea, London has been a major settlement for two millennia.

London was the world's most populous city from c. 1831 to 1925.

Mathched 8 keywords on: [note 4] London wa the world 's most populous city from c. 1831 to 1925 .

Mathched words: {'most', 'c.', 'city', 'wa', '1831', 'world', '1925', 'populous'}

Matched 1 : [note 4] London was the world's most populous city from c . 1831 to 1925.

London was beyond all comparison the largest town in England.

Mathched 5 keywords on: By the 11th century , London wa beyond all comparison the largest town in England .

Mathched words: {'comparison', 'wa', 'town', 'largest', 'England'}

Matched 1 : By the 11th century, London was beyond all comparison the largest town in England.

No Matches in Wikipedia page of " England ."

London was the focus of the Peasants' Revolt in 1381.

Mathched 5 keywords on: London wa the focus of the Peasants ' Revolt in 1381 .

Mathched words: {'wa', 'Peasants', 'Revolt', 'focus', '1381'}

Matched 1 : London was the focus of the Peasants' Revolt in 1381.

London was also a centre of England's Jewish population before their expulsion by Edward I in 1290.

Mathched 9 keywords on: London wa also a centre of England 's Jewish population before their expulsion by Edward I in 1290 .

Mathched words: {'wa', 'population', 'Edward', 'Jewish', 'centre', 'expulsion', 'England', 'also', '1290'}

Matched 1 : London was also a centre of England's Jewish population before their expulsion by Edward I in 1290.

No Matches in Wikipedia page of " England ."

No Matches in Wikipedia page of " Edward I ."

London was still very compact.

Mathched 4 keywords on: By the end of the Tudor period in 1603 , London wa still very compact .

Mathched words: {'compact', 'very', 'still', 'wa'}

Matched 1 : By the end of the Tudor period in 1603, London was still very compact.

London was the world's largest city from c.1831 to 1925.

Mathched 6 keywords on: According to Samuel Johnson : London wa the world 's largest city from c.1831 to 1925 .

Mathched words: {'city', 'wa', 'world', '1925', 'largest', 'c.1831'}

Matched 1 : According to Samuel Johnson:

London was the world's largest city from c.1831 to 1925.

London is vulnerable to flooding.

Mathched 3 keywords on: The Thames is a tidal river , and London is vulnerable to flood .

Mathched words: {'is', 'flood', 'vulnerable'}

Matched 1 : The Thames is a tidal river, and London is vulnerable to flooding.

London is "one of the World's Greenest Cities" with more than 40 per cent green space or open water.

Mathched 13 keywords on: The London Natural History Society suggest that London is `` one of the World 's Greenest Cities ' ' with more than 40 per cent green space or open water .

Mathched words: {'or', 'water', 'space', '40', 'Cities', 'is', 'one', 'World', 'green', 'open', 'Greenest', 'more', 'cent'}

Matched 1 : The London Natural History Society suggest that London is "one of the World's Greenest Cities" with more than 40 per cent green space or open water.

London is the most populous city and metropolitan area of the European Union and the second most populous in Europe.

No Matches in Wikipedia page of " London ."

No Matches in Wikipedia page of " European Union ."

No Matches in Wikipedia page of " Europe ."

London been Christian, and has a large number of churches, particularly in the City of London.

Mathched 9 keywords on: London ha traditionally been Christian , and ha a large number of church , particularly in the City of London .

Mathched words: {'particularly', 'number', 'ha', 'been', 'church', 'Christian', 'City', 'and', 'large'}

Matched 1 : London has traditionally been Christian, and has a large number of churches, particularly in the City of London.

Mathched 9 keywords on: London ha traditionally been Christian , and ha a large number of church , particularly in the City of London .

Mathched words: {'particularly', 'number', 'ha', 'been', 'church', 'Christian', 'City', 'and', 'large'}

Matched 2 : London has traditionally been Christian, and has a large number of churches, particularly in the City of London.

London is the second-largest Hindu temple in England and Europe.

No Matches in Wikipedia page of " London ."

No Matches in Wikipedia page of " England ."

No Matches in Wikipedia page of " Europe ."

London is also home to sizeable Muslim, Hindu, Sikh, and Jewish communities.

Matched 10 keywords on: London is also home to sizeable Muslim , Hindu , Sikh , and Jewish communities .

Matched words: {'Sikh', 'Hindu', 'home', 'is', 'Muslim', 'Jewish', 'sizeable', 'also', 'and', 'communities'}

Matched 1 : London is also home to sizeable Muslim, Hindu, Sikh, and Jewish communities.

London is also home to 44 Hindu temples, including the BAPS Shri Swaminarayan Mandir London.

Matched 11 keywords on: London is also home to 44 Hindu temples , include the BAPS Shri Swaminarayan Mandir London .

Matched words: {'Swaminarayan', 'Hindu', 'Mandir', 'include', 'home', 'is', 'BAPS', '44', 'temples', 'Shri', 'also'}

Matched 1 : London is also home to 44 Hindu temples, including the BAPS Shri Swaminarayan Mandir London.

Matched 11 keywords on: London is also home to 44 Hindu temples , include the BAPS Shri Swaminarayan Mandir London .

Matched words: {'Swaminarayan', 'Hindu', 'Mandir', 'include', 'home', 'is', 'BAPS', '44', 'temples', 'Shri', 'also'}

Matched 2 : London is also home to 44 Hindu temples, including the BAPS Shri Swaminarayan Mandir London.

London is the world's most expensive office market for the last three years according to world property journal (2015) report.

No Matches in Wikipedia page of " London ."

London is one of the pre-eminent financial centres of the world as the most important location for international finance.

Matched 11 keywords on: London is one of the pre-eminent financial centre of the world a the most important location for international finance .

Matched words: {'most', 'international', 'financial', 'world', 'is', 'location', 'important', 'one', 'centre', 'finance', 'pre-eminent'}

Matched 1 : London is one of the pre-eminent financial centres of the world as the most important location for international finance.

London is the world top city destination as ranked by TripAdvisor users .

Matched 8 keywords on: As of 2016 London is the world top city destination a rank by TripAdvisor user .

Matched words: {'destination', 'city', 'world', 'top', 'is', 'user', 'TripAdvisor', 'rank'}

Matched 1 : As of 2016 London is the world top city destination as ranked by TripAdvisor users.

London is a major international air transport hub with the busiest city airspace in the world.

Mathched 10 keywords on: London is a major international air transport hub with the busiest city airspace in the world .
Mathched words: {'city', 'international', 'world', 'transport', 'is', 'airspace', 'hub', 'major', 'busiest', 'air'}
Matched 1 : London is a major international air transport hub with the busiest city airspace in the world.

London is the centre of the National Rail network, with 70 per cent of rail journeys starting or ending in London.
Mathched 12 keywords on: London is the centre of the National Rail network , with 70 per cent of rail journey start or end in London .
Mathched words: {'or', 'end', 'Rail', 'network', '70', 'rail', 'journey', 'is', 'National', 'start', 'centre', 'cent'}
Matched 1 : London is the centre of the National Rail network, with 70 per cent of rail journeys starting or ending in London.
Mathched 12 keywords on: London is the centre of the National Rail network , with 70 per cent of rail journey start or end in London .
Mathched words: {'or', 'end', 'Rail', 'network', '70', 'rail', 'journey', 'is', 'National', 'start', 'centre', 'cent'}
Matched 2 : London is the centre of the National Rail network, with 70 per cent of rail journeys starting or ending in London.

London is home to designers Vivienne Westwood, Galliano, Stella McCartney, Manolo Blahnik, and Jimmy Choo, among others.
Mathched 14 keywords on: London is home to designer Vivienne Westwood , Galliano , Stella McCartney , Manolo Blahnik , and Jimmy Choo , among others ; it renowned art and fashion school make it an international centre of fashion alongside Paris , Milan , and New York City .
Mathched words: {'home', 'designer', 'Vivienne', 'is', 'Blahnik', 'Choo', 'McCartney', 'Stella', 'Galliano', 'Manolo', 'Westwood', 'others', 'Jimmy', 'and'}
Matched 1 : London is home to designers Vivienne Westwood, Galliano, Stella McCartney, Manolo Blahnik, and Jimmy Choo, among others; its renowned art and fashion schools make it an international centre of fashion alongside Paris, Milan, and New York City.
No Matches in Wikipedia page of " Vivienne ."
Can not find the target page in Wikipeadia. [https://en.wikipedia.org/wiki/Stella McCartney](https://en.wikipedia.org/wiki/Stella_McCartney), Manolo

London is a major centre for television production, with studios including BBC Television Centre, The Fountain Studios and The London Studios.
No Matches in Wikipedia page of " London ."

London is the "greenest city" in Europe with 35,000 acres of public parks, woodlands and gardens.
Mathched 11 keywords on: A 2013 report by the City of London Corporation said that London is the `` greenest city '' in Europe with 35,000 acres of public park , woodland and garden .
Mathched words: {'acres', 'city', 'greenest', '35,000', 'garden', 'woodland', 'is', 'Europe', 'public', 'park', 'and'}
Matched 1 : A 2013 report by the City of London Corporation said that London is the "greenest city" in Europe with 35,000 acres of public parks, woodlands and gardens.
No Matches in Wikipedia page of " Europe ."

London is not the capital of England, as England does not have its own government.
No Matches in Wikipedia page of " London ."

No Matches in Wikipedia page of " England ."

London is World's Most Expensive City | PropertyTime.

Mathched 7 keywords on: [note 2] London is consider to be one of the world 's most important global cities and ha been term the world 's mos t powerful , most desirable , most influential , most visit , most expe nsive , innovative , sustainable , most investment friendly , most popu lar for work , and the most vegetarian friendly city in the world .

Mathched words: {'most', 'city', 'be', 'expensive', 'world', 'is', 'gl obal'}

Matched 1 : [note 2]

London is considered to be one of the world's most important global cit ies and has been termed the world's most powerful, most desirable, mo st influential, most visited, most expensive, innovative, sustainable, most investment friendly, most popular for work, and the most vegetaria n friendly city in the world.

London is 'the most desirable city in the world to work in.

Mathched 8 keywords on: [note 2] London is consider to be one of the world 's most important global cities and ha been term the world 's mos t powerful , most desirable , most influential , most visit , most expe nsive , innovative , sustainable , most investment friendly , most popu lar for work , and the most vegetarian friendly city in the world .

Mathched words: {'desirable', 'most', 'city', 'be', 'world', 'is', 'gl obal', 'work'}

Matched 1 : [note 2]

London is considered to be one of the world's most important global cit ies and has been termed the world's most powerful, most desirable, mo st influential, most visited, most expensive, innovative, sustainable, most investment friendly, most popular for work, and the most vegetaria n friendly city in the world.

London is on fire as the West's fastest growing city.

No Matches in Wikipedia page of " London ."

London is the world capital of the 21st century.

No Matches in Wikipedia page of " London ."

London is world capital of culture.

Mathched 4 keywords on: [note 2] London is consider to be one of the world 's most important global cities and ha been term the world 's mos t powerful , most desirable , most influential , most visit , most expe nsive , innovative , sustainable , most investment friendly , most popu lar for work , and the most vegetarian friendly city in the world .

Mathched words: {'world', 'be', 'is', 'global'}

Mathched 4 keywords on: Globally , the city is amongst the big four fa shion capital of the world , and accord to official statistic , London is the world 's third busiest film production centre , present more liv e comedy than any other city , and ha the biggest theatre audience of a ny city in the world .

Mathched words: {'world', 'live', 'is', 'capital'}

Matched 1 : [note 2]

London is considered to be one of the world's most important global cit ies and has been termed the world's most powerful, most desirable, mo st influential, most visited, most expensive, innovative, sustainable, most investment friendly, most popular for work, and the most vegetaria n friendly city in the world.

Matched 2 : Globally, the city is amongst the big four fashion capitals of the world, and according to official statistics, London is the world's third busiest film production centre, presents more live comedy than any other city, and has the biggest theatre audience of any city in the world.

London is the HR centre of opportunity in the UK.
No Matches in Wikipedia page of " London ."
No Matches in Wikipedia page of " UK ."

Siena is Italo Calvino's death place.
No Matches in Wikipedia page of " Siena ."
No Matches in Wikipedia page of " Italo Calvino ."

Walter Jon Williams is Aristoi (novel)'s author.
No Matches in Wikipedia page of " Walter Jon Williams ."

Harry Martinson's award is Nobel Prize in Literature.
Matched 5 keywords on: Harry Martinson is Notable award of Nobel Prize in Literature 1974 (share with Eyvind Johnson) .
Matched words: {'award', 'Prize', 'is', 'Nobel', 'Literature'}
Matched 1 : Harry Martinson is Notable awards of Nobel Prize in Literature 1974 (shared with Eyvind Johnson) .

A Hard Day's Night (film) stars Bridgette Wilson
No Matches in Wikipedia page of " Bridgette Wilson ."

Ayrton Senna's birth place is Bologna.
Matched 4 keywords on: This is by reference to the following recollection given by Symonds in an interview in 2014 , to mark the 20th anniversary of Senna 's death : That season , Senna took two more podium finishes—third at the British and Portuguese Grands Prix—and place 9th in the Drivers ' Championship with 13 point overall .
Matched words: {'point', 'place', 'follow', 'is'}
Matched 4 keywords on: A shortened version of the Adelaide circuit (which remains the site of Senna 's last Formula One win) and the chicane remain in use for local motorsport event , and a commemorative concrete plaque installed in 1995 , bear Senna 's signature and hand print , is also located there .
Matched words: {'locate', 'bear', 'is', 'site'}
Matched 1 : This is by reference to the following recollection given by Symonds in an interview in 2014, to mark the 20th anniversary of Senna's death:
That season, Senna took two more podium finishes—third at the British and Portuguese Grands Prix—and placed 9th in the Drivers' Championship with 13 points overall.
Matched 2 : A shortened version of the Adelaide circuit (which remains the site of Senna's last Formula One win) and the chicane remain in use for local motorsport events, and a commemorative concrete plaque installed in 1995, bearing Senna's signature and hand prints, is also located there.
No Matches in Wikipedia page of " Bologna ."

Israel is Menachem Begin's team.
No Matches in Wikipedia page of " Israel ."
Matched 3 keywords on: Project Renewal is still being implemented today for at-risk communities in Israel .
Matched words: {'Israel', 'be', 'is'}

Matched 1 : Project Renewal is still being implemented today for at-risk communities in Israel.

The Poison Belt's author is Arthur Conan Doyle.
No Matches in Wikipedia page of " Arthur Conan Doyle ."

Wladyslaw Reymont's office is Nobel Prize in Literature.
Matched 5 keywords on: Władysław Reymont is Notable award of Nobel Prize in Literature 1924 .
Matched words: {'award', 'Prize', 'is', 'Nobel', 'Literature'}
Matched 1 : Władysław Reymont is Notable awards of Nobel Prize in Literature 1924 .

A Hard Day's Night (film) stars Vanessa Hudgens
Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/Vanessa_Hudgens

Sebastian Koch's spouse is Mario Lopez.
No Matches in Wikipedia page of " Sebastian Koch ."
No Matches in Wikipedia page of " Mario Lopez ."

Edgar Allan Poe's death place is Baltimore.
Matched 4 keywords on: The earliest survive home in which Poe live is in Baltimore , preserve a the Edgar Allan Poe House and Museum .
Matched words: {'live', 'home', 'is', 'Baltimore'}
Matched 1 : The earliest surviving home in which Poe lived is in Baltimore, preserved as the Edgar Allan Poe House and Museum.
No Matches in Wikipedia page of " Baltimore ."

Stephen Harper's birth place is Toronto.
Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/Stephen_Harper

Non-Stop (novel)'s author is Tanith Lee.
No Matches in Wikipedia page of " Tanith Lee ."

Christian I of Denmark's birth place is Oldenburg (Oldenburg).
No Matches in Wikipedia page of " Denmark ."
No Matches in Wikipedia page of " Oldenburg ."

David Bowie's death place is London.
No Matches in Wikipedia page of " David Bowie ."
No Matches in Wikipedia page of " London ."

Level 7's award is Mordecai Roshwald.
No Matches in Wikipedia page of " Mordecai Roshwald ."

Sighetu Marmației is Elie Wiesel's last place.
Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/Sighetu_Marmației

Tsutomu Hata's office is Japan.
Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/Tsutomu_Hata

Heike Kamerlingh Onnes' award is Nobel Prize in Literature.
No Matches in Wikipedia page of " Heike Kamerlingh Onnes ."

2012 (film) stars Art Carney.
No Matches in Wikipedia page of " Art Carney ."

Lithuania Governorate is Adam Mickiewicz's nascence place.
No Matches in Wikipedia page of " Lithuania Governorate ."
No Matches in Wikipedia page of " Adam Mickiewicz ."

Boston Celtics is Kevin Garnett's honour.
No Matches in Wikipedia page of " Kevin Garnett ."

Denmark is Stjepan Mesic's role.
No Matches in Wikipedia page of " Denmark ."
Mathched 3 keywords on: Contentious material about live person that is unsourced or poorly source must be remove immediately , especially if potentially libelous or harmful.Find source : ' Stjepan Mesić ' - news · newspaper · book · scholar · JSTOR (March 2016) (Learn how and when to remove this template message) .
Mathched words: { 'live', 'be', 'is' }
Mathched 3 keywords on: Contentious material about live person that is unsourced or poorly source must be remove immediately , especially if potentially libelous or harmful.Find source : ' Stjepan Mesić ' - news · newspaper · book · scholar · JSTOR (February 2016) (Learn how and when to remove this template message) .
Mathched words: { 'live', 'be', 'is' }
Matched 1 : Contentious material about living persons that is unsourced or poorly sourced must be removed immediately, especially if potentially libelous or harmful.Find sources: "Stjepan Mesić" - news · newspapers · books · scholar · JSTOR (March 2016) (Learn how and when to remove this template message).
Matched 2 : Contentious material about living persons that is unsourced or poorly sourced must be removed immediately, especially if potentially libelous or harmful.Find sources: "Stjepan Mesić" - news · newspapers · books · scholar · JSTOR (February 2016) (Learn how and when to remove this template message).

The Urth of the New Sun's author is Cordwainer Smith.
No Matches in Wikipedia page of " Cordwainer Smith ."

Roy Orbison's death place is Hendersonville, Tennessee.
No Matches in Wikipedia page of " Roy Orbison ."
No Matches in Wikipedia page of " Tennessee ."

Martell Webster's team is Charlotte Bobcats.
No Matches in Wikipedia page of " Martell Webster ."

Rome is Alfonso XIII of Spain's nascence place.
No Matches in Wikipedia page of " Rome ."
No Matches in Wikipedia page of " Alfonso XIII ."
No Matches in Wikipedia page of " Spain ."

Los Angeles Clippers is Mike Miller (basketball player)'s team.
No Matches in Wikipedia page of " Mike Miller ."

Paul van Zeeland's office is Belgium.
No Matches in Wikipedia page of " Paul van Zeeland ."
No Matches in Wikipedia page of " Belgium ."

A Hard Day's Night (film) stars Maggie d'Abo.

Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/Maggie_dAbo

Abraham Lincoln's birth place is Hodgenville, Kentucky.
No Matches in Wikipedia page of " Abraham Lincoln ."
No Matches in Wikipedia page of " Kentucky ."

Maria das Neves' office is Israel.
No Matches in Wikipedia page of " Maria das Neves ."
No Matches in Wikipedia page of " Israel ."

Techexcel's foundation place is Lafayette, California.
No Matches in Wikipedia page of " California ."

Superbad (film) stars Nick Jonas.
No Matches in Wikipedia page of " Nick Jonas ."

Oklahoma City Thunder is Amir Johnson's squad.
No Matches in Wikipedia page of " Oklahoma City Thunder ."
No Matches in Wikipedia page of " Amir Johnson ."

David Bowie's death place is New York City.
No Matches in Wikipedia page of " David Bowie ."
No Matches in Wikipedia page of " New York City ."

Nobel Prize in Literature is Max Planck's honour.
No Matches in Wikipedia page of " Max Planck ."

Niles, Ohio is William McKinley's role.
No Matches in Wikipedia page of " Ohio ."
No Matches in Wikipedia page of " William McKinley ."

Desi Arnaz's death place is Del Mar, California.
No Matches in Wikipedia page of " Desi Arnaz ."
No Matches in Wikipedia page of " California ."

Nobel Prize in Literature is William Henry Bragg's honour.
Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/William_Henry_Bragg

Melbourne is Elizabeth Taylor's last place.
No Matches in Wikipedia page of " Melbourne ."
Mathched 4 keywords on: :12-18 Filming wa finally complete in July 1962.:39 The film 's final cost wa \$ 62 million , make it the most expensive film made up to that point .
Mathched words: {'point', 'finally', 'final', 'cost'}
Matched 1 : :12-18 Filming was finally completed in July 1962.:39 The film's final cost was \$62 million, making it the most expensive film made up to that point.

Michael Chang's birth place is Hoboken, New Jersey.
No Matches in Wikipedia page of " Michael Chang ."
No Matches in Wikipedia page of " New Jersey ."

United States is Information Builders' innovation place.
No Matches in Wikipedia page of " United States ."

Godzilla (1998 film) stars Matthew Broderick.

No Matches in Wikipedia page of " Matthew Broderick ."

Cincinnati is William Howard Taft's nascence place.

No Matches in Wikipedia page of " Cincinnati ."

Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/William_Howard_Taft

Hubert Humphrey's death place is Hamburg.

Can not find the target page in Wikipeadia. https://en.wikipedia.org/wiki/Hubert_Humphrey

Hannibal's death place is Gebze.

No Matches in Wikipedia page of " Hannibal ."

No Matches in Wikipedia page of " Gebze ."

Nobel Peace Prize is Henry Dunant's honour.

No Matches in Wikipedia page of " Henry Dunant ."

Bernard Archard is Stephen Dunham's better half.

No Matches in Wikipedia page of " Bernard Archard ."

No Matches in Wikipedia page of " Stephen Dunham ."

Chet Atkins' death place is Nashville, Tennessee.

No Matches in Wikipedia page of " Chet Atkins ."

No Matches in Wikipedia page of " Tennessee ."

Derek Fisher's team is Boston Celtics.

No Matches in Wikipedia page of " Derek Fisher ."

Kwame Brown's team is Dallas Mavericks.

No Matches in Wikipedia page of " Kwame Brown ."

Anna Faris is Tony Parker's better half.

No Matches in Wikipedia page of " Anna Faris ."

No Matches in Wikipedia page of " Tony Parker ."

James Blish's birth place is East Orange, New Jersey.

No Matches in Wikipedia page of " James Blish ."

No Matches in Wikipedia page of " New Jersey ."

Nobel Prize in Physiology or Medicine is Herbert Spencer Gasser's honour.

No Matches in Wikipedia page of " Herbert Spencer Gasser ."

MailChannels' foundation place is Vancouver.

Mathched 4 keywords on: Vancouver 's property crime rate is particularly high , rank among the highest for major North American cities .

Mathched words: {'is', 'rank', 'rate', 'property'}

Matched 1 : Vancouver's property crime rate is particularly high, ranking among the highest for major North American cities.

China is the capital of United Kingdom

No Matches in Wikipedia page of " China ."

No Matches in Wikipedia page of " United Kingdom ."

Nobel Prize in Literature is Theodor Mommsen's honour.

Mathched 5 keywords on: He receive the Nobel Prize in Literature in 1902 for be `` the greatest live master of the art of historical write ,

with special reference to his monumental work , A History of Rome '' , after have been nominate by 18 member of the Prussian Academy of Sciences .

Mathched words: {'live', 'be', 'Prize', 'Nobel', 'Literature'}

Matched 1 : He received the Nobel Prize in Literature in 1902 for being "the greatest living master of the art of historical writing, with special reference to his monumental work, A History of Rome", after having been nominated by 18 members of the Prussian Academy of Sciences.

Ernest Rutherford's birth place is Cambridge.

No Matches in Wikipedia page of " Ernest Rutherford ."

No Matches in Wikipedia page of " Cambridge ."

Output of first version:

Exception Occured :

1: 1.0

2: 1.0

3: 0.0

4: 1.0

5: 1.0

6: 1.0

7: 0.0

8: 1.0

Exception Occured : Barac Oobama

9: 0.0

10: 1.0

11: 0.0

12: 1.0

13: 1.0

14: 0.0

15: 1.0

16: 1.0

/home/mushfika/anaconda3/lib/python3.6/site-packages/wikipedia/wikipedia.py
:389: UserWarning: No parser was explicitly specified, so I'm using the best available HTML parser for this system ("lxml"). This usually isn't a problem, but if you run this code on another system, or in a different virtual environment, it may use a different parser and behave differently.

The code that caused this warning is on line 389 of the file /home/mushfika/anaconda3/lib/python3.6/site-packages/wikipedia/wikipedia.py. To get rid of this warning, pass the additional argument 'features="lxml"' to the BeautifulSoup constructor.

```
lis = BeautifulSoup(html).find_all('li')
```

Disambiguation error: Hard

17: 0.0

18: 1.0

19: 1.0

Exception Occured : Poison Belt

20: 0.0
21: 1.0
22: 0.0
23: 0.0
24: 1.0
25: 1.0
26: 1.0
27: 0.0
28: 1.0
29: 1.0
Exception Occured : Sighetu Marma
30: 1.0
31: 1.0
32: 1.0
33: 1.0
34: 0.0
35: 1.0
36: 0.0
Disambiguation error: Urth
37: 0.0
Disambiguation error: Hendersonville
38: 0.0
39: 0.0
40: 1.0
41: 0.0
Disambiguation error: Paul
Exception Occured : Zeeland
42: 0.0
43: 0.0
44: 1.0
Disambiguation error: Maria
Disambiguation error: Neves
45: 0.0
Disambiguation error: Lafayette
46: 0.0
47: 0.0
48: 0.0
49: 1.0
50: 0.0
51: 1.0
52: 1.0
53: 1.0
54: 0.0
55: 1.0
56: 1.0
57: 0.0
58: 1.0

59: 0.0
60: 1.0
61: 0.0
62: 0.0
63: 1.0
64: 1.0
65: 0.0
66: 0.0
67: 1.0
68: 0.0
69: 1.0
70: 1.0
71: 1.0
72: 1.0
73: 1.0
74: 1.0
75: 1.0
76: 0.0
77: 1.0
78: 1.0
79: 1.0
80: 1.0
Exception Occured : Greenest Cities
81: 1.0
82: 1.0
Disambiguation error: Christian
83: 1.0
84: 1.0
85: 0.0
86: 1.0
87: 1.0
88: 1.0
89: 1.0
90: 1.0
91: 1.0
92: 0.0
93: 1.0
94: 1.0
95: 1.0
96: 1.0
97: 1.0
98: 1.0
99: 1.0
100: 1.0
101: 1.0