SENTIMENT ANALYZING MACHINE THAT CAN UNDERSTAND ENUMERATED LANGUAGES API

DENNIS B. DE LEON

CHRISTIAN NOEL C. MOLINA

PATRICK DALE F. RUGAYAN

ERIC B. SUMALINOG

A Thesis Proposed to the

Faculty of the College of Science

Technological University of the Philippines

Ayala Blvd., Manila

In Partial Fulfillment

Of the Requirements for the Degree

Bachelor of Science in Computer Science

March 2018

Technological University of the Philippines
College of Science
Manila

APPROVAL SHEET

This thesis hereto entitled

"Sentiment Analyzing Machine that can Understand Enumerated Languages API"

Prepared and submitted by **Dennis B. De Leon, Christian Noel C. Molina, Patrick Dale F. Rugayan** and **Eric B. Sumalinog** in partial fulfillment of the requirement for the degree Bachelor of Science in Computer Science has been examined and is recommend for acceptance and approval for **ORAL EXAMINATION**.

FERNANDO L. RENEGADO
Adviser

Approved by the Committee on oral examination with a grade of **PASSED** on February 19, 2018.

MARIA CARMELA F. FRANCISCO
Chair Person

JAN EILBERT L. LEE
Member

AMADOR B. PERAGRINO
Member

Accepted in partial fulfillment of requirements for the degree **Bachelor of Science in Computer Science.**

Date:

Prof. FIDELA Q. ARAÑES
Dean

Acknowledgment

In light of the success of this study, we the authors would like to express our gratitude to the people who have been with us in our toughest time and have their support and encouragement throughout the entire time until the accomplishment of this study.

First and foremost to God. For the everyday blessings and protection He has given us. Thank You for giving us the strength and knowledge to finish this study.

To our adviser, Prof. FERNANDO L. RENEGADO thank you for sharing your knowledge to us. Thank you for assisting and supporting us until we finish our study. We would’nt be able to finish this study without your guidance. Thank you for your patience and understanding.

To our colleagues who have been there through the thick and thin of our life, who laughed and cried with us. Thank you for all your support to us.

To our panel Prof. MARIA CARMELA FRANCISCO, Prof. AMADOR PEREGRINO, Prof. JAN EILBERT LEE and Prof. FERNANDO RENEGADO for the wisdom and advice you shared to us.

And lastly to our family, for the love, support, understanding and patience you have given us through our journey in college and for being there every moment of our life.

D. De Leon

C.N. Molina

P.D. Rugayan

E. Sumalinog

Abstract

The SAMUEL API was developed to analyze millions of different sentiments. It is a modular API that can be used easily by any developers on their projects. The API was developed using python as the core and JSON as the data interchange. Its objectives are; (1) To be a machine that can understand English and Filipino language. (2) To classify the given sentiments in two polarities, positive or negative polarity. (3) To summarize the given sentiments, opinions or comments on a particular topic, discussion, or argument and create a concise and compact conclusion out of it. (4) To have a modular design of API that can be used and customized easily by any developers. (5) To have a comprehensive and easy to use Linkifier for non-developers to parse and analyze a supported website with just a link. It was evaluated by 30 respondent consist of IT/CS/IS students from the Technological University of the Philippines (TUP) using the ISO 9126 evaluation instrument to assess the acceptability of the system. The weighted frequency rating was interpreted as Highly Acceptable or in overall 540 frequency rating score; seventy percent (70%) or three hundred eighty (380) of evaluation responses was favored to four (4) points which has a descriptive rating equivalent of "Highly Acceptable", the other twenty percent (20%) or one hundred seven (107) of the response was favored to three (3) points which has a descriptive rating equivalent of "Very Acceptable", as for the other nine percent (9%) or fifty-one (51) response was favored to two (2) which has a descriptive rating equivalent of "Acceptable", and for the remaining one percent (1%) or two (2) of the response was favored to one (1) which has a descriptive rating equivalent of "Not Acceptable". It indicated that the project will be really helpful in analyzing millions of different sentiments.

Table of Contents

Title Page 1

Approval Sheets 2

Acknowledgement 3

Abstract 4

Table of Contents 5

List of Tables 7

List of Figures 8

List of Appendices 9

CHAPTER 1 THE PROBLEM AND ITS SETTING 10

Introduction 10

Background of the Study 11

Objectives of the Study 12

Scope and Limitations of the Study 13

CHAPTER 2 CONCEPTUAL FRAMEWORK 15

Review of Related Literature 15

Related Studies 35

Conceptual Model of the Study 39

Operational Definition of Terms 42

CHAPTER 3 METHODOLOGY 44

	Project Design	44
	Project Development	55
	Operation and Testing Procedure	56
	Evaluation Procedure	57
	Evaluation Instrument	57
CHAPTER 4	**RESULT AND DISCUSSION**	**59**
	Project Description	59
	Project Structure	60
	Project Capabilities and Limitations	67
	Test Results	68
	Project Evaluation	69
CHAPTER 5	**SUMMARY OF FINDINGS, CONCLUSIONS, AND RECOMMENDATIONS**	**72**
	Summary	72
	Conclusions	73
	Recommendations	74
REFERENCES		**76**
CURICULUM VITAE		**86**

List of Tables

Table		Page
1	Modules and Testing Procedures	56
2	Numerical Rating and its Equivalent	58
3	Test Results for the Functionality of the Project	68

List of Figures

Figure		Page
1	Intersection of Text Mining	22
2	The Conceptual Model of the Study	39
3	SAMUEL Project Design	44
4	Bayesian Inference	48
5	Landing Page	60
6	About SAMUEL API page	61
7	Developer's Registration Page	61
8	API Key Creation Page	62
9	Documentation Page	62
10	API Keys Page	63
11	Developer's Dashoard Page	63
12	SAMUEL API Linkifier Landing Page	64
13	Linkifier Page	64
14	Parsing Status	65
15	Parsing Complete	65
16	Linkifier Result Page	66
17	SAMUEL API Core Server Log	66

List of Appendices

Appendix		Page
A	Evalutaion Instrument	79
B	Answered Evalution Sheet	81
C	Summary of Respondents of Evaluation	82
D	Gantt Chart	83
E	User’s Manual	84

CHAPTER 1

THE PROBLEM AND ITS SETTING

Introduction

Our day-to-day actions had been affected by other's input which has a powerful impact in our behavior. These judgements and choices we had does not only range from picking a hero to fight in an all-out war in a certain game, how to project in a subject like modelling, or what to think of your new-found friend. The most distinct and deviated decision you could ever make in your life is to do not let judgements or criticism affect you negatively. During the pre-modern era, where the internet is seldom available for people to encounter, particular humanity would look for thoughts and inputs on some ranges on particular things from different sources: be it family, comrades, or customer reviews which are printed on paperback then. Unlike in this era, information is so accessible that you can instantly collect millions of feedback and advice about your inquiry.

New possibilities were opened for information exchange. Exceptional opportunities for citizens was also created to publicly raise their opinions with the growth of social media and the Internet, but it has severe bottlenecks when it comes to creating an analysis of these opinions. Even the necessity to gain a real-time understanding of citizens concerns has grown rapidly. The viral nature of social media which is swift and scattered, some issues get distributed as fast as possible and unexpectedly became significant through words of opinions disclosed online which in turn has been known as the sentiments of the users. The decision makers and people do not yet realize how to make use of this mass communication and how they can interact cleverly with thousands of users with the help of sentiment analysis.

People tend to seek review sites, e-commerce sites, online opinion sites and social media to get feedbacks on how a particular product or service is perceived in the market. Similarly, organizations use surveys, opinion polls, and social media as an instrument to gather feedbacks on their products and services. Sentiment analysis or opinion mining is the computational study of opinions, sentiments, and emotions expressed in the text. The use of sentiment analysis have been widely taken advantage of since the information it produces can result in the monetization of products and services. For instance, by retrieving a consumer’s evaluation on a marketing campaign, an organization can measure how successful the campaign was and learn how to adjust things for greater success in the future. Product feedback is also helpful in establishing better products, which can have a direct impact on profit, as well as in comparing competitor’s offerings.

Background of the Study

The World Wide Web is a huge faucet of documents, many of which express opinion, provide input, advice, feedback, review, comments, critiques, and blogs. These important documents contain information which can be an asset to people that has difficulty in making their decisions. For example, multiple product reviews can help an entrepreneur to promote his Product and know how to improve it; a politician having a hard time with his opponent can have a political strategy by securing the lifeline of advices and inputs from strategists and political connoisseurs; event critiques can lend a hand by saying constructive criticism to the event organizers and offer an advice to further improve their events in the future. However, these inputs and comments ranges to billions of advices to read, and it is impossible for people with ample time to read and reflect at each one of them. Thus, providing automatic analyzation and conclusion to the opinions would be such a great leap for it will save countless amount of time and it is much more effective than manually contemplating on every single comment online. The task of developing the solution to the said problem is called Sentiment Analysis or in other words, Opinion Mining.

In this comprehensive project, we attempt to analyze how the sample sentiment scales can identify the connective, conjunctions, and phrases used in its text and cross reference it to a dictionary that will provide its meaning and placement on the scale. As a result, the key phrase which expresses the sentiment orientation of the author can be identified. The method will be a combination of classical analysis methods and machine learning based or if not applicable, clustering based, to achieve a higher accuracy.

Objectives of the Study

The general objectives object of the study is to develop a Sentiment Analyzing API that can help the developers to easily analyze different sentiments on their projects.

Specifically it aims to:

1. Design the project with the following features:

  a. A machine that can understand English and Filipino language;
  b. Can classify the given sentiments in two polarities: positive or negative polarity;
  c. Can summarize the given sentiments, opinions or comments on a particular topic, discussion, or argument and create a concise and compact conclusion out of it;
  d. A Modular design of API that can be used and customized easily by any developers; and
  e. A comprehensive and easy to use Linkifier for non-developers to parse and analyze a supported website with just a link;

2. Create and implement the project using the following:
   a. Python will as Core Library of the API.
   b. JSON will be the Data Interchange of the API.
   c. For the Implementation, a key will be provided for the developers to have access to the API; and
   d. SAMUEL API website for the Linkifier and Creation of API keys.
3. Test and improve the project based on portability and utility.
4. Evaluate the acceptability of the project using ISO 9126.

Scope and Limitations of the Study

This project is all about analyzing sentiments, comments, reviews, opinions and create a conclusion out of it and determine whether the given sentiments are positive or negative. The main focus of the project is to create a modular API (Application Program Interface) for the developers.

The proposed project is limited only in understanding English and Filipino language only, it cannot also understand man-made languages such as Jejemon, Beki Lingo, etc. The Linkifier only supports Reddit, Youtube and Webistes with common forum layout.

CHAPTER 2

CONCEPTUAL FRAMEWORK

Review of Related Literature

API

API or Application Program Interface (sometimes also known as software libraries) is a set of routines, protocols, and tools for building an efficient application software. It is a set of methods that help software components to communicate with each other. API is a great help for developers by providing building blocks that can be put together easily. Just like the GUI or Graphical User Interface, it makes it easier for the developers to program by using certain technologies in building applications.

A good API is measured by its usability, it is important for the API to provide only the needs of the user. This principle is called Information Hiding, making the API modular through hiding the implementation details of the modules so that the users don't need to study the deeper part of the modules. Joshua Bloch, Kin Lane, and Michi Henning are three of the several authors that created recommendations on how to design a good and efficient API.

In developing an API, a descriptive documentation is really important. It must provide what the API can offer, how to use it and everything the client needs to know how to use the API effectively.

There are API’s available for operating systems, web-based applications, software applications, database system and computer hardware. The most popular API available now is Google Maps API, YouTube API, Amazon API and many others.

Machine Learning

In computer science, machine learning is one of its subfields, it is a method of analyzing big data. If humans can learn from past experiences, the machine can learn from past data and that is the idea of machine learning. According to Arthur Samuel in 1959, machine learning is a field of study that allows the computer to create insights from past data without being explicitly programmed.

One of the biggest problem today is making models out of massive data to make data driven predictions or decisions expressed as outputs, and that’s where machine learning comes in because it automates analytical model building. Thomas H. Davenport says that “...in this growing volumes of data we need fast moving modeling steams to keep up”, he also said that “Humans can typically create one or two good models a week; machine learning can create thousands of them a week”.

There are many approaches or algorithms in creating a machine learning programs like decision tree learning, artificial neural networks, deep learning, clustering and many others. But there’s a challenge in creating an effective machine learning programs, inaccurate pattern recognition and sometimes not enough training data available that’s why it often fails to deliver.

Right now there are machine learning applications that are widely publicized, here are some of them: The heavily hyped, self-driving car that was created by Google, Online recommendation offers like those from Amazon and Netflix, Knowing the customer’s sentiments, and Fraud Detection.

Natural Language Processing

Natural language processing (NLP) utilizes computational techniques for the purpose of learning, understanding, and producing human language content. It can be defined as the automatic or semi-automatic processing of human languages. It is a wide topic area in a sense that it covers any kind of computer manipulation of natural language. It could be as simple as counting word frequencies to compare different writing styles to and as complex as being able to understand complete human utterances and giving helpful responses to them.

The term NLP is sometimes used rather more narrowly, often excluding information retrieval and sometimes even excluding machine translation. Most of the time it is contrasted with computational linguistics, with NLP being thought of as more applied. Nowadays, alternative terms are often preferred, like Language Technology or Language Engineering.

During the early days, computational approaches to language research concentrated on automating the analysis of the linguistic structure of languages and developing basic technologies such as machine translation, speech recognition, and speech synthesis. In today’s modern society, researchers now polish and use such tools in real-world applications, creating spoken dialogue systems and speech-to-speech translation engines, mining social media for information about health or finance, and identifying sentiment and emotion toward products and services. Natural language processing undeniably aims to make the computers perform useful tasks involving natural human languages, making its technologies become increasing. Phones and handheld computers can now support predictive text and handwriting recognition, web search engines can now give access to information locked up in unstructured text, and machine translation now allows us to retrieve texts written in Chinese and read them in Spanish. By providing more human-machine interfaces, and more sophisticated access to stored information, language processing has come to play a central role in the multilingual information society.

Systems that implement natural language processing take sentences as its input and output’s structure represents meaning of the sentences. The nature of the output highly depends on the task that is currently at hand. For instance, a natural language understanding system that serves as an interface to the specific database might accept questions in English which relate to the kind of data held by the database. For this case, the output of the system perhaps can be expressed in terms of structured SQL queries that can be straightly submitted to the database.

Unfortunately, there are two (2) major problems in particular that make NLP really difficult to handle which eventually cause different techniques to be used than those associated with the construction of compilers for processing artificial languages.

The first problem, the existing level of ambiguity in natural languages, while the second is, the existence of complex semantic information even in simple sentences. Generally, language processors confront a large number of words which mostly have much alternative use and large grammars that allow various phrase types to be composed of the same string of words. They are indeed made more complicated since irregularity of languages and the different kinds of ambiguity exist within sentences.

In the real natural language processing systems, the stages of NLP namely: namely, morphological processing, syntax analysis, semantic analysis, and pragmatic analysis rarely occur as separated, sequential processes. In the overview that follows it is assumed that syntactic analysis and semantic analysis will be dealt with by the same mechanism - the parser.

The first stage in the process after accepting the input is the morphological processing. It is the stage where the system aims to break strings of language input into sets of tokens corresponding to discrete words, sub words, and punctuation forms. Its primary concern is to recognize how base words have been modified to form other words with similar meanings but frequently with different syntactic categories. The output of this stage is a string of tokens, which can then be used for lexicon lookup that may contain tense, number, gender and proximity information, and in some cases, may also contain additional syntactic information for the parser.

The next stage of the process is syntax analysis. It is known that a language processor must carry out a number of distinct functions which is basically based on syntax and semantic analysis. The purpose of syntax analysis is to check if a sentence is correct and to break it up into a structure of syntactic relationships between different words. A syntactic analyzer or parser does this by using a dictionary of word definitions which is called as lexicon, and a set of syntax rules known as the grammar. In a simple lexicon, it only contains the syntactic category of each word, and in a simple grammar, it only describes rules that indicates how syntactic categories can be combined to form phrases of different types. On the other hand, semantic analysis is done by expanding the lexicon to include semantic definitions for each word there is, and by extending the grammar which specifies how the semantics of any phrase is formed through the semantics of its component parts. After applying all of the relevant rules, the semantics are organized into larger sub-phrases until a semantic expression describing the whole sentence is produced.

After semantic analysis comes the final stage of processing which is pragmatics analysis. Unfortunately, there is an ambiguity in distinguishing the context of semantics and pragmatics. It is said that semantic analysis associates meaning with isolated sentences, while pragmatic analysis interprets the results of semantic analysis from the perspective of a specific context. In some cases, pragmatic analysis simply fits actual objects/events which exists in a given context with object references obtained during semantic analysis. In other cases, pragmatic analysis can disambiguate sentences which cannot be fully disambiguated during the syntax and semantic analysis phases.

At the end of the day, determining the meaning of an utterance is really one-half of the story of natural language processing: in many situations, a response then needs to be generated, either in natural language alone or in combination with other modalities.

For many of today's applications, what is required here is rather trivial and can be handled by means of canned responses. Increasingly, however, we are seeing natural language generation techniques applied in the context of more sophisticated back-end systems, where the need to be able to custom-create fluent multi-sentential texts on demand becomes a priority.

Text Analytics

Text analytics also called as text mining, is a broad umbrella term used for describing a range of technologies that uncovers business value in unstructured textual data and transforms it into a structured and numerical data format. Subsequently, it applies statistics, linguistics, machine learning, data analysis, visualization techniques and knowledge on how to combine techniques for handling texts, ranging from individual words to entire document databases, to identify and extract pertinent information. While the term text analytics and text mining are often used interchangeably, one can rather conceptualize that text mining is a subset of text analytics that focuses on applying data mining techniques in the domain of textual information. Text mining only considers the structural relationship between words which does not include phonetics, pragmatics, discourse, and such.

Text analytics processes and analyzes texts to extract meanings, patterns, and structure hidden in the given data. The development of the field came from the problem with the beginning of the information age as a massive amount of textual data starts to pile up. People and businesses needed to comprehend those, so they need to find a way to extract information without manually processing it as it is not practical for any individual or organization to do so. Over time, text analysis became a loosely integrated framework by borrowing different techniques from data mining, machine learning, and natural language processing (NLP), information retrieval, and knowledge management.

Text analytics applications are widely used in the business environment and are implemented into various industries from discovering trends and predicting outcomes, understanding market dynamics, preventing crimes and detecting frauds using social media data, improving patient outcomes and providing better care in hospitals, to mining biomedical literature and discovering new drugs. It incorporates numerous tools and techniques that are used to derive insights from unstructured data, such as concept extraction, summarization, topic tracking, deep learning, and sentiment analysis.

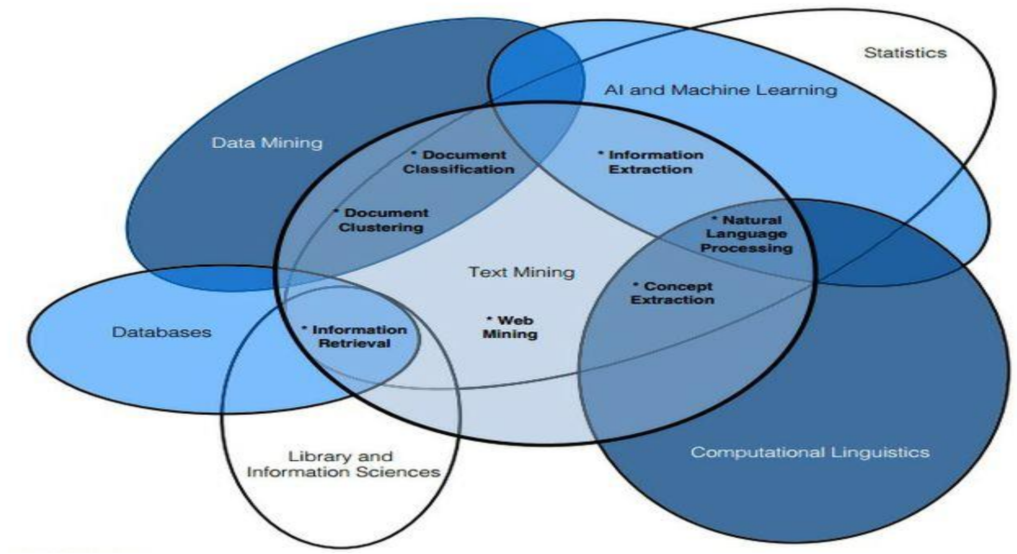


Figure 1. Intersection of Text Mining

The steps and tasks of a typical text analytics application begins with the collection of documents, in which a text mining tool retrieves a particular document then pre-process it by checking formats and character sets. Afterward, it undergoes a text analysis phase wherein techniques are repeatedly done until the information is extracted. The fundamental strategy in all of the components is to find a pattern that needs to match a particular rule, then applies that rule to eventually annotate the text. Each component performs a specific process on the text such as sentence segmentation which divides the text into sentences, tokenization that identifies individual words in the text, part-of-speech tagging which determines relationship among words, shallow syntactic parsing that divides text by noun phrase, verb phrase and alike, named entity recognition in which entities in the text such as people are distinguished. And dependency analysis that identifies dependency of phrases and words to each other like subordinate clauses and pronominal anaphora.

Sentiment Analysis

Opinions are one of the most central standpoint to all human activities. It is the underlying key to influence other behaviors and with that in mind, different businesses' and organizations' lean on their customer's opinions about their products and services. The mere fact made the field of sentiment analysis flourished throughout with the explosive growth of social media, from reviews, forum discussions, blogs, micro-blogs, Twitter, comments, to postings on all social network sites on the web, as individuals and public consumers tend out to highly express their sentiments and opinions using these tools.

Sentiment Analysis, also known as Opinion Mining, is a field of study that thoroughly analyzes and evaluates people's sentiments, evaluations, reactions, appraisals, attitudes, and emotions towards a particular or involved topic concerning any entity of products, services, organizations, individuals, issues, events, topics, and their attributes. Despite its many names namely: sentiment analysis, opinion mining, opinion extraction, sentiment mining, subjectivity analysis, effect analysis, emotion analysis, and review mining, which runs slightly different tasks and goals, they all now fall under the same concept and field.

Sentiment analysis can occur and are investigated under three (3) main levels. The first level is the Document Level Analysis in which the whole sentiment is classified whether it is positive or negative and assumes that it is only about a single entity. The second level is known as the Sentence Level Analysis which usually involves two steps; namely, the subjectivity classification and sentiment classification, in which the analysis aims to determine whether each sentence is positive, negative, or neutral.

In subjectivity classification, it distinguishes whether the sentence itself is objective that represents factual information or subjective that expresses personal feelings, views, emotions, or beliefs and can be achieved through different methods such as the Naïve Bayesian classification. This intermediate step will help to filter out the sentences with those that have no opinions and eventually determine to an extent if sentiments about entities and their aspects are positive or negative. Throughout the two

(2) levels, the analysis does not exactly know what does the person actually likes or dislikes, and that's where the third level of analysis comes, the Aspect Level Analysis. This third level of analysis focuses on the opinion itself instead of just looking at language constructs, where the idea is that an opinion consists of a sentiment and a target of that opinion, discovering their sentiments on entities and/or their aspects.

Unsurprisingly, sentiment words or also known as opinion words, are the most important aspect in identifying a sentiment's polarity, as these words are commonly used to express positive or negative sentiments. Sentiment words are individual words such as fascinating, amazing, and incredible, which are positive sentiment words; disgusting, terrible, and worst, which are negative sentiment words. Also, it includes phrases and idioms such as in good spirit which basically means the person is indeed happy. The list of such words and phrases is called sentiment lexicon or opinion lexicon. Although sentiment lexicon is significant for performing sentiment analysis, it is way too far from enough to solve the problems that the field is currently facing. Some of the difficult problems are that a positive or negative sentiment word may have opposite orientations in different application domains, and other sentences that contains sentiment words may not express any sentiment at all. Additionally, the sentence that is containing sarcasm with or without sentiment words are hard to deal with, and many sentences without sentiment words can also imply opinions.

Subsequently, sentiment analysis is actually Natural Language Processing (NLP) problem, it almost touches every aspect of the field including coreference resolution, negation handling, and word sense disambiguation which makes it more difficult to handle. Yet, it is useful to remember that sentiment analysis is a highly restricted NLP problem because the system does not need to fully understand the semantics of each sentence or document. It needs to understand some of its aspects such as positive or negative sentiments and their target entities or topics. In such sense, it becomes a great platform for the field of NLP as touching the field of sentiment analysis almost also touches the forefront fields of NLP itself.

A sentiment is indeed really formatted in such unstructured ways, and this is where text analysis comes into play, for the data to be converted into a more structured format that facilitates analysis of data and getting deeper insights into the sentiments. There are existing various techniques to achieve such requirements such as expressing the opinion as a quintuple (ej , ajk , soijkl , hi , tl) where ej is the entity target, ajk is an aspect/feature of the entity ej, soijkl is the sentiment value of the opinion from the opinion holder; soijkl is +ve, -ve, or neutral, or more granular ratings, hi is an opinion holder, and tl is the time when the opinion is expressed. Once the data becomes structured, it will be much easier to analyze and perform sentiment analysis.

As stated by Katrekar, sentiment analysis is an evolving field with a variety of useful applications. Although sentiment analysis tasks are challenging due to their natural language processing origins, much progress have been made over the last few years due to the high demand for it. Not only do companies want to know how their products and services are perceived by consumers (and compare to competitors), but consumers want to know the opinions of others before making buying decisions. The growing need for product insights – and the technical challenges the field is currently facing –will keep sentiment analysis and opinion mining relevant for the foreseeable future. Next-generation opinion mining systems need a deeper bond between complete knowledge bases with reasoning methods inspired by human thought and psychology. This will lead to a better understanding of natural language opinions and will more efficiently bridge the gap between unstructured information in the form of human thoughts and structured data that can be analyzed and processed by a machine. The result: intelligent opinion mining systems will be capable of handling semantic knowledge, making analogies, continuous learning, and detecting emotions — leading to highly efficient sentiment analysis.

Computational Linguistics

The idea of giving computers the ability to process human language is as old as the idea of computers themselves, and with that idea, came the science of computational linguistics. Computational Linguistics (CL) is basically the discipline that concerns dealing with various models, methods, technologies, systems and applications which automate the processing of languages, both spoken and written. Therefore it includes fields such as Speech Processing (SP) that processes spoken words, and Natural Language Processing (NLP) that processes texts. However, the term computational linguistics is often used interchangeably and related to natural language processing. If someone needs to be more specific between the two terms, one may consider that computational linguistics is the scientific and theoretical side while natural language processing is one of its engineering domain.

Computational linguistics has developed hastily from a relatively obscure auxiliary of both AI and formal linguistics into a thriving scientific discipline, and later on, has been an intermediate position between Computer Science and Artificial Intelligence, Linguistics and Cognitive Science, and Engineering. Computer Science itself shares its roots with Computational Linguistics as parsing, which is an essential in designing compilers for programming languages, is also one of the most significant components of any natural language processing engine, and both are the results of the Chomskian theory of formal languages.

Traditional linguistics also shares some properties with CL, yet CL has indeed some new characteristics in comparison. Foremost, computational linguistics obviously saves linguistic problems with computational techniques like statistical analysis of linguistic data, cross-language feature extraction, and the knowledge-based perception and production of linguistic materials. Secondly, multiple components of computational linguistics can assist linguistic research from different angles. Lastly, computational linguistics is certainly a multi-disciplinary subject, similar to traditional linguistics.

Computational linguistics produced technologies which are acquired in various areas from different communities and became an important area of industrial development. Specifically, the focus of research in CL and NLP has changed over the past three decades from the study of small prototypes and theoretical models to powerful learning and processing systems applied to a large collection of texts. Computer scientists started to lean to the field of CL for dealing automatic translations, telecommunication engineers for working out with speech processing, and linguists for handling the structure and evolution of languages. Truly, it is a field that eventually became a multidisciplinary area of research with an interdisciplinary between the far areas of Humanities and hard sciences such as Computer Science.

Soft Computing

Soft Computing is a solution or method for developing a system, program or machine in which the objective is to make its output and computation more intelligent and wiser by aiming a similarity of how a human being think and decide such as solving complicated task and providing output by applying consciousness and wisdom that the computer cannot easily obtain. Tolerating the lack of accuracy, lack of explicitness and approximation dependency are the principles of this method.

Soft computing is a collection of methodologies that implies learning from experience, applying theoretical computation, simulating biological processes, complicated representations and so on. Being the intelligent and challenging method had its drawbacks such as inaccuracy, imprecision and relativity in the domain experiences.

Modern application such as data mining techniques, image processing, customer profiling, facial recognition, fraud detection and the plagiarism checker are developed and implemented in the soft computing solution.

Python

Python that was created by Guido van Rossum is a high-level programming language that is practical to use in the development of prototype programs purposeful to programming tasks that doesn't affect the maintainability process which developers time and workload is usually spend. Its syntax can be interpreted in several operating system which requires dynamically typing and a bunch of classes and data types that supports standard technicalities and task on an internal system of computer. It also provides standard library and dynamic modules that uses to compile other programming languages, extended interface for various application and capability to run in different operating systems.

Python covers modules, classes and inheritance which defines it as an object-oriented programming language. It also features advanced concepts which utilize its dynamic components and libraries. Various standard programming concepts that defines the language are the variety of data types such as complex and infinitive length integers, dictionaries, lists, tuples, error handling, dynamically typed data types, generators, memory management, unique libraries and so on.

Python can be freely modified which lead different developers, hobbyist and programmers from different community to contribute and feature contemporary changes that is modern and significant in the current era. It provides free software that doesn't consume monetary to download and to include in various applications.

Flask

Flask is a python micro web framework that is based on Werkzeug toolkit and Jinja2 template engine. It does not require particular tools or libraries, no database abstraction layer or pre-existing components that are available on any other libraries. But, Flask supports extensions that can add application features as if they were implemented in Flask itself, that's what it makes it special. Flask has so many features, but one distinctive feature of Flask is the easy and efficient creating of RESTful API as the bridge of data to the core of the application or system

Pinterest and LinkedIn are some web applications that are using Flask as their framework. The latest release as of May 2017 is 0.12.2.

JSON (JavaScript Object Notation)

JSON is a lightweight data-interchange language, a text format that are easy for humans to read and write, also easy for machines to parse and generate. It can be used nowadays for many programming languages like C, C++, C#, Java, JavaScript, Perl, Python, and many others. Because of this is an ideal data-interchange language.

JSON is built on two structure. The first one is called collection of name or value pairs, in other languages it is called as an object, record, struct, dictionary, hash table, keyed list or associative array. The other structure is called an ordered list of values, in other languages it is called as array, vector, list or sequence.

Google Translate

A free machine translation tool developed by Google that translates text from and to different languages. It is usually integrated in browser extensions and software applications using their website interface, mobile apps and API. The multilingual tool can support over 100 languages as of May 2017.

The tool is made possible by gathering linguistic data or adopting a statistical machine translation service that looks for patterns and recognizing millions of documents to navigate for its best result. As of November 2016, the technology company announced that it would use an engine called Google Neural Machine Translation which can translate whole sentences rather than every word so that it can be differentiate with the humans speaking and proper grammar acknowledgement.

The machine translation's accuracy is highly being criticized from people who threatens their language professionals' career, but most people accepts the uncertainty of this tool considering that it is constantly improving.

Unsupervised Lexicon-Based Classification

It is a technique for Sentiment Analysis for comparing lexicon data from different sentiments and classification such as positive, negative and neutral by assigning labels and weights to the recorded list of words. All data that will be included in the process of this method must acquire the labels to derive a probable justification, analysis of accuracy and expected to be equally predictive. It includes calculating orientation of words and phrases of a document and building a classifier from labeled instances of data such as text and sentences.

Its contrary method of classifying sentiments is called supervised classifier. The difference is that it leverages the co-occurrences of words and data in a statistical manner across the document by using training data. The task is describing the labels of every lexicon in a machine-learning approach while the unsupervised method uses a dictionary of words that is annotated with word's polarity and strength. The method adapted a Greedy algorithm which possesses a threshold to determine the exploitation and computational output of a traditional problem-solving heuristic.

Common lexical resources such as MPQA, SentiWordNet, WordNet-Affect, SenticNet is regarded in integrating this technique.

Text Normalization

A process of changing text that is different from before is called Text Normalization. It transforms set of strings into a canonical form that allows separation of concerns and processes that comes with the application, system and operation that integrates with it. It is most commonly suitable in advance concepts of language manipulations such as Natural Language Processing tasks like Machine Translation, Opinion Mining, Sentiment Analysis and Information Retrieval.

This pre-processing technique must conform to the system or application it specifies to work as long as it complies with the Unicode standard and natural language.

It has only few requirements to be implemented. One must be aware of the type that it is being transformed and processed for and must be dependent on the procedure it is being produced for.

Normalization is being presented in converting different type of characters with diacritical marks, forming to upper and lower cases, decomposing ligatures from different languages such as Mandarin, Nihongo, Arabic, and so on.

Regular Expressions

A regular expression also called regex, regexp, rational expression or RE is a sequence of characters that consist of alphanumeric and special characters as well as simply metacharacters used to define for searching combination of strings, pattern matching and text replacements. This concept began when Stephen Cole Kleene, a notable American mathematician use regular language also called regular sets to define most of his mathematical notation and symbols. Its first common use is with the UNIX text-processing utilities, being in the POSIX standard and in Perl syntax.

Standard usage of the concept is for search engines, splitting and replacing set of strings in word processors and editors, filtering data in documents, checking validity of email addresses etc. It has a relative version which is called wildcard expression that is frequently used in file management. Most languages are integrated with this method such as PHP, JavaScript, Perl, C#, Java, Ruby and Python that works as built-in or as a library.

Programs like grep, awk, ed and sed and certain utility such as lexical analysis are heavily dependent in this concept.

Related Studies

Sentiment Analysis of Twitter posts about news

The study has made some improvements of the previous studies existing. Improvements in the machine learning technique that they used is the great help for the success of this study because it gives a better result than the other existing techniques.

The study has shown that the tweets about news can be automatically collected and successfully analyzed for their sentiment. Multinomial Naïve Bayes classifier using unigram + bigram presence is what they used to achieve higher accuracy. They've got up to 90.79% accuracy higher than the existing sentiment analysis. This is an impressive result and can be applied for practical applications dealing with sentiment analysis of Twitter posts about news in particular and Twitter posts in general.

Multilingual Sentiment Analysis on Social Media

This study is unlike the usual sentiment analysis that only focuses on one language, typically the English language because it is the international language. This study presented a multi lingual solution of sentiment analysis by taking both English and Dutch into account.

They come up with their own solution to solve the problem. For the part of speech tagging they used an existing approach, but for the language identification, they proposed a new graph based approach they called LIGA. For polarity detection, they also used the idea of the previous works to create their own approach called Rule Based Emission Model (RBEM) algorithm, but for more accuracy, they also create a technique that uses eight heuristic rules to solve the polarity of the sentiments. Both LIGA and RBEM is a solid foundation that is easily extended.

Automatic Analysis of Document Sentiment

This study focuses on the subjectivity detection and polarity classification, they showed that subjectivity detection can compress reviews into much shorter extracts that still retain polarity information at a level comparable to that of the full review. They prove that the subjectivity extracts are more effective input than the originating document, because they are not only shorter but also cleaner representation of the intended polarity.

They have also shown that implementing the minimum cut framework results better in the development of efficient and effective sentiment analysis algorithm. Utilizing this framework can lead to ideal improvement in polarity classification accuracy of the sentiments.

Sentiment Analysis within and across Social Media Streams

They conclude that even though the general proportion of positive to negative is similar across streams, there are marked differences between the two polarities. They indicate that stream-specific topical tendencies must be taken into account when mining sentiment.

Their stream adaptation shows the importance of each stream as a source of training data. They proved that the classifiers that are built using reviews are most generalizable to other streams, followed by Twitter, with Twitter-based model that performs as the native classifier, 8 out of 10 for blogs and 5 out of 10 for reviews.

They also shows that combining training data from different streams is a big help to boost the performance and the efficiency, and combining data from several topics that may even produce classifiers that outperforms other approaches.

Their study of the relative usefulness of social media streams as sources of training data allows more informed design of sentiment analysis tools wherein resources are spent on collecting and labeling data best suited for the task.

Sentiment Analysis using Deep Convolutional Neural Networks with Distant Supervision

In this study they described a CNN system for automatic sentiment prediction and presented an in depth analysis of said system. They investigated deeper network architectures. Furthermore they showed the importance of high quality word-embedding as well as the impact of the distant supervised phase.

They empirically demonstrated that deeper network architectures need a significant amount of training data in the distant-supervised phase. They discussed the importance of the distant-supervised phase with respect to the supervised phase. They presented an in-depth analysis of the mistakes the classifier makes hinting at how to further improve the system. They showed that their CNN outperforms the system based on hand-crafted features by 3.39 points. This indicates that the trend in sentiment analysis shifts towards the use of deep learning methods.

Conceptual Model of the Study

The figure below shows the conceptual model of the study that will serve as a guide in developing the SAMUEL API. This conceptual method used the Input – Process – Output to group and components of the study.

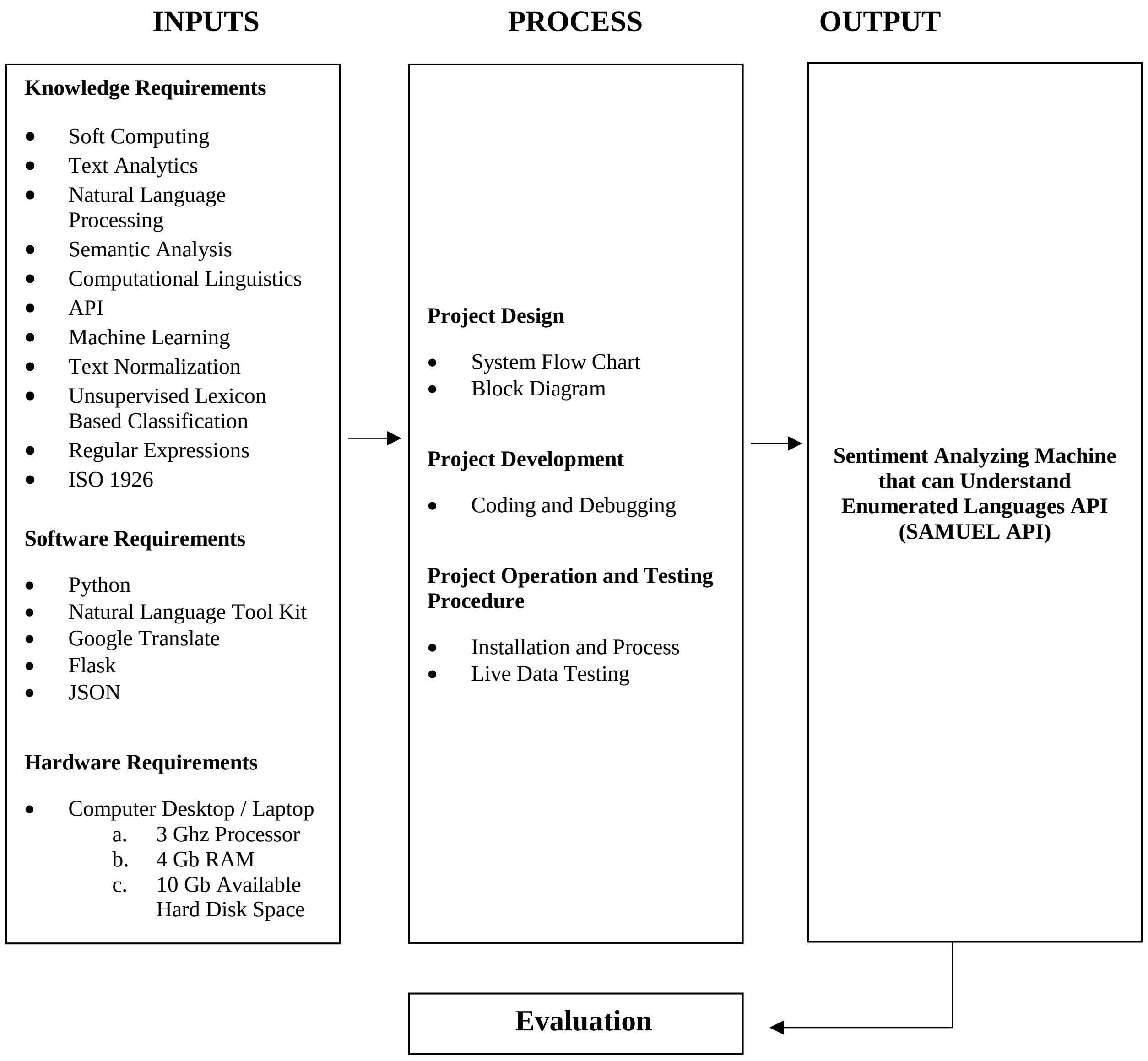


Figure 2. The Conceptual Model of the Study

The conceptual model, as illustrated in Figure 2 shows the different stages of the processes involved in order to achieve the objectives of the study.

Input

The input phase is categorized in three major components namely: knowledge requirements, software requirements and hardware requirements.

The knowledge requirements consist of the following: Soft Computing, Text Analytics, Natural Language Processing, Semantic Analysis, Computational Linguistics, API, Machine Learning, Text Normalization, Unsupervised Lexicon based Classification, Regular Expressions, all this knowledge are needed to come up with an accurate and reliable sentiment analyzer.

The software requirements of the application supports the following language, platform and resources such as: Python, Natural Language Tool Kit, Google Translate, Flask, and JSON. These tools are essential for continuous development and maintenance of the project.

The hardware requirements is Computer Desktop or Laptop.

Process

The process section in the conceptual model of the study is referred to as the software development phase which consist of three parts namely; first, the Project Design which includes the System Flowchart and Block Diagram. Second, Project Development which includes the coding and debugging. Lastly, the Project Operation and Testing Procedure which includes Installation and Process, and Live Data Testing.

Output

Following all the essential and recommended requirements the Sentiment Analyzing Machine that can Understand Enumerated Languages API was developed.

Evaluation

The evaluation phase which determines the frequency of the respondents regarding the proposed project. The concept of the whole study was evaluated in order to know the acceptability of the project. The evaluation was based on the following criteria, namely: Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability.

Operational Definition of Terms

Ambiguity refers to the quality of being open to more than one interpretation.

Backpropagation refers to the process used in neural networks that calculates the error of each neuron.

Chomskyan Theory refers to the theory of universal grammar, the idea that all languages is innate and notion.

Core Library refers to the library that receives instructions and create calculations or actions based on the instructions.

Crowdsourcing refers to the practice of gathering information from a large number of people.

ISO 9126 is the international standard for the evaluation of computer software's quality.

Lexicon refers to the vocabulary of a particular language, an individual speaker or group of speakers, or a subject, sometimes another term for dictionary.

Modular refers to a software design architecture that separates stand alone components of a program.

Parser refers to the receiver of inputs in the form of sequential source program instructions.

Sentiments refers to the formal of though, opinion, or idea based on a feeling about a situation or a way of thinking about something.

Opinions refers to the belief, judgement, or way of thinking about something.

Comments refers to the spoken or written statement that expresses an opinion about someone or something.

Linkifier refers to the application of the SAMUEL API that can parse supported forum websites and analyze it to documents of sentiments and syntactic tokens.

CHAPTER 3

METHODOLOGY

This chapter focused on the methods used in conducting the study. This includes the Project Design, Project Development, Operation and Testing Procedures, and the Evaluation Process.

Project Design

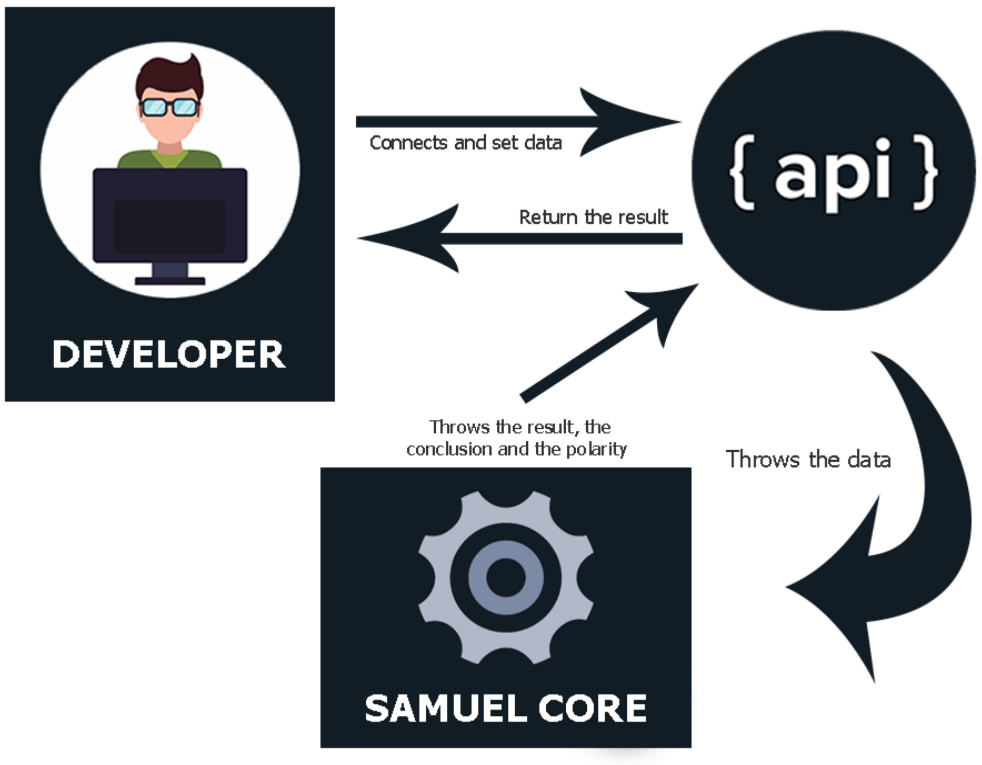


Figure 3. Architectural System Design of the Developed System

As shown in Figure 3.1, the user or the developer needs to connect first to the API and specify the area in which the API will draw the statements from. The API will check the API key and validate the data specified by the user. Once it is authenticated the API will then throw the data given by the user to the Samuel Core. Once caught by the Samuel Core, it will process the data and classify each statement in two different polarities, positive and negative, and it will draw out its conclusion from the statements.

The process always starts with what is known as *normalization*. Different NLP and analytics system requires different forms of normalization to be feed to them, so a normalized text for one system cannot always be considered “normalized” by the other. Normalizing a text can be as simple as removing stop words in a text, words which do not contain any significance e.g. *the, and, that, to,* and then returning a sequence of tokens of words. There are also various techniques that can be done in the process of text normalization, such as case conversion, correcting spellings, and *stemming* and *lemmatization* of words which reduces the inflectional forms of words to a common base form in the text e.g. *car, cars, car's, cars' => car*. After the text has been normalized, the system can now start processing those data, first off with the sentiment analysis.

The sentiment analyzer uses an unsupervised lexicon-based technique instead of a supervised machine learning technique such as *SVM (Support Vector Machines)* and *ME (Maximum Entropy)* due to the lack of a well-labeled training dataset. In this technique, the analyzer uses a lexicon, particularly the *VADER Lexicon (Valence Aware Dictionary and Sentiment Reasoner),* which is a dictionary or book of words that have associated polarity scores with them, on how positive or negative the word is. However, the use of lexicon does not mean it is inferior to other existing techniques, indeed, existing lexicons such as *VADER* can even outperform machine learning models in the same domain which they were trained for as well as human raters in correctly classifying sentiments into positive, neutral, or negative classes.

The analyzer checks the existing words in the lexicon, then assigns a corresponding valence to the word in which the valence represents the polarity score of the word. After the initial valence has been set for each word, it will now check the context in which the word is contained. First of all, it will check whether the word is capitalized in which it demands emphasis of the word, then it later goes on to check the surrounding adverbs and adjectives of the word and adjust the valence appropriately according to the words' distances, and whether those words are negated or not. The analyzer also checks for existing emoticons, and even exclamation and question marks around the text for additional emphasis. Then, it aggregates the valence of each word to give the final sentiment of the text.

Then let's proceed then to the next system which is the topic modeller. Topic modelling involves an extra step after normalization which is known as *feature extraction*. This process, often related with *dimensionality reduction*, builds derived values that are informative and non-redundant, namely *features*, from an initial set of measured data.

Features, are unique and measurable characteristics or properties for each given observation of phenomenon in a dataset. For this matter, feature extraction can largely reduce the resources required to perform complex analysis on large data since it drastically reduces the number of variables involved in the computation whilst keeping the data sufficiently accurate. Feature extraction aligned towards text data commonly involves transforming the textual data into a numeric data since computers can only make sense of numbers and not words and the most known concept about this is the *Vector Space Model.*

Vector space model is a very popular concept for information retrieval and document ranking. It is composed of numeric vectors of specific terms which forms the vector dimension. Hence, for each document *D* in a document vector space *VS*, the number of dimensions or columns in each document will be the total number of unique words in all of the documents in the vector space. Each of containing weights that can be either a word frequency in a document or the average frequency of occurrence. There are existing several feature extraction techniques for this model such as the *Bag-of-Words* and the *TF-IDF (Term Frequency - Inverse Document Frequency)* model.

The topic modeller uses an unsupervised approach which is known as *LDA (Latent Dirichlet Allocation).* It is a three-level hierarchical probabilistic directed acyclic model, known as *Bayesian model* from *Bayesian statistics*, in which each document is modeled as a mixture of a definite set of topics. The number of topics to be defined by the technique are predetermined prior to the creation of the model. There also exists two (2) hyperparameters of *Dirichlet distribution* alpha α which sets the per-document topic distribution and beta β which sets the per-topic word distribution. In such case, the dirichlet distribution $p(\theta \mid \alpha)$ of order $K \geq 2$ with parameters $\alpha=\{\alpha_1, ..., \alpha_K: \alpha_i > 0\}$ and where $\theta=\{\theta_1, \ldots, \theta_K: \theta_i \geq 0, \sum_{i=1}^{K} ■ \theta_i=1\}$, which is a probability simplex, has a

probability density function of

$$f(x_1, \ldots, x_K; \alpha_1, \ldots, \alpha_K) = \frac{1}{\mathrm{B}(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1},$$

Where the *Beta function* is written in terms of the

Gamma function

$$\mathrm{B}(\boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)},$$

$$\Gamma(n) = (n-1)!$$

Dependencies among variables can be thoroughly represented by the *plate notation*, a method of efficiently representing repeating variables in a graphical model of *Bayesian inference*.

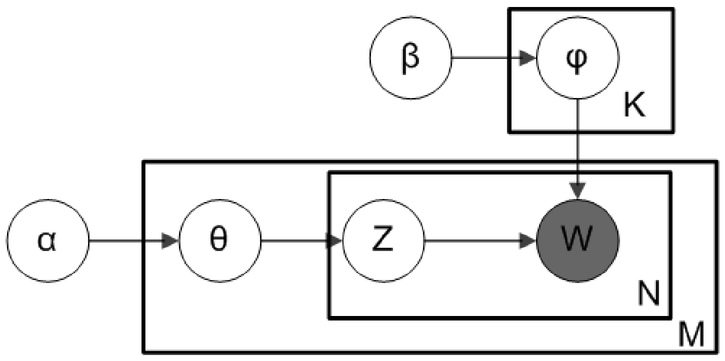


Figure 4. Bayesian Inference

LDA Plate Notation

K is the number of topics

N is the number of words in the document

M is the number of documents to analyze

α is the parameter of the Dirichlet prior on the per-document topic distributions

β is the parameter of the Dirichlet prior on the per-topic word distribution

θ_m is the topic distribution for document m

φ_k is the word distribution for topic k

z_{mn} is the topic for the nth word in document m

w_{mn} is the specific word.

In this regard, high alpha will make documents appear more similar to each other and high beta will make topics more similar to each other, affecting the Dirichlet distribution θ and φ respectively. It can also be seen that words *w* is the only observable variable and the rest are latent variables in which *w* and *z* are of multinomial.

The process starts with initializing the necessary parameters, then for each document, randomly initialize each word to one of the *k* topics. Afterwards, for each word *w* in each document *m*, word *w* is reassigned with a topic *k* with probability $p(z_n \mid \theta)$ $x\ p(w_n \mid z_n, \varphi_k)$ considering all other words and their topic assignments. Once the process is repeated several times, the model will be able to reach a roughly stable state in which one could properly estimate the mixture of topics in a document and the words associated to each topic.

The next system is the summarizer, which provides several ranking and reranking unsupervised extraction-based techniques to summarize the documents. But before one could go into any ranking algorithms, one should established, a *cosine similarity matrix* which is built from the TF-IDF model

$$\textbf{idf-modified-cosine}(x, y) = \frac{\sum_{w \in x,y} \textbf{tf}_{w,x} \textbf{tf}_{w,y} (\textbf{idf}_w)^2}{\sqrt{\sum_{x_i \in x} (\textbf{tf}_{x_i,x} \textbf{idf}_{x_i})^2} \times \sqrt{\sum_{y_i \in y} (\textbf{tf}_{y_i,y} \textbf{idf}_{y_i})^2}},$$

Which can be seen derived from the mathematical equation

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|_2 \|\mathbf{B}\|_2} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}}$$

This matrix constructs a *similarity graph* containing numeric floating-point values that determines how similar the document is to the rest of the documents. Since each document can be represented by a vector of *N*-dimension, one could compare it with other documents with their corresponding vectors and thus calculate its similarity with cosine where the value of 1 represents total similarity and the value of 0 represents total difference of documents. Hence, the diagonal part of the matrix is indeed a series of 1 since the corresponding document is totally similar to itself. A threshold can be set within the matrix, in which if a similarity score falls the indicated threshold, it is immediately converted to a value of 0; otherwise, a value of 1.

After the cosine similarity matrix is provided, ranking algorithms can now be properly implemented. Let's start with the default summarizer used by the system which is *DivRank (Diverse Rank)*. It is a novel ranking algorithm motivated from a general time-variant random walk, known as *vertex-reinforced random walk*, in an information network that balances prestige and diversity among top ranked vertices. The algorithm has the form

$$p_{T+1}(v) = (1-\lambda)p^*(v) + \lambda \sum_{u \in V} \frac{p_0(u,v) \cdot N_T(v)}{D_T(u)} p_T(u)$$

Where

$$D_T(u) = \sum_{v \in V} p_0(u,v) N_T(v).$$

Here, $p_T(u, v)$ is reinforced with $N_T(v)$, which is the number of times the walk has visited v up to time T, since at time T+1, the random walk moves to state v with probability $p_T(u, v) \propto p_0(u, v)N_T(v)$ for any state u given that the random walk stays at state u at time T. Furthermore, when $\lambda = 1$, chosen in the interval [0, 1], it yields to a standard vector-reinforced random walk where $p^*(v)$ is a distribution which represents the prior preference of visiting vertex v.

It could be realized as a topic-sensitive distribution where one can see the same in the *personalized PageRank,* while $p_0(u, v)$ is the "organic" transition probability prior to any reinforcement which can be estimated in a regular time that a random homogenous walk. It is assumed that there is always an “organic” link from a vertex to itself.

$$p_0(u,v) = \begin{cases} \alpha \cdot \dfrac{w(u,v)}{deg(u)}, & \text{if } u \neq v \\ 1-\alpha, & \text{if } u = v. \end{cases}$$

From an initial uniform probability distribution, in which the ranking of sentences will take place, as an initial state *u*, several iterations must be performed until a given threshold is met. In such case, the distribution will eventually reach a stable state, known as a *stationary distribution*, wherein the sentences can be ranked according to the value of distribution and the summarization of the documents can now be performed choosing the top *N* sentences.

Other ranking algorithms like *LexRank* (*Lexical PageRank*), which will be discussed next, will have some subtle similarities in the process.

$$p(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} \frac{\text{idf-modified-cosine}(u,v)}{\sum_{z \in adj[v]} \text{idf-modified-cosine}(z,v)} p(v)$$

It directly uses the cosine similarity matrix in its computation where *N* is the number nodes in the similarity graph and *d* is the *damping factor*, similar to the λ constant in DivRank, which assures the convergence of the method. Again, from a uniform distribution, the distribution is then re-iterated several times until it becomes a stationary distribution. The formula of LexRank is constructed from the existing *PageRank* formula, which was first proposed for computing web page prestige, and still serves as the underlying mechanism behind the Google search engine.

The next one that will be examined can be viewed as both a ranking algorithm and reranking algorithm in terms of its process, namely *Grasshopper* (*Graph Random-walk with Absorbing StateS that HOPs among PEaks for Ranking*). It incorporates centrality, diversity, and priority and is achieved through implementing a framework of an *absorbing Markov chain random walks*. A random walk is defined in a graph in which the ranked items become an absorbing state, dragging down the significance of similar unranked items. The formal definition of algorithm is to create an initial Markov chain *P* from a simillar graph *W*, a prior ranking *r*, and a λ constant which balances the two.

$$P = \lambda\tilde{P} + (1-\lambda)\mathbf{1}\mathbf{r}^{\top},$$

The $\tilde{P}$ is a matrix in which *W*: $\tilde{P}_{ij}=w_{ij}/\sum_{k=1}^{n} w_{ik}$, *so that* P_{ij} is the probability that the walker moves form *j* to *i*, teleporting the random walk *P* by interpolating each row with *r*. Afterwards, compute the stationary distribution of *P* then pick the first item of $g_1 = argmax_i\ \pi_i$. Subsequently, repeat until all items are ranked in which ranked items turns into absorbing states, then compute the expected number of visits *v* for all remaining items, picking the next item $g_{|G|+1} = argmax_i\ v_i$ as next ranked item, accordingly *v* equates to

$$\mathbf{v} = \frac{N^{\top}\mathbf{1}}{n-|G|},$$

Where *G* is the set of the items ranked so far, and *N* is known as the *fundamental matrix*

$$N = (\mathbf{I} - Q)^{-1},$$

Such that matrix *Q*, which corresponds to row of unranked items, is taken from matrix *P* wherein ranked items are placed before the unranked ones; thus, *P* can be written as

$$P = \begin{bmatrix} \mathbf{I}_G & \mathbf{0} \\ R & Q \end{bmatrix}$$

The last one that we will be discussing here is a pure reranking algorithm known as *Maximal Marginal Relevance* (*MMR*). It aims to reduce redundancy while maintaining query relevance in retrieved documents. The motivation comes from the fact that even though pure relevance is necessary to rank such documents appropriately, there too exists the fact that there is a vast sea of relevant documents that contains partially or fully duplicative information, especially nowadays, which means utilization of pure relevance for ranking documents is not enough.

The algorithm establishes a “relevant novelty” criterion in which the metric is a linear combination of relevance and novelty independence, this combination is called *marginal relevance*. In this sense, a document has a high marginal relevance if it is both relevant to the query and contains minimal similarity to the previous selected documents, and as such, the method is called maximal marginal relevance since its objective is to maximize the marginal relevance.

$$MMR \overset{\text{def}}{=} Arg \max_{D_i \in R \setminus S} \left[\lambda(Sim_1(D_i, Q) - (1-\lambda) \max_{D_j \in S} Sim_2(D_i, D_j)) \right]$$

Given the definition above, *R* is the ranked list of documents; D_i is the current document in process; *Q* is the query or user profile; *S* is the subset of documents in *R* already selected, making *R*/*S* the set difference in which it is the set of as yet unselected documents in *R; Sim*$_1$ is the similarity metric used in the ranked list; *Sim*$_2$ can choose to either use the same metric or different ones. Furthermore, if parameter $\lambda = 0$, it computes a maximal diversity ranking among documents and when $\lambda = 1$, it incrementally computes a standard relevance-ranked list.

The core will throw the result back to the API, and the API will return the result to the user and will be displayed by the user however he likes to display it.

Project Development

The project will start by gathering and researching the information about the project and should be end up with the completed output. It considers how to determine the project development technique to be use for the application and the steps to implement to make it possible. In defining process to different techniques, it considers to be implemented based on the desired output.

a. Software Components

- Planned and designed the flow of core software for the API
- Encoded the system flow
- Developed all the modules
- Tested the program from various benchmarking tools

Operation and Testing Procedure

The operation and testing for the research project will undertake to tests for inconsistencies and errors of the developed project using the JavaScript and Python. The application was modified and debugged in order to pass the common software criteria for testing. To operate and test the API, the following steps follows:

Table 1.

Modules and Testing Procedures

Modules	Steps Undertaken
1. Linkifier	a. Paste the link b. Select the Linkify Button then proceed
2. API Key Creation	a. Register to SAMUEL API Website b. Go to API Keys Menu c. Click the Add Key Button d. Enter the Name of the Key then Click the Add Button
3. API Key Validation	a. Copy the API Key b. Use the API key to connect to SAMUEL API Core
4. SAMUEL Core a. Normalization b. Feature Extraction c. Topic Modelling d. Summarizer	a. Set all the Data needed b. Post request to SAMUEL API Core c. Display the Return Data • Polarity & Percentage • Summarize Text • Dashboard

Evaluation Procedure

The evaluation procedure will determine the acceptability of the project. The project will be evaluated by the selected 30 IT/CS/IS students of Technological University of the Philippines.

The following steps were followed:

1. The respondents were given a questionnaire by the researchers to answer it correctly;
2. The researchers discussed the flow of the project for which users tested the performance of the system;
3. The respondents evaluated the system to determine the acceptability of the project; and
4. The frequency will be used to know the result of the evaluation using the formula (x/n)*100, where x is the sum of rating of respondent and n is the number of respondents.

Respondents' Profile

The respondents are made up of 30 individuals that are IT/CS/IS students of Technological University of the Philippines

Evaluation Instrument

The respondents answered the evaluation instrument which was adopted from ISO 1926 Evaluation-Criteria for quality software such as Understandability, Functionality, Usability, and Portability. As shown in the table, the respondents used a rating scale of 1 to 4 wherein 4 corresponds to “Highly Accepted” and 1 corresponds to “Not Acceptable”

Table 2.

Numerical Rating and its Equivalent

Numerical Rating	**Equivalent**
4	Highly Accepted
3	Very Acceptable
2	Acceptable
1	Not Acceptable

CHAPTER 4

RESULT AND DISCUSSION

This chapter discusses the Result of the Study. It discusses the Project Description, Project Structure, Project Capabilities and Limitations, and Project Evaluation.

Project Description

The researchers developed an API (Application Programming Interface) that can analyze sentiments. To create a conclusion, to classify the sentiments into two polarities, and give their respective percentage positive or negative, and give their respective percentage.

SAMUEL API Core

A python core that accepts the sentiments to process it. From normalizing the text, classifying the polarity and its percentage, making a topic model out of the sentiments, to creating a compact and concise conclusion, and to returning the result to the API.

Modular API – For Developers

A modular API that can be used and customized easily by any developers on how they want to use it. They just need to register to the SAMUEL website and generate an API key that they will use to POST request to the SAMUEL API Core, and then the core will return the results of the posted data.

Linkifier – For Non-Developers

A web application of the Samuel API that lets the normal user to analyze sentiments from other websites like Reddit, YouTube and other supported forum websites. They just need to paste the link in the Linkifier website and let the SAMUEL API do the rest.

Project Structure

SAMUEL API has a website where the developers can register and generate an API key to use for their projects. And also the Linkifier for the non-developers who wants to analyze a corpus of sentiments.

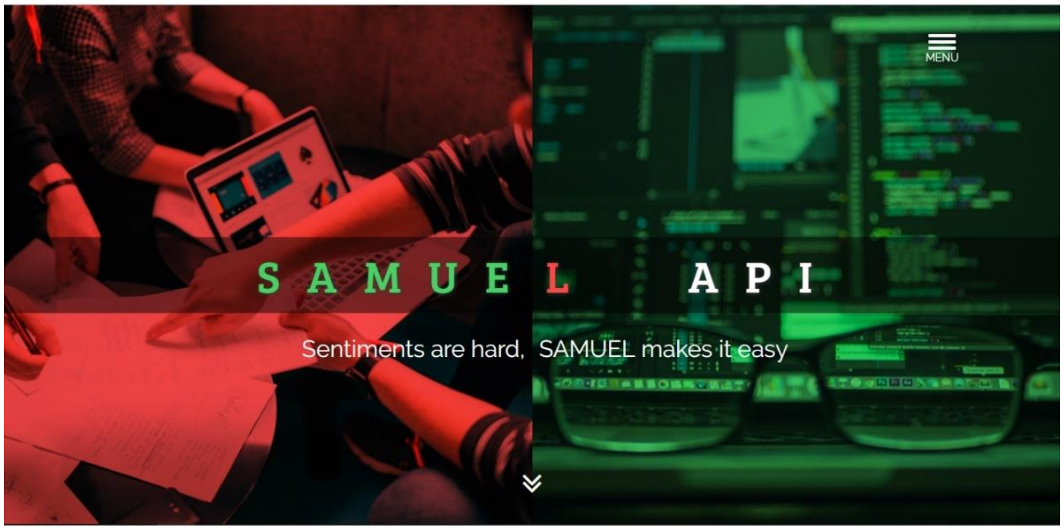
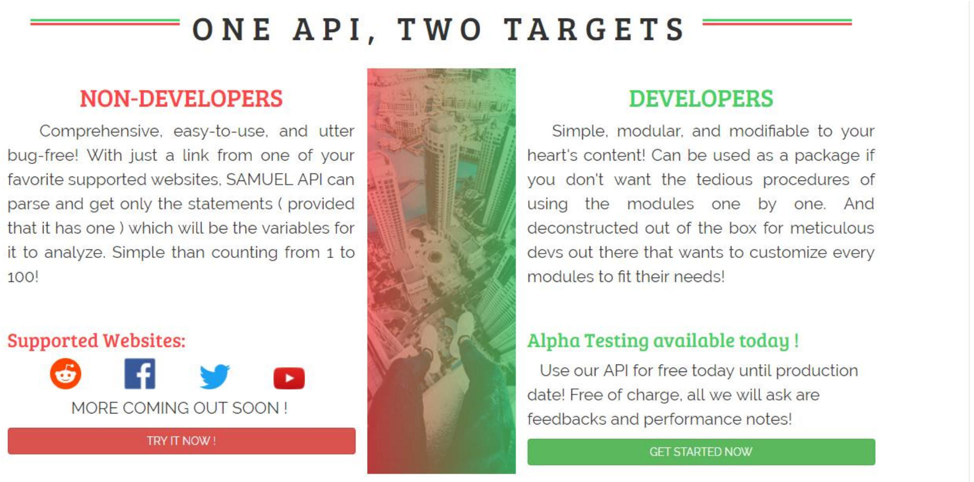


Figure 5. Landing Page

Figure 5 shows the landing page of the SAMUEL API.



ONE API, TWO TARGETS

NON-DEVELOPERS

Comprehensive, easy-to-use, and utter bug-free! With just a link from one of your favorite supported websites, SAMUEL API can parse and get only the statements (provided that it has one) which will be the variables for it to analyze. Simple than counting from 1 to 100!

Supported Websites:

MORE COMING OUT SOON !

DEVELOPERS

Simple, modular, and modifiable to your heart's content! Can be used as a package if you don't want the tedious procedures of using the modules one by one. And deconstructed out of the box for meticulous devs out there that wants to customize every modules to fit their needs!

Alpha Testing available today !

Use our API for free today until production date! Free of charge, all we will ask are feedbacks and performance notes!

Figure 6. About SAMUEL API Page

Figure 6 shows the information about the SAMUEL API

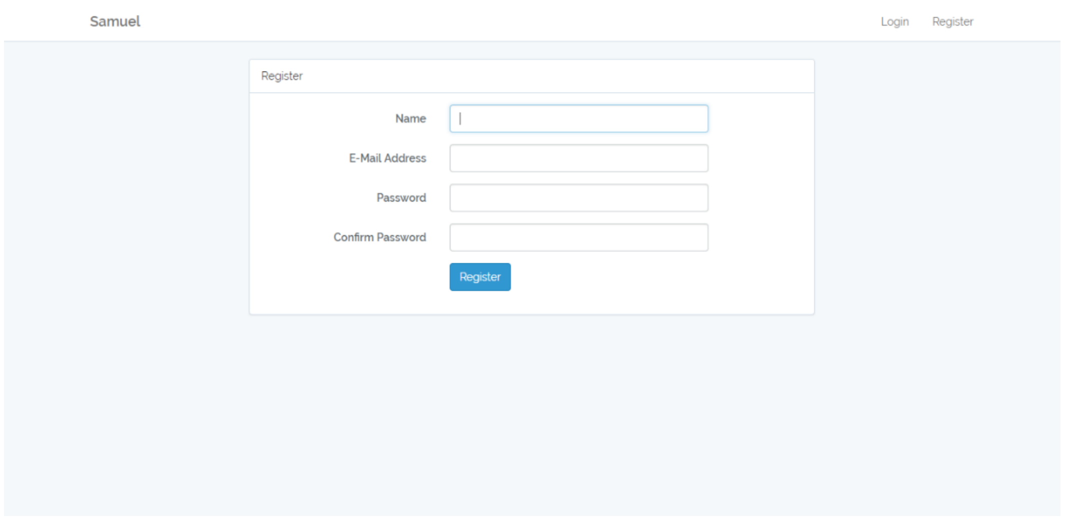
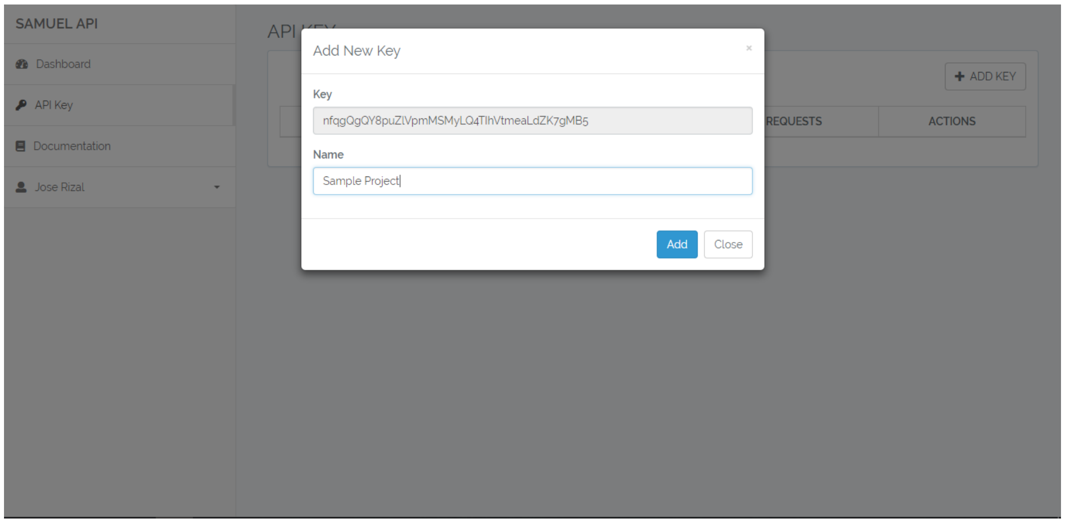


Figure 7. Developer's Registration Page

Figure 7 shows the registration page for the developers who will use the SAMUEL API.

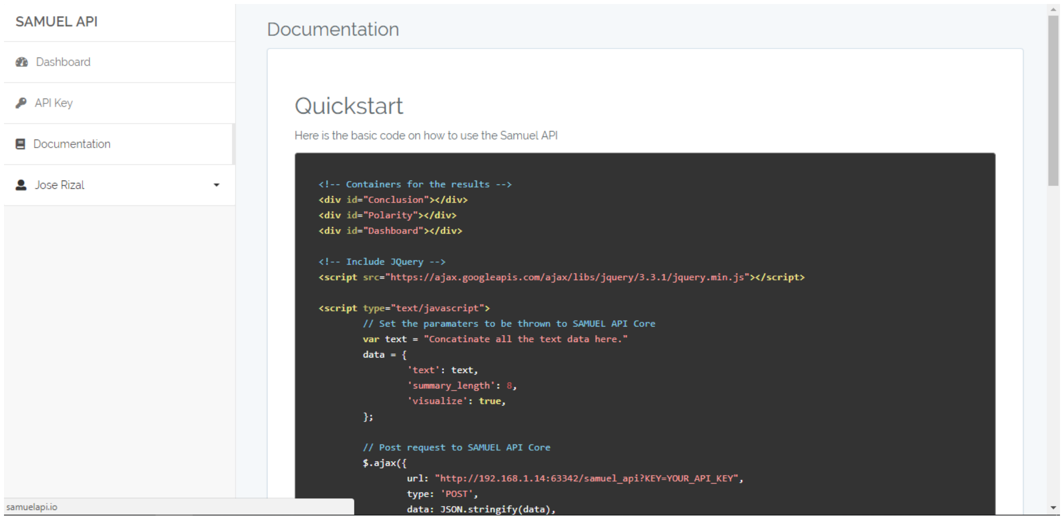


Key: nfqgQgQY8puZlVpmMSMyLQ4TlhVtmeaLdZK7gMB5

Name: Sample Project

Figure 8. API Key Creation

Figure 8 shows the page where the developer can add a new API Key and name it based on the project he will use it to.



Documentation

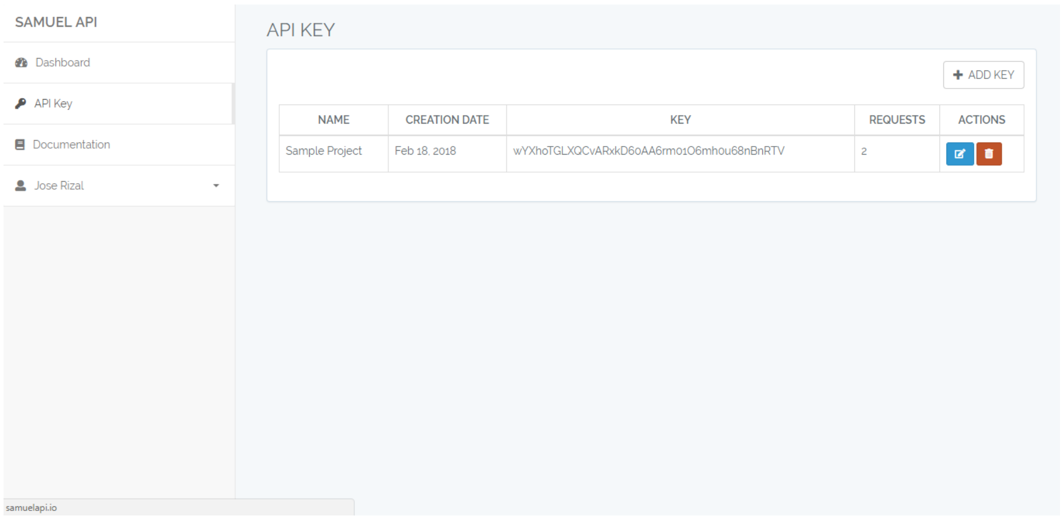
Quickstart

Here is the basic code on how to use the Samuel API

```
<!-- Containers for the results -->
<div id="Conclusion"></div>
<div id="Polarity"></div>
<div id="Dashboard"></div>

<!-- Include JQuery -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script type="text/javascript">
        // Set the paramaters to be thrown to SAMUEL API Core
        var text = "Concatinate all the text data here."
        data = {
                'text': text,
                'summary_length': 8,
                'visualize': true,
        };

        // Post request to SAMUEL API Core
        $.ajax({
                url: "http://192.168.1.14:63342/samuel_api?KEY=YOUR_API_KEY",
                type: 'POST',
                data: JSON.stringify(data),
```

Figure 9. Documentation Page

Figure 9 shows the documentation page where the developer can refer to on how to use the SAMUEL API.

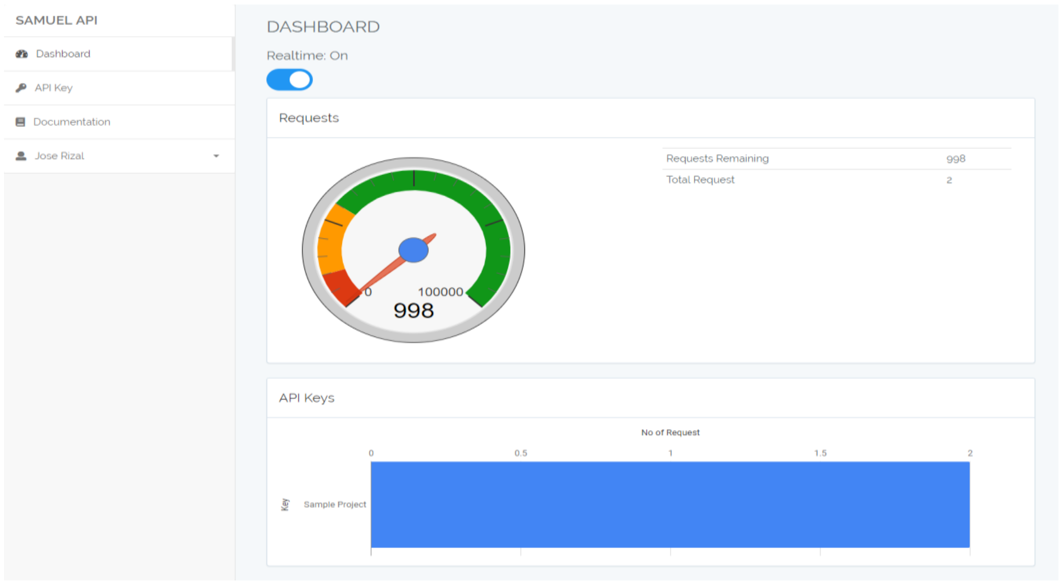


API KEY

NAME	CREATION DATE	KEY	REQUESTS	ACTIONS
Sample Project	Feb 18, 2018	wYXhoTGLXQCvARxkD60AA6rmo1O6mhou68nBnRTV	2	

Figure 10. API Keys Page

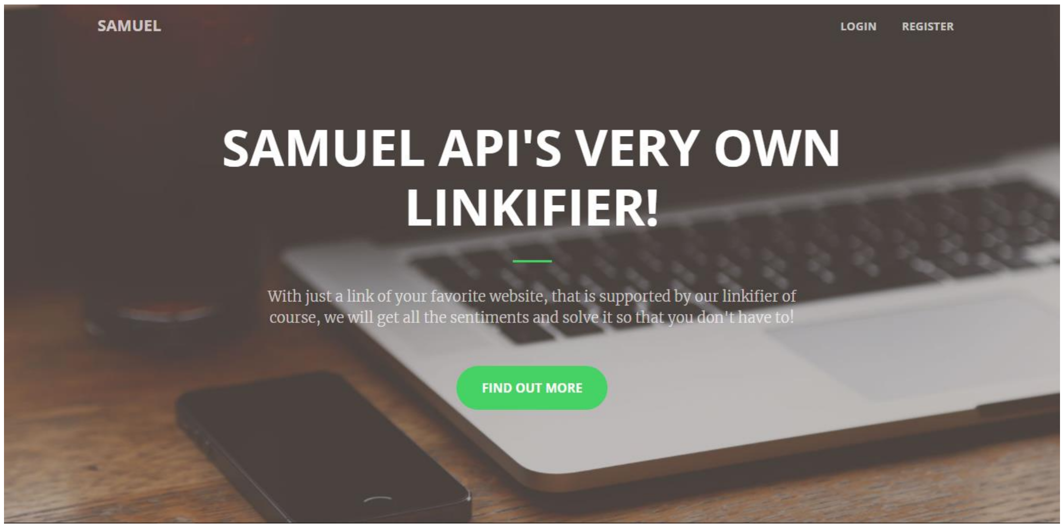
Figure 10 shows the API keys page, where the developers can manage his API keys, and to see how many requests a specific key have.



Requests Remaining	998
Total Request	2

Figure 11. Developer’s Dashboard Page

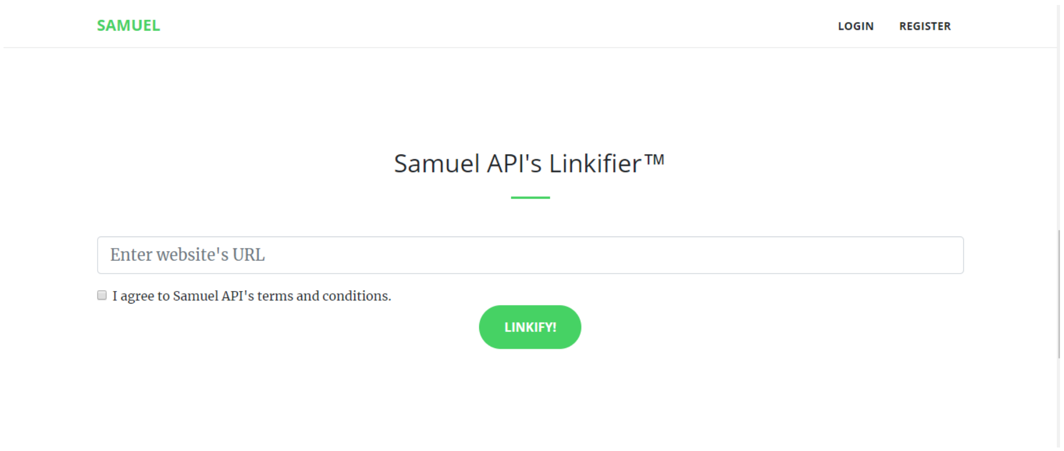
Figure 11 shows the developer’s dashboard page where he can track how many requests left on his account.



With just a link of your favorite website, that is supported by our linkifier of course, we will get all the sentiments and solve it so that you don't have to!

Figure 12. SAMUEL API Linkifier Landing Page

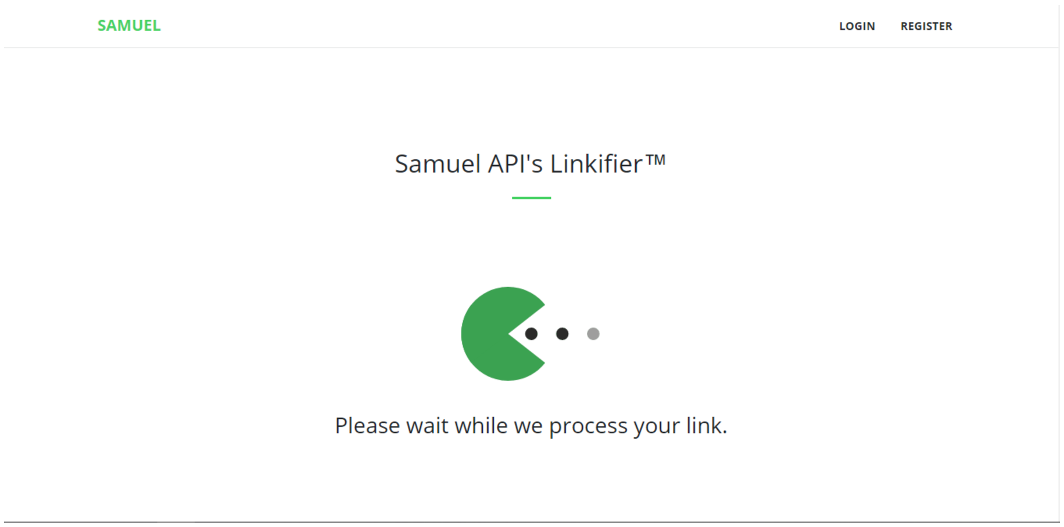
Figure 12 shows the SAMUEL API Linkifier page and the brief information about it.



I agree to Samuel API's terms and conditions.

Figure 13. Linkifier

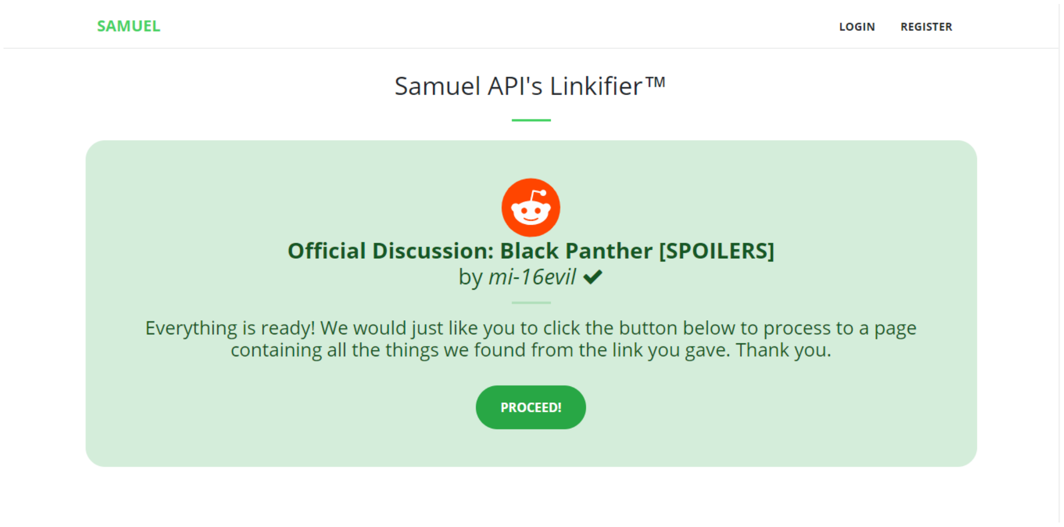
Figure 13 shows the linkifer where a normal user or non-developer user can paste the link of the supported website.



Please wait while we process your link.

Figure 14. Parsing Status Page

Figure 14 shows that the link pasted by the user is parsing.

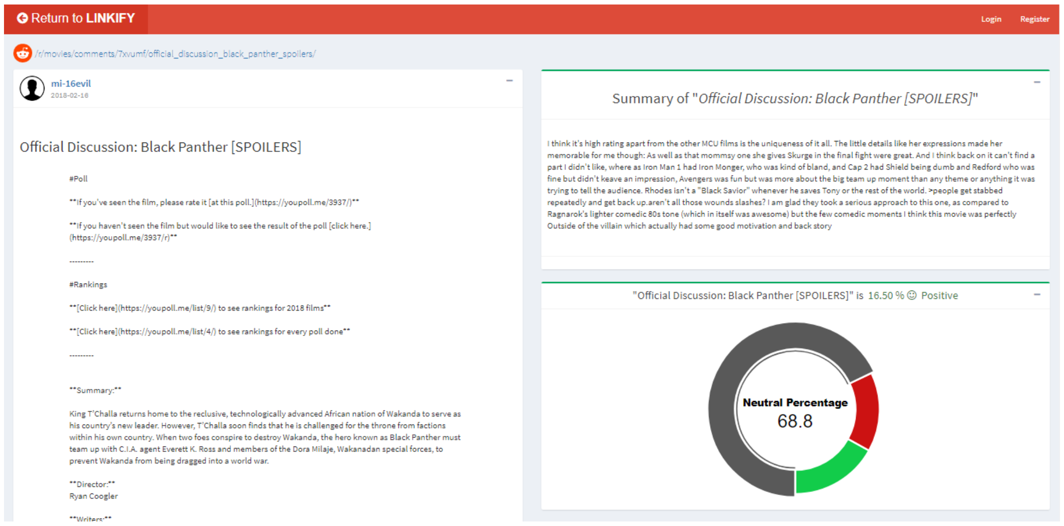


Official Discussion: Black Panther [SPOILERS]
by *mi-16evil* ✓

Everything is ready! We would just like you to click the button below to process to a page containing all the things we found from the link you gave. Thank you.

Figure 15. Parsing Complete

Figure 15 shows that the parsing is complete and about the details of the parsed link.



/r/movies/comments/7xvumf/official_discussion_black_panther_spoilers/

mi-16evil
2018-02-16

Official Discussion: Black Panther [SPOILERS]

#Poll

If you've seen the film, please rate it [at this poll.](https://youpoll.me/3937/)

If you haven't seen the film but would like to see the result of the poll [click here.](https://youpoll.me/3937/r)

#Rankings

[Click here](https://youpoll.me/list/9/) to see rankings for 2018 films

[Click here](https://youpoll.me/list/4/) to see rankings for every poll done

----

Summary:

King T'Challa returns home to the reclusive, technologically advanced African nation of Wakanda to serve as his country's new leader. However, T'Challa soon finds that he is challenged for the throne from factions within his own country. When two foes conspire to destroy Wakanda, the hero known as Black Panther must team up with C.I.A. agent Everett K. Ross and members of the Dora Milaje, Wakanadan special forces, to prevent Wakanda from being dragged into a world war.

Director:
Ryan Coogler

Writers:

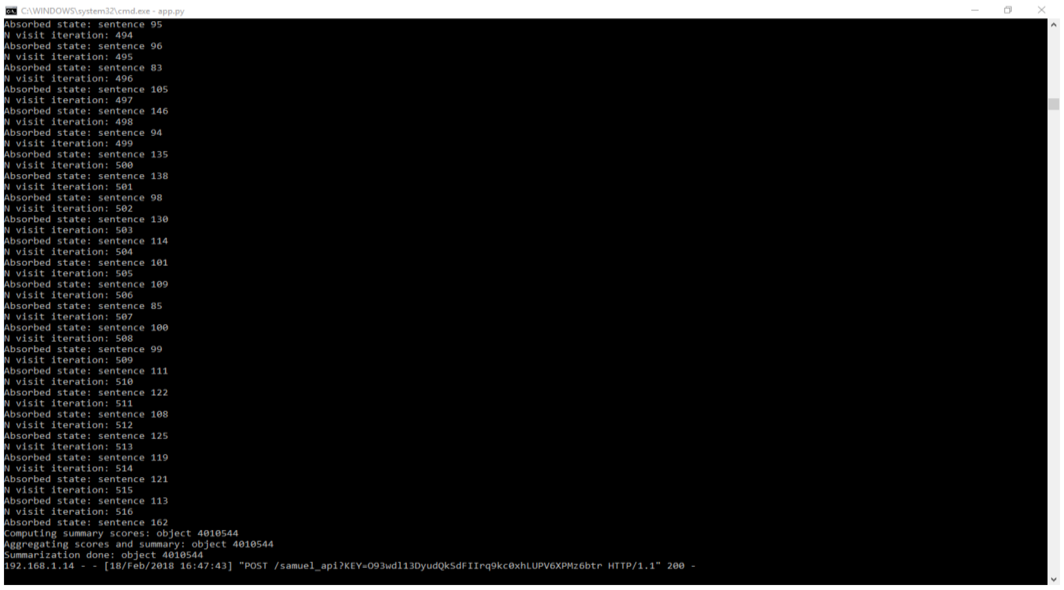
Summary of "*Official Discussion: Black Panther [SPOILERS]*"

I think it's high rating apart from the other MCU films is the uniqueness of it all. The little details like her expressions made her memorable for me though: As well as that mommsy one she gives Skurge in the final fight were great. And I think back on it can't find a part I didn't like, where as Iron Man 1 had Iron Monger, who was kind of bland, and Cap 2 had Shield being dumb and Redford who was fine but didn't keave an impression, Avengers was fun but was more about the big team up moment than any theme or anything it was trying to tell the audience. Rhodes isn't a "Black Savior" whenever he saves Tony or the rest of the world. >people get stabbed repeatedly and get back up.aren't all those wounds slashes? I am glad they took a serious approach to this one, as compared to Ragnarok's lighter comedic 80s tone (which in itself was awesome) but the few comedic moments I think this movie was perfectly Outside of the villain which actually had some good motivation and back story

"Official Discussion: Black Panther [SPOILERS]" is 16.50 % ☺ Positive

Figure 16. Linkifier Result Page

Figure 16 shows the result of the parsed link. The conclusion, the polarity and percentage, and the dashboard.



```
Absorbed state: sentence 95
N visit iteration: 494
Absorbed state: sentence 96
N visit iteration: 495
Absorbed state: sentence 83
N visit iteration: 496
Absorbed state: sentence 105
N visit iteration: 497
Absorbed state: sentence 146
N visit iteration: 498
Absorbed state: sentence 94
N visit iteration: 499
Absorbed state: sentence 135
N visit iteration: 500
Absorbed state: sentence 138
N visit iteration: 501
Absorbed state: sentence 98
N visit iteration: 502
Absorbed state: sentence 130
N visit iteration: 503
Absorbed state: sentence 114
N visit iteration: 504
Absorbed state: sentence 101
N visit iteration: 505
Absorbed state: sentence 109
N visit iteration: 506
Absorbed state: sentence 85
N visit iteration: 507
Absorbed state: sentence 100
N visit iteration: 508
Absorbed state: sentence 99
N visit iteration: 509
Absorbed state: sentence 111
N visit iteration: 510
Absorbed state: sentence 122
N visit iteration: 511
Absorbed state: sentence 108
N visit iteration: 512
Absorbed state: sentence 125
N visit iteration: 513
Absorbed state: sentence 119
N visit iteration: 514
Absorbed state: sentence 121
N visit iteration: 515
Absorbed state: sentence 113
N visit iteration: 516
Absorbed state: sentence 162
Computing summary scores: object 4010544
Aggregating scores and summary: object 4010544
Summarization done: object 4010544
192.168.1.14 - - [18/Feb/2018 16:47:43] "POST /samuel_api?KEY=O93wdl13DyudQkSdFIIrq9kc0xhLUPV6XPMz6btr HTTP/1.1" 200 -
```

Figure 17. SAMUEL API Core Server Log

Figure 17 shows the logs of SAMUEL API Core server for every request made the users.

Project Capabilities and Limitations

The following are the capabilities of the developed project.

SAMUEL API Core

1. The core can understand English and Filipino language.
2. Create a compact and concise conclusion out of it.
3. Classify the polarity and calculate the percentage.
4. Create a dashboard of the topics that occurred in the sentiments

Modular API – For Developers

1. A modular API that can be used on how the developers wants to.
2. No need for installation, developers only need an API Key and post request to the SAMUEL API Core.
3. Able to generate more than one API key per account.

Linkifier – For Non-Developers

1. Analyze a supported website and displays the result.
2. Easy to use by any users, with just a link of supported website.

The following are the limitations of the developed project.

SAMUEL API Core

1. The core can only understand English and Filipino language.
2. Processing time is based on the size of the sentiments

Modular API – For Developers

1. Requests are not unlimited

Linkifier – For Non-Developers

1. Can only analyze the supported websites listed below:
    a. Reddit;
    b. YouTube; and
    c. Other forum sites with a common forum layout

Test Results

The procedures taken by the researchers have been successfully conducted by testing the functionality, efficiency and portability.

Table 3.

Modules and Test Results

Modules	Result
1. Linkifier	a. The Linikifier parsed the pasted website b. The Linkifier displayed the results after clicking the proceed button. c. The website was opened using the installed browsers that most desktops and phones have.
2. API Key Creation	a. A new account was succesfully created. b. A new api key was succesully created.
3. API Key Validation	a. After post requesting the created API key, the SAMUEL Core returned the result.

	b. After post requesting a random API key, the SAMUEL Core returned “Invalid API Key”
4. SAMUEL Core	a. The core process the sentiments and yield a result. b. The API responds after post requesting to SAMUEL Core c. The Linkifier shows the Conclusion, Polarity, Percentage, and Dashboard after parsing the content of website link pasted. d. The accuracy of the polarity is directly proportional to the number of sentiments, it means that the more the sentiments, the more it will yield to an accurate result. e. After throwing the data to SAMUEL Core, it took an average of 30 seconds to process the sentiments gathered. f. The speed of internet connection and the size of the data thrown to the core are the factors of the slow processing time

Project Evaluation

The project was evaluated by 30 respondents consisting of IT, CS and IS students of Technological University of the Philippines Manila Campus. They determined the merit, worth and significance of the project based on the criteria that was set as standards. The calculated scale and frequency for each criterion is summarized in Table 6.

It can be shown that under the Functionality criterion 73% of the respondents rated it as highly acceptable in the sub-criteria suitability. In Accuracy 70% of the respondents rated it highly acceptable. While in Security 67% of the respondents rated it as highly acceptable.

Under Reliability criterion, 73% of the respondents rated it as highly acceptable in Maturity. In Fault Tolerance 53% of the respondents rated it as highly acceptable and 30% rated it as very acceptable. In reliability compliance 67% says it is highly acceptable.

In the efficiency criterion, 77% of the respondents rated it as highly acceptable in Understandability. In Learnability 60% of the respondents rated it as highly acceptable. In operability 67% says it is highly acceptable.

As for the maintainability criterion, 60% of the respondents rated it as highly acceptable in Analyzability. In Changeability 63% of the respondents rated it as highly acceptable. In Stability 60% says it is highly acceptable.

And for the portability criterion, 87% of the respondents rated it as highly acceptable in adaptability. In Instability 67% of the respondents rated it as highly acceptable. In portability compliance 83% says it is highly acceptable.

Table 6

Scale Rating and Equivalent Frequency of Respondent's Answer

Criteria	Scale				Frequency			
	4	3	2	1	4	3	2	1
Functionality								
1. Suitability	22	4	3	1	73%	13%	10%	3%
2. Accuracy	21	4	5	0	70%	13%	17%	0%
3. Security	20	6	4	0	67%	20%	13%	0%
Reliability								
1. Maturity	22	4	4	0	73%	13%	13%	0%
2. Fault Tolerance	16	9	4	1	53%	30%	13%	3%
3. Reliability Compliance	20	8	2	0	67%	27%	7%	0%
Usability								
1. Understandability	24	4	2	0	80%	13%	7%	0%
2. Learnability	18	10	2	0	60%	33%	7%	0%
3. Operability	22	5	3	0	73%	17%	10%	0%
Efficiency								
1. Time Behavior	23	6	1	0	77%	20%	3%	0%
2. Resource Utilization	25	4	1	0	83%	13%	3%	0%
3. Efficiency Compliance	21	5	4	0	70%	17%	13%	0%
Maintainability								
1. Analyzability	18	10	2	0	60%	33%	7%	0%
2. Changeability	19	10	1	0	63%	33%	3%	0%
3. Stability	18	5	7	0	60%	17%	23%	0%
Portability								
1. Adaptability	26	2	2	0	87%	7%	7%	0%
2. Instability	20	8	2	0	67%	27%	7%	0%
3. Portability Compliance	25	3	2	0	83%	10%	7%	0%
Total	380	107	51	2	70%	20%	9%	0%

Chapter 5

SUMMARY OF FINDINGS, CONCLUSIONS, AND RECOMMENDATIONS

This chapter presents the Summary of the Findings, Conclusions, and Recommendations based on the Results of the Evaluation, Remarks and Suggestions for the Improvement of the Study.

Summary

Based from the tests and evaluations conducted regarding the performance of the Sentiment Analyzing Machine that can Understand Enumerated Languages Application Programming Interface (SAMUEL API), the following are the findings of the study:

The SAMUEL API was created according to the desired processes of text translation, normalizer, classifier, topic modeler and summarizer. The API was built using Python that implements various methodology and algorithms within the field of natural language processing and unsupervised machine learning. Two types of user are specified in operating the AP: non-developer and developer. Non-Developers are within the boundary of using the website in requesting for the sentiments polarity, summary and topic data of any supported domain of the system. Developers are required to apply for registration and project creation to obtain the unique API key provided in our system that can be integrated within their own software projects, hence the documentation is provided for the developer’s operation references. The system was designed according to the satisfaction of the client, developer and based on the ISO 9126 standard evaluation.

Test results showed that the system is properly working. All the processes and operation presented were achieved.

Thirty respondents that consists of CS/IT/IS students as evaluators are affirmed that the system attained its intended purpose in integrating the SAMUEL API. The majority of the evaluators rated it as “HIGHLY ACCEPTABLE”.

Conclusions

In consideration of the objectives of the study and the results of the evaluation conducted, the following conclusions were derived:

1. The SAMUEL API is composed of five major processes namely text translation, text normalizer, text sentiment classifier, text topic modeler, and text summarizer;
2. Documents are evaluated trough English and Filipino language using the free machine translation tool that is extended into a library called Google Translate;
3. Sentiments polarity are classified using an unsupervised lexicon-based technique called VADER (Valence Aware Dictionary and Sentiment Reasoner).
4. Unsupervised approach is implemented for topic modeller called Latent Dirichlet Allocation (LDA);
5. Developed summarizer uses natural language processing technique and ranking algorithms such as Lexical PageRank, Diverse Rank, Graph Random-walk with

Absorbing States that HOPs among Peaks for Ranking (GRASSHOPPER) and Maximum Marginal Relevance;

6. The SAMUEL API core was created using Python programming language, JavaScript as its modular controller and JSON as its data-interchanger format;
7. Supported web domain of the system is extended for Reddit, Youtube and few local Internet forums;
8. The SAMUEL API supports two types of users, developer and non-developer;
9. The SAMUEL API was tested based on the processes given; and
10. The SAMUEL API was evaluated and thereby which was interpreted as “HIGHLY ACCEPTABLE” using the ISO 9126 Evaluation.

Recommendations

The following are the recommendations of the study:

1. Improve the normalization module in cleaning the data from special characters such as hyperlinks, icons, and other various character combination that are beyond scope. Also, in tokenizing sentences and tokens from raw and cluttered data, in which some are not recognized without the presence of proper structure of delimiters and words;
2. Implement an abstraction-based summarizer and NLG for the summarizer to achieve a more human-like summary of the text;
3. Implement an aspect classifier that would aggregate what does each percentage of the polarity – positive, negative, neutral – represents about a particular topic.

4. Implement a testing framework for SAMUEL that can thoroughly automate and analyze the efficiency and correctness of each module, such as using ROUGE-1 computing F1 scores and the like;
5. Implement a wider range of languages to be accommodated by the API that does not solely rely to Google Translate since it is not that accurate in translating some certain context and topics; Thus it can return a corrupt data sent by user;
6. Implement a thorough Named Entity Recognizer throughout the module so that the concerned modules, especially the normalizer, could work properly to identify significant words that have meaning in the given context;
7. Implement a GUI for the whole module that analyzes real-time how the data are processed so it can be seen where the data might get corrupted or altered in which the user does not intend. It can also be use for debugging purposes to actively improve the module;
8. Implement a wider range of options in the API in controlling the parameters of the machine learning algorithms used by the system; Additionally, a wider range also of algorithms to choose from, from unsupervised to supervised machine learning methods;
9. Expand the web domains that are supported by the system that features feedbacks, comments, reviews, discussions, etc; and
10. Use the study as research input for further improvement.

REFERENCES

Abhimanyu Chopra, Abhinav Prashar, Chandresh Sain. (2013). *Natural Language Processing.*

Application Program Interface. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Application_programming_interface

Beal, V. (n.d.). *API -application program interface*. Retrieved from webopedia.com: https://www.webopedia.com/TERM/A/API.html

Bordosi, J. (2000, October 30). *Regular Expressions Explained*. Retrieved from proftpd.org: http://www.proftpd.org/docs/howto/Regex.html

Buckler, C. (2013, November 15). *RegExper: Regular Expressions Explained*. Retrieved from sitepoint.com: https://www.sitepoint.com/regexper-regular-expressions-explained

C. Musto, G. Semeraro, M. Polignano. (n.d.). *A comparison of Lexicon-based approaches for Sentiment Analysis of microblog posts*. Retrieved from ceur-ws.org/: http://ceur-ws.org/Vol-1314/paper-06.pdf

Deriu, J. (2016, April 22). *Sentiment Analysis using Deep Convolutional Neural Networks with Distant Supervision*. Retrieved from e-collection.library.ethz.ch: http://e-collection.library.ethz.ch/eserv/eth:49518/eth-49518-01.pdf

Eisentein, J. (2016, November 21). *Unsupervised Learning for Lexicon-Based Classification*. Retrieved from arxiv.org: https://arxiv.org/abs/1611.06933

Eleanor Clark, Kenji Araki. (2011). Retrieved from sciencedirect.com: https://www.sciencedirect.com/science/article/pii/S1877042811024049

Flask - web framework. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Flask_(web_framework)

Gebremeskel, G. (2011). *Sentiment Analysis of Twitter Posts About News.*

Google Translate. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Google_Translate

Greene, L. (2016, Febuary 2). Retrieved from taus.net: https://www.taus.net/think-tank/articles/everything-you-ever-wanted-to-know-about-google-translate-and-finally-got-the-chance-to-ask

JavaScript. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/JavaScript

Journals, T. M. (2011, May 26). *Lexicon-Based Methods for Sentiment Nalysis*. Retrieved from mitpressjournals.org: https://www.mitpressjournals.org/doi/abs/10.1162/COLI_a_00049

JSON. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/JSON

Liddy, E. D. (2001). *Natural Language Processing.*

Link, S. (n.d.). *Soft Computing*. Retrieved from link.springer.com: https://link.springer.com/journal/500

Machine Learning. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Machine_learning

Machine Learning. (n.d.). Retrieved from sas.com: https://www.sas.com/en_ph/insights/analytics/machine-learning.html

MDN. (n.d.). *About JavaScript*. Retrieved from developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript

MDN. (n.d.). *Regular Expressions*. Retrieved from developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

Mejova, Y. A. (2012, May). *Sentiment Analysis within and across social media streams.*

NLP. (n.d.). Retrieved from webopedia.com: https://www.webopedia.com/TERM/N/NLP.html

Normalizing Text. (n.d.). Retrieved from docs.oracle.com: https://docs.oracle.com/javase/tutorial/i18n/text/normalizerapi.html

Pang, B. (2006, August). *Automatic Analysis of Document Sentiment*. Retrieved from cs.cornell.edu: http://www.cs.cornell.edu/home/llee/extra/pang-thesis.pdf

Philips, I. (2012, April 17). *What is Text Normalization*. Retrieved from ianp.org: http://ianp.org/2012/04/17/what-is-text-normalization/

Python - Programming Language. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Python_(programming_language)

Regular Expression. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Regular_expression

Rouse, M. (n.d.). *JavaScript*. Retrieved from searchmicroservices.techtarget.com/: http://searchmicroservices.techtarget.com/definition/JavaScript

Sentiment Analysis. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Sentiment_analysis

Soft Computing. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Soft_computing

Stanford. (2014, Febuary 26). *Computational Linguistics*. Retrieved from plato.stanford.edu: https://plato.stanford.edu/entries/computational-linguistics/

Steve. (n.d.). *Google Translate, Multi-Purpose Dictionary*. Retrieved from blog.thelinguist.com: https://blog.thelinguist.com/google-translate-doesnt-work

Support, G. (n.d.). *About Regular Expressions (regex)*. Retrieved from support.google.com: https://support.google.com/analytics/answer/1034324?hl=en

Text Mining. (n.d.). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Text_mining

Text Normalization. (2017, September 21). Retrieved from en.wikipedia.org: https://en.wikipedia.org/wiki/Text_normalization

Tromp, E. (2011, July). *Multilingual Sentiment Analysis on Social Media*. Retrieved from win.tue.nl: http://www.win.tue.nl/~mpechen/projects/pdfs/Tromp2011.pdf

Tyler Bladwin, Yunyao Li. (2015). *An In-depth Analysis of the Effect of Text Normalization in Social Media*. Retrieved from aclweb.org: http://www.aclweb.org/anthology/N15-1045

Uszkoreit, H. (1996). *CL Intro*. Retrieved from coli.uni-saarland.de: http://www.coli.uni-saarland.de/~hansu/what_is_cl.html

Appendix A

Evaluation Instrument

ISO 9126

EVALUATION INSTRUMENT FOR SENTIMENT ANALYZING MACHINE THAT CAN UNDERSTAND ENUMERATED LANGUAGES APPLICATION PROGRAMMING INTERFACE

Name (Optional):________________________ Course/Year/Section: ______

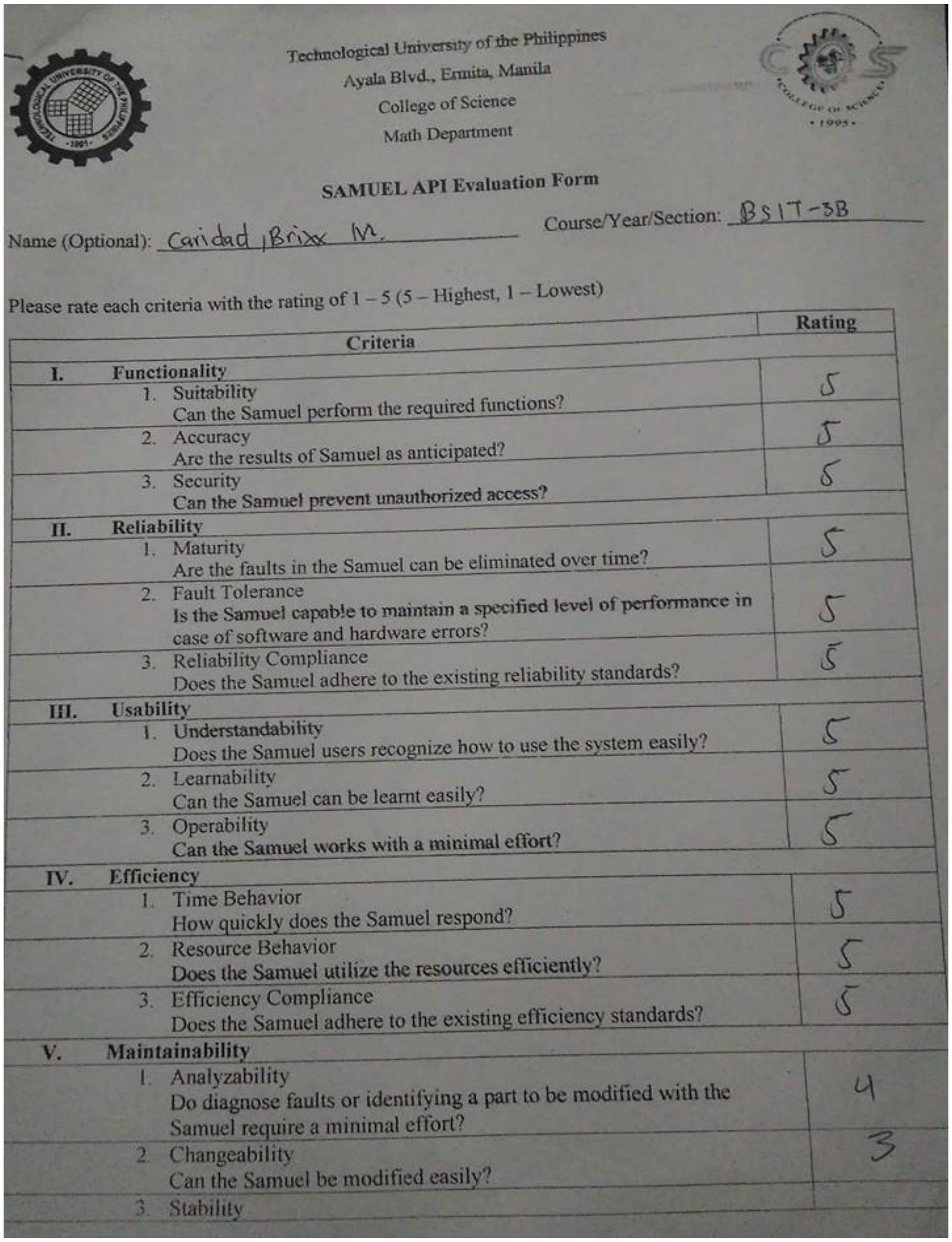
Please rate each criteria with the rating of 1 – 4 (4 – Highest, 1 – Lowest)

Criteria	Rating
I. Functionality	
1. Suitability Can the SAMUEL perform the required functions?	
2. Accuracy Are the results of SAMUEL as anticipated?	
3. Security Can the SAMUEL prevent unauthorized access?	
II. Reliability	
1. Maturity Can the faults of the SAMUEL can be eliminated over time?	
2. Fault Tolerance Is the SAMUEL capable to maintain a specified level of performance in case of software and hardware errors?	
3. Reliability Compliance Does the SAMUEL adhere to the existing reliability standards?	
III. Usability	
1. Understandability Does the SAMUEL users recognize how to use the system easily?	
2. Learnability Can the SAMUEL can be learnt easily?	
3. Operability Can the SAMUEL works with a minimal effort?	
IV. Efficiency	
1. Time Behavior How quickly does the SAMUEL respond?	
2. Resource Behavior Does the SAMUEL utilize the resources efficiently?	
3. Efficiency Compliance Does the SAMUEL adhere to the existing efficiency standards?	

V. Maintainability	
1. Analyzability Do diagnose faults or identifying a part to be modified with the SAMUEL require a minimal effort?	
2. Changeability Can the SAMUEL be modified easily?	
3. Stability Can the SAMUEL continue functioning after the change?	
VI. Portability	
1. Adaptability Can the SAMUEL be moved easily to the other environment?	
2. Installability Can the SAMUEL be installed easily?	
3. Portability Compliance Does the Samuel adhere to the existing portability standards?	

Appendix B

Answered Evaluation Sheet Sample



Technological University of the Philippines
Ayala Blvd., Ermita, Manila
College of Science
Math Department

SAMUEL API Evaluation Form

Name (Optional): Caridad, Brixx M. Course/Year/Section: BSIT-3B

Please rate each criteria with the rating of 1 – 5 (5 – Highest, 1 – Lowest)

Criteria	Rating
I. Functionality	
1. Suitability Can the Samuel perform the required functions?	5
2. Accuracy Are the results of Samuel as anticipated?	5
3. Security Can the Samuel prevent unauthorized access?	5
II. Reliability	
1. Maturity Are the faults in the Samuel can be eliminated over time?	5
2. Fault Tolerance Is the Samuel capable to maintain a specified level of performance in case of software and hardware errors?	5
3. Reliability Compliance Does the Samuel adhere to the existing reliability standards?	5
III. Usability	
1. Understandability Does the Samuel users recognize how to use the system easily?	5
2. Learnability Can the Samuel can be learnt easily?	5
3. Operability Can the Samuel works with a minimal effort?	5
IV. Efficiency	
1. Time Behavior How quickly does the Samuel respond?	5
2. Resource Behavior Does the Samuel utilize the resources efficiently?	5
3. Efficiency Compliance Does the Samuel adhere to the existing efficiency standards?	5
V. Maintainability	
1. Analyzability Do diagnose faults or identifying a part to be modified with the Samuel require a minimal effort?	4
2. Changeability Can the Samuel be modified easily?	3
3. Stability	

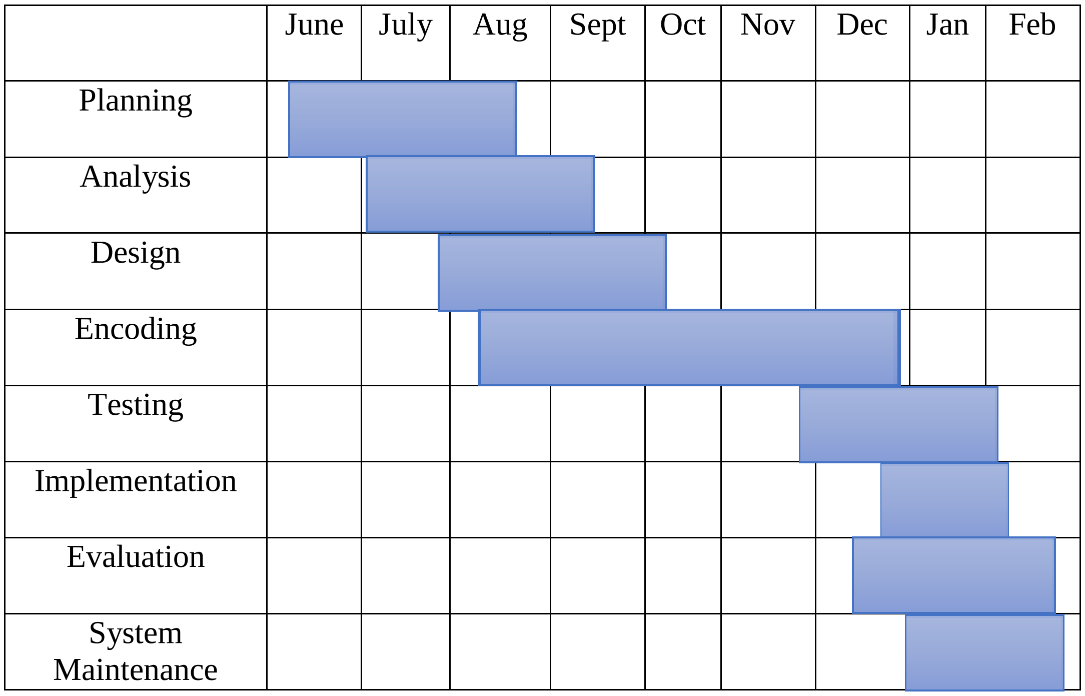
APPENDIX C

Summary of Respondents of Evaluation

Respondent	Functionality	Reliability	Usability	Efficiency	Maintainability	Portability
1	4.0	4.0	3.7	3.7	3.7	4.0
2	4.0	4.0	4.0	4.0	3.7	4.0
3	4.0	3.7	4.0	3.7	4.0	4.0
4	4.0	3.7	4.0	3.7	4.0	4.0
5	4.0	3.7	4.0	4.0	4.0	4.0
6	4.0	3.7	3.0	4.0	4.0	4.0
7	4.0	4.0	4.0	4.0	4.0	4.0
8	4.0	4.0	4.0	3.3	4.0	4.0
9	4.0	4.0	4.0	4.0	4.0	4.0
10	4.0	4.0	4.0	3.7	3.3	3.7
11	4.0	4.0	4.0	4.0	4.0	4.0
12	2.7	3.7	3.0	2.7	3.3	4.0
13	3.3	3.7	3.3	3.7	3.3	3.7
14	4.0	3.7	3.3	4.0	3.7	4.0
15	4.0	3.7	4.0	4.0	3.7	3.7
16	4.0	4.0	3.7	4.0	3.7	3.3
17	3.7	4.0	4.0	4.0	3.3	3.7
18	3.3	3.0	3.7	3.7	3.7	3.0
19	2.7	4.0	3.0	4.0	2.7	4.0
20	4.0	4.0	4.0	4.0	4.0	4.0
21	4.0	3.7	3.3	3.7	3.0	3.7
22	4.0	3.0	4.0	4.0	2.7	4.0
23	3.3	2.3	4.0	3.0	2.7	4.0
24	1.7	2.0	3.3	4.0	3.3	2.7
25	2.0	2.0	3.3	4.0	4.0	4.0
26	2.7	3.0	3.7	3.3	3.0	3.3
27	3.3	3.7	4.0	3.3	3.7	3.3
28	4.0	3.3	3.7	4.0	3.7	4.0
29	3.7	3.7	3.0	3.7	3.0	3.7
30	2.0	2.0	2.0	2.0	2.0	2.0

Appendix D

GANTT CHART



Appendix E

User's Manual

The system wa inspired in the process of opinion mining by implementing the use of natural language processing called Sentiment Analysis. The core discipline of the system is to deliver classified polarity, conclusions and dashboard of the topics based on the quantified and identified sentiments of documents. The distinction of instruction between the two target users (developer and non-developer) are as follows:

Developer – Initially, start by going to the Linkifier website. Afterwards, click the register button and proceed to fill up the information for registering a new account. Then, create a new project by clicking the Add new button. Eventually, you will write the project name and proceed to the next step to create the project and copy the provided unique API key for your application. In the meantime, you can read the system's documentation and dashboard in the left side section and derive the instructions trough integrating the SAMUEL API into your system or application.

Non-Developer – First, proceed by going to the Linkifier website. Next, access the SAMUEL API's Linkifier by clicking the try now in the non-developer's section. Then, enter the website's url in the textbox and type the number of maximum quantity of statement to be parsed and wait for few seconds. Finally, click the proceed button and for few minutes, the awaited results will be displayed upon including the discussion and documents from the parsed url.

Software Requirements

The system supports major browser such as Chrome, Firefox, Opera, Internet Explorer and Safari therefore the system requires an Internet connection.

Developers Information

Name: Dennis B. De Leon

Email: dennis.deleon@tup.edu.ph

Contact: +63 999-739-0325

Name: Christian Noel C. Molina

Email: devchrismolina@gmail.com

Contact: +63 926-143-7044

Name: Patrick Dale F. Rugayan

Email: pdale.rugayan@gmail.com

Contact: +63 947-219-9299

Name: Eric B. Sumalinog

Email: erc.sumalinog@gmail.com

Contact: +63 975-296-3434

Dennis B. De Leon

718 Benita St. Gagalangin Tondo Manila – 1013

+639997390325

dennis.deleon@tup.edu.ph

linkedin.com/in/hexxableyd

github.com/hexxableyd

OBJECTIVES

- I am highly determinate to work on a team that solves real-world problems on different areas of the industry by utilizing my technical, analytical and communicational skills that I've gathered through the years. Be part of the IT industry that will allow me to always conquer new heights, and adapt to the ever-increasing demands of the world.

PERSONAL INFORMATION

Age:	20
Date of Birth:	May 16, 1998
Nationality:	Filipino
Gender:	Male
Weight:	98 kg
Height:	5'6''
Civil Status:	Single

TECHNICAL EXPERTISE

Languages	Java, C#, C, PHP and Laravel (PHP Framework), Javascript and Ember (JS Framework), C, C++, C#, Python, SQL, HTML/CSS, Node JS (JS Runtime Script)
Frameworks and Tools	JUnit**,** Google AngularJS, Laravel, Django, Twitter Bootstrap, Git/Github
Software	Netbeans, Sublime, Jetbrains WebStorm / PhpStorm / PyCharm / IntelliJ IDEA, Microsoft Visual Studio 2017, WAMP/XAMPP

EDUCATIONAL BACKGROUND

Tertiary:	**Technological University of the Philippines** **Bachelor of Science in Computer Science** Ermita, Manila	2014 – 2018

CHARACTER REFERENCES

Available upon request

I hereby attest that all information contained here are true and correct

____

Dennis B. De Leon

Christian Noel C. Molina

B9 L9 Treelane II Phase C Bayan Luma III City of Imus,
Cavite, Philippines 4103

+63 926-143-7044 | (046) 471-7118

devchrismolina@gmail.com

linkedin.com/in/devchriscross

github.com/devchriscross

OBJECTIVES

- I am highly determinate to work on a team that solves real-world problems on different areas of the industry by utilizing my technical, analytical and communicational skills that I've gathered through the years. Be part of the IT industry that will allow me to always conquer new heights, and adapt to the ever-increasing demands of the world.

PERSONAL INFORMATION

Age:	20
Date of Birth:	September 29, 1997
Nationality:	Filipino
Gender:	Male
Weight:	60 kg
Height:	5'6''
Civil Status:	Sibgke

TECHNICAL EXPERTISE

Languages	Java, Python, C, C++, C#, Javascript, PHP, SQL, HTML/CSS, MATLAB
Frameworks and Tools	Netbeans, Sublime, Jetbrains WebStorm/PhpStorm/PyCharm/IntelliJ IDEA, Microsoft Visual Studio 2017, WAMP/XAMPP
Software	JUnit**,** Google AngularJS, Laravel, Django, Twitter Bootstrap, Git/Github

EDUCATIONAL BACKGROUND

Tertiary:	**Technological University of the Philippines** **Bachelor of Science in Computer Science** Ermita, Manila	2014 – 2018

CHARACTER REFERENCES

Available upon request

I hereby attest that all information contained here are true and correct

Christian Noel C. Molina

Patrick Dale F. Rugayan

37F D. Alger St. 2nd Avenue Caloocan City

+639472199299 / (02) 2941093

pdale.rugayan@gmail.com

http://patrickrugayan.jobs180.com/

OBJECTIVES

- A Computer Science graduate looking for a challenging work environment to utilize the skills and the knowledge I learned, and help me to grow as an IT professional.

PERSONAL INFORMATION

Age:	20
Date of Birth:	December 22, 1997
Place of Birth	Pampanga
Nationality:	Filipino
Gender:	Male
Weight:	65 kg
Height:	6’1’’
Civil Status:	Single

TECHNICAL EXPERTISE

Languages	C, C#, Visual Basic, Java, Android, HTML, CSS, JavaScript, Python, PHP, MySQL, JSON
Frameworks and Tools	Bootstrap, Code Igniter, JQuery, Laravel, Composer, Git
Software	Microsoft Office, Visual Studio, Visual Studio Code, NetBeans, Android Studio, Adobe Photoshop, Adobe Premiere Pro

EDUCATIONAL BACKGROUND

Tertiary:	**Technological University of the Philippines** **Bachelor of Science in Computer Science** Ermita, Manila	2014 – 2018
Secondary:	**Lakan Dula High School** Tondo, Manila	2010 – 2014
Elementary:	**Bario Obrero Elementary School** Tondo, Manila	2004 - 2010

CHARACTER REFERENCES

Available upon request

I hereby attest that all information contained here are true and correct

Patrick Dale F. Rugayan

Eric B. Sumalinog

51 Sto Tomas St Maypajo Caloocan City

+639752963434

ercsumalinog@gmail.com

linkedin.com/in/ercsumalinog

OBJECTIVES

- To use the course and training I obtained to maximize opportunities and contribution to a company that established a well-developed environment in utilizing logical, technical and analytical competence in an ever-changing demand in technologies.

PERSONAL INFORMATION

Age:	20
Date of Birth:	March 6, 1998
Place of Birth	Cebu City
Nationality:	Filipino
Gender:	Male
Weight:	45 kg
Height:	165 cm
Civil Status:	Single

TECHNICAL EXPERTISE

Languages	Java, Python, C#, PHP, JavaScript, HTML, CSS, SQL.
Frameworks and Tools	Laravel, Android, Twitter Bootstrap, Composer, Semantic UI, jQuery, Less, Sass
Software	Netbeans, Android Studio, IntelliJ IDEA, PyCharm, Visual Studio 2015, Visual Studio Code, Sublime Text, WAMP, MySQL Server, Github.

EDUCATIONAL BACKGROUND

Tertiary:	**Technological University of the Philippines** **Bachelor of Science in Computer Science** Ermita, Manila	2014 – 2018

CHARACTER REFERENCES

Available upon request

I hereby attest that all information contained here are true and correct

Eric B. Sumalinog