Ministry of Science and Higher Education of the Republic of Kazakhstan

Suleyman Demirel University

Shynggyskhan Turganbekov

Automatic tagging of tasks

THESIS

Presented in Partial Fulfilment for the

Master of Technical Sciences Degree in Computer Science

(degree code: 7M06102)

Department of Computer Science

Faculty of Engineering and Natural Sciences

Supervisor: **Assistant Professor, PhD Sultanova Nazerke**

Kaskelen 2023

Suleyman Demirel University
Faculty of Engineering and Natural Sciences
Department of Computer Science

Dean of Faculty

Associate Professor

PhD Zhamanov A.

«06» 06 2023

Topic of the thesis:

Automatic tagging of tasks

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science" SDU, 2021-2023

Head of Department _____ Assistant Professor, PhD Mukash Zh.

Academic Supervisor _____ Assistant Professor, PhD Sultanova N.

Master student _____ Shynggyskhan Turganbekov

Kaskelen 2023

Ministry of Science and Higher Education of the Republic of Kazakhstan

Suleyman Demirel University

Shynggyskhan Turganbekov

Automatic tagging of tasks

THESIS

Presented in Partial Fulfilment for the

Master of Technical Sciences Degree in Computer Science

(degree code: 7M06102)

Department of Computer Science

Faculty of Engineering and Natural Sciences

Supervisor: **Assistant Professor, PhD Sultanova Nazerke**

Kaskelen 2023

Suleyman Demirel University
Faculty of Engineering and Natural Sciences
Department of Computer Science

Dean of Faculty

Associate Professor

PhD Zhamanov A.

_____

« _____ »_____2023

Topic of the thesis:

Automatic tagging of tasks

Thesis submitted as part of the requirements for the award of the MSc in "7M06102 - Computer Science" SDU, 2021-2023

Head of Department _____ Assistant Professor, PhD Mukash Zh.

Academic Supervisor _____ Assistant Professor, PhD Sultanova N.

Master student _____ Shynggyskhan Turganbekov

Kaskelen 2023

Declaration

I confirm that this is my own work and the use of all material from other sources has been properly and fully acknowledged.

Shynggyskhan Turganbekov

2023

Acknowledgements

Thanks to the government of Republic of Kazakhstan for providing a grant and free education. Thanks to my supervisor for giving a great deal of knowledge of Machine learning. And for the fact that he professionally supervised the writing of the thesis, for his patience and for making my thesis understandable. Thanks for correcting me when I made mistakes and for motivating when there was no time. thank you for doing your job honestly. Thanks to my group mates for being with me on this path. And for my family for believing in me.

Dedication

This thesis is dedicated to my wife, Nazipa Aitu.

I also want to dedicate this thesis to my supervisor, Nazeke Sultanova. Her expertise has been essential to completing this work successfully.

I want to to dedicate this thesis to all the others who have supported me. Their contributions have been important in making this project a reality.

Abstract

In the modern era, the widespread application of natural language processing has led to a growing need for automated tagging systems. These tags hold significant importance in people's lives as they aid in search engine optimization and minimize time wastage.

The purpose of this research is to address the challenge of automating the tagging process for algorithmic problems. Specifically, our focus lies in developing a machine learning-based approach for accurately assigning difficulty tags to problem descriptions. While acknowledging the limitations of existing machine learning methods, we consider them as a foundational step for future investigations in this domain.

By leveraging advanced techniques in natural language processing and machine learning, this study aims to enhance the efficiency and effectiveness of automated task tagging systems. The outcomes of this research can potentially contribute to the development of improved methods for problem classification, facilitating more efficient information retrieval and task prioritization.

Аңдатпа

Қазіргі дәуірде табиғи тілді өңдеуді кеңінен қолдану таңбалаудың автоматтандырылған жүйелеріне қажеттіліктің өсуіне әкелді. Бұл тегтердің адамдардың өмірінде маңызы зор, өйткені олар іздеу жүйелерін оңтайландыруға көмектеседі және уақыт шығындарын азайтады.

Бұл зерттеудің мақсаты алгоритмдік міндеттер үшін таңбалау процесін автоматтандыру проблемасын шешу болып табылады. Атап айтқанда, біздің назарымыз проблемаларды сипаттауға күрделілік тегтерін дәл беру үшін машинамен оқытуға негізделген тәсілді әзірлеу болып табылады. Машина оқытудың қолданыстағы әдістерін шектеуді мойындай отырып, біз оларды осы саладағы болашақ зерттеулер үшін негіз қалаушы қадам ретінде қарастырамыз.

Табиғи тілді өңдеудің және машинамен оқытудың озық әдістерін пайдалана отырып, бұл зерттеу міндеттерді таңбалаудың автоматтандырылған жүйелерінің тиімділігі мен пәрмеңділігін арттыруға бағытталған. Бұл зерттеудің нәтижелері ақпаратты неғұрлым тиімді іздестіруге және міндеттерді басымдыққа алуға ықпал ете отырып, проблемаларды жіктеудің жетілдірілген әдістерін әзірлеуге әлеуетті ықпал етуі мүмкін.

Аннотация

В современную эпоху широкое применение обработки естественного языка привело к росту потребности в автоматизированных системах маркирования. Эти теги имеют большое значение в жизни людей, поскольку они помогают в оптимизации поисковых систем и минимизируют потери времени.

Целью этого исследования является решение проблемы автоматизации процесса маркирования для алгоритмических задач. В частности, наш фокус заключается в разработке основанного на машинном обучении подхода для точного присвоения тегов сложности описаниям проблем. Признавая ограничения существующих методов машинного обучения, мы рассматриваем их как основополагающий шаг для будущих исследований в этой области.

Используя передовые методы обработки естественного языка и машинного обучения, это исследование направлено на повышение эффективности и действенности автоматизированных систем маркировки задач. Результаты этого исследования могут потенциально способствовать разработке усовершенствованных методов классификации проблем, содействуя более эффективному поиску информации и приоритизации задач.

Abbreviations

ML Machine Learning

NLP Natural Languages Processing

HTML Hypertext Markup Language

TFIDF Term Frequency-Inverse Document Frequency

CSV Comma-Separated Values

RDF Resource Description Framework

KNN K-Nearest Neighbors

tf term frequency

idf inverse document frequency

RF Random Forest

RFC Random Forest Classifier

MNB Multinomial Naive Bayes

SVM Support Vector Machine

LR Logistic Regression

Table of Contents

Chapter 1

Introduction

In recent years, many people have become interested in programming, and in order to be expert in their field, they began to study competitive programming in depth. In this regard, more and more platforms related to programming and tutorials began to appear on the Internet. However, the problem is actually based on some algorithm(slide-window, dfs, bfs etc.) and for the people it is even hard to identify the strategy of solving problems. In most cases textual representation could not be very effective for people, and the system which will recommend strategies of solving such problems will be very useful, efficient and educational. And we focused on creating a machine with issues with automatic tags based on textual representation. But as our first steps are a little bit complicated we started with basic things such as identifying the difficulty of the problem. These targets can be either Easy, Medium or Hard.Each problem has it‘s individual difficulty, so it is a multi-class classification problem. An example of a data sample for our problem is shown on figure 1.1

1.1 Motivation

We want to talk about the process of labeling or tagging a specific text, specifically queries. There are now more user-generated questions than ever before thanks to the quick expansion of Q&A websites like Quora, Stack Overflow, and forums on different platforms. These queries could easily get lost in the large

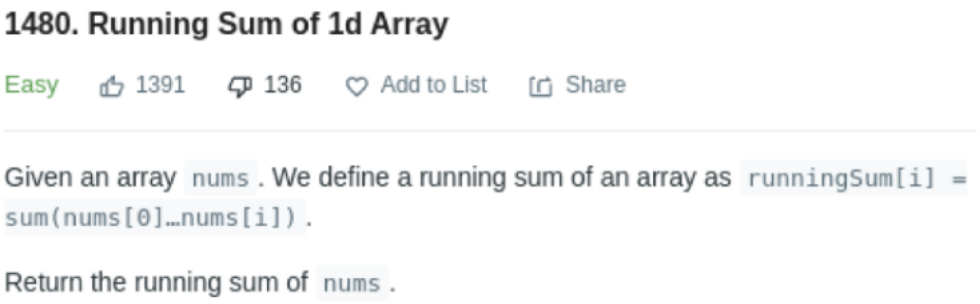

Figure 1.1: Leetcode problem.

sea of unanswered questions if they are not properly organized. Questions are grouped and given tags or labels to help alleviate this problem. Similar questions can be grouped together by using tags, improving the chances that they will be seen by the intended target audience and receive insightful responses.While employing staff to perform manual tagging is one possibility, this strategy has a number of serious disadvantages. First of all, because Q&A websites are so massive, it is almost impossible for staff to personally categorize every question. Additionally, human biases might affect manual tagging and result in inaccurate classification. A more reliable system is needed since manual tagging is labor- and error-intensive. Automatic labeling is useful in this situation. In a system that assigns tags automatically, questions are fed into a model. Due to its interconnectedness, the internet has emerged as a crucial global resource for billions of people. Our comfort and productivity can be considerably improved by tagging technology in a number of ways. However, the most widely used techniques mostly rely on time-consuming, expensive manual or semi-human labeling. We suggest an automatic tagging technique based on Natural Language Processing (NLP) to overcome these constraints. By automating the tagging process, this method avoids the aforementioned drawbacks of human tagging and high expenses while allowing users to access the information they need quickly.

1.2 Literature review

A variety of computer techniques are used in natural language processing (NLP), which is based on theoretical ideas and aims to automatically analyze and represent human discourse. From the days of punch cards and batch processing,

when sentence analysis may take several minutes, to the current era dominated by behemoths like Google, which can analyse millions of webpages in under a second, NLP research has improved throughout time. In order to offer new light on the past, present, and future of NLP research, this review paper examines current achievements in the field. This survey article reframes the development of NLP research as the convergence of three interrelated curves: Syntax, Semantics, and Pragmatics, drawing inspiration from the idea of "jumping curves" in business management and marketing prediction. In the end, it is anticipated that this convergence would push NLP research in the direction of obtaining natural language understanding.

There were only a few dozen exabytes of data on the web between the invention of the Internet and 2003, the year social networks like MySpace, Delicious, LinkedIn, and Facebook were launched. Today, though, a comparable amount of information is produced every week. As the Social Web has grown, people now have access to new platforms that make it easy and affordable to share content, concepts, and opinions with millions of other internet users. However, because this vast amount of information is primarily produced for human consumption and is largely unstructured, it is difficult for machines to handle it directly. Automated text analysis necessitates a deep comprehension of natural language in machines, which we have yet to attain. Here on figure 1.2 shown NLP sample.

Figure 1.2: brand x pictures.

Online information retrieval, aggregation, and processing up until this point have primarily relied on formulas based on the textual representation of web pages. Such algorithms are excellent at obtaining texts, decomposing them into smaller

chunks, examining the spelling, and tallying the quantity of words. However, their abilities are considered to be relatively constrained when it comes to parsing phrases and retrieving significant information. High-level symbolic abilities are in fact needed for natural language processing (NLP) (Dyer, 1994), including:

1. development and spread of dynamic bindings;

2. manipulation of constituent structures that are recursive;

3. acquiring and using episodic, semantic, and lexical memories;

4. management of numerous learning/processing modules and information flow between such modules;

5. the connection between perceptual/motor experiences and basic language constructs (such as objects and actions);

6. illustration of abstract ideas.

A broad range of abilities are needed to move from simple Natural Language Processing (NLP) to what is widely referred to as natural language understanding (Allen, 1987). Currently, most available methods use algorithms based on word co-occurrence frequencies and mostly rely on the syntactic representation of text. These algorithms, however, are constrained since they can only process data that is immediately available to them. Humans, on the other hand, effortlessly trigger a cascade of semantically related thoughts, pertinent experiences, and sensory perceptions with each word we come across when reading literature. This skill enables us to quickly and effectively carry out challenging NLP tasks like semantic role labeling, textual entailment, and word-sense disambiguation.

Computational models simulate how the human brain interprets natural language in an effort to close the cognitive gap. They accomplish this by making use of semantic elements that might not be conveyed clearly in the text. Computational models are useful in both science and practice. They are useful for scientific investigation because they let scientists look into the nature of linguistic communication. At the same time, they accomplish useful tasks by facilitating efficient human-machine connection.

The instruments required by conventional research fields to fully address the complexities of language comprehension and production are lacking. It is difficult to study using only traditional methods because of the problem's extensive nature. However, one method is to create computer programs that represent intricate theories, then assess their effectiveness by looking at their advantages and disadvantages. The shortcomings of these computational models can be found, and small advancements can be achieved. Through this repeated process, computational models are able to make precise predictions about human behavior, which psycholinguists can then further investigate. We may eventually obtain a deeper knowledge of human language processing through this joint project including forward-thinking experts from fields like psycholinguistics, neuroscience, anthropology, philosophy, and computer science.

This review paper adopts a different strategy from earlier surveys that focused on certain NLP research topics or applications, such as assessment standards, knowledge-based systems, text retrieval, and connectionist models. Through three different paradigms—the bag-of-words model, the bag-of-concepts model, and the bag-of-narratives model—it focuses on the development of NLP research. This survey essay investigates the gradual shift of NLP research from lexical semantics to compositional semantics by borrowing the notion of "jumping curves" from the discipline of business management. Additionally, it offers insights into the developing field of narrative-based NLP technology, providing a sneak preview of potential future developments in the area.

1.3 Related works

According to the definition provided by the Oxford English Dictionary, personality encompasses a combination of qualities, traits, and characteristics that distinguish individuals from one another. In simpler terms, personality represents the unique expression of an individual's self. Consequently, the focus of this study is both comprehensive and ambitious, as it aims to explore a wide range of factors contributing to human characteristics, thereby revealing various possibilities for action and enhancing our understanding of them. To ensure its validity from

a psychological standpoint, it was imperative to establish the relevance of this study. Although numerous studies, articles, and scientific research exist on this subject, the preliminary validation was crucial in expanding the interpretation of the proposed research techniques. It also reaffirmed the tasks assigned to support the study and contributed to the overall knowledge gained from the project.

Conducted an evaluation to differentiate between inference and traditional assessments or methods used to predict personality types in [1]. The assessment revealed several noteworthy aspects:

1. Inference methods exhibited a certain level of discrepancy or "distance" from accurately predicting real personality traits, resulting in errors.

2. One point of concern was the lack of previous hypotheses or established theories guiding the inference process.

3. Another issue identified was the challenge of retaining observed patterns for generalization purposes.

4. The success and progress of machine learning approaches in predicting personality were found to be relatively low.

5. A significant discrepancy in success rates was observed between training, validation, and underfitting stages, suggesting the need for improvement in model generalizability.

Overall, these findings highlight key areas of attention when considering the differentiation between inference-based approaches and traditional methods for personality prediction.

The existing literature supports two main biases: the psychological bias and the technical bias related to machine learning or intelligent systems. From a psychological perspective, the literature predominantly contributes to establishing a solid foundation for this study by comparing traditional methods of evaluating human personality types with automated methods. It also explores innovative and relevant approaches, such as those discussed [2], [3], and [4].

Regarding the field of Artificial Intelligence (AI) and its resources, this research

sheds light on feature selection, models, and even methods like self-attention-based feature selection. It also delves deeper into the selection of discriminative and generative models. The literature does not favor or endorse one specific approach over the other in relation to the addressed study field. Instead, it contributes to the exploration of various model selection techniques, optimization of hyperparameters, and their practicality in the realm of psychology.

The related work indicates that the performance difference between approaches is generally insignificant. Moreover, the quality of the data used is rarely considered as the focal point of the experiments. Overall, the evaluation metrics for the models tend to align with expected standards and consistency for classification problems.

1.4 Statement of Results

In chapter 4 we will show our results. We will consider which method efficiently in automatic tagging of tasks. In figure 1.1 given leetcode problem, so this is our important data. The present study delved into the domain of multi-class text classification, specifically examining the performance of machine learning approaches. Our findings emphasize the existing gaps in the capabilities of machine learning techniques and underscore the need for further advancements in the form of deep learning models. The purpose of this Thesis is to address the challenge of automating the tagging process for algorithmic problems. Specifically, our focus lies in developing a machine learning-based approach for accurately assigning difficulty tags to problem descriptions. While acknowledging the limitations of existing machine learning methods, we consider them as a foundational step for future investigations in this domain.

Chapter 2

Background

2.1 Natural Language Processing

NLP research has been focused on a variety of tasks since its inception in the 1950s, including machine translation, information retrieval, text summarization, question-answering, information extraction, topic modeling, and, more recently, opinion mining. Syntax was the subject of a lot of research in the early days of NLP. In addition to being inspired by the prevalent idea of syntax-driven processing, which gained implicit or explicit support from academics in the field, this emphasis on syntax was prompted by the obvious requirement for syntactic processing in language interpretation.

Although the relevance of semantics in NLP was clear from the start, the research community first concentrated on solving syntax because machine learning techniques were more directly applicable in that domain. However, some researchers understood the importance of semantics and thought it was a more difficult challenge, or they thought a semantically-driven strategy would be more successful. Using semantic categories and semantic case frames, for instance, groups led by Masterman and Ceccato investigated semantic pattern matching. Ceccato, in particular, used semantic networks to represent knowledge and world information to improve linguistic semantics.

As stressed by Minsky in 1968, other works recognized the requirement of incor-

porating outside knowledge to interpret and react to linguistic input. In Schank's 1975 work, the function of semantics was clearly emphasized, with a focus on general-purpose semantics that included case structures for representation and semantically motivated processing. These changes signaled an increasing understanding of the value of semantics and the application of semantic knowledge in furthering NLP research.

First-order logic (FOL), a logical system made up of axioms and rules of inference, has since become one of the most extensively used representation systems in NLP. Predicates with complex relational structures and supporting syntax, semantics, and, to a lesser extent, pragmatic expressions, are successfully formalized using FOL. The correct placement of symbol groups is determined by syntax, assuring their well-formedness. Well-formed expressions' meanings are defined by semantics. Contrarily, pragmatics makes use of contextual information to forge stronger links between various semantic components, which is essential for tasks like word sense disambiguation.

However, monotonicity is a problem for logic-based methods. The number of entailed sentences grows as more knowledge is added to the knowledge base, which runs counter to a common trait of human reasoning—the capacity for free and flexible change of opinion. Solutions like linear logic and default logic have been suggested as ways to solve this problem. Raymond Reiter developed default logic, which seeks to formalize presumptions such "all birds fly" . However, issues arise when truths that are generally true but are exceptions to the general rules, such "penguins do not fly," are formalized using default logic.

The production rule, which Chomsky established in 1956, is another frequently used model for defining natural language. There is a working memory that stores ongoing memory claims in a production rule system. These production rules are stored in a volatile working memory. Each production rule is expressed as "IF (conditions) THEN (actions)", with each rule consisting of an antecedent collection of conditions and a consequent set of actions.

A production rule system's basic operation consists of a cycle of three steps: "recognize," "resolve conflict," and "act." Until no more rules are applicable to

working memory, this cycle continues. The system recognizes the rules whose antecedent criteria are satisfied by the working memory at hand during the "recognize" step. The conflict set is the name given to these rules. The system chooses an appropriate subset of rules from the conflict set to be applied during the "resolve conflict" stage. The system conducts the chosen actions, which may include changing the working memory, in the final "act" stage.

The modularity of production rule systems is one of their benefits. The system can be managed and maintained more easily by grouping the production rules into modules.

Each rule in a production rule system runs independently of the others, making it simple to add and remove rules. Production rule systems' modular design makes them flexible and adaptable. Production rule systems have a straightforward control structure, and the rules themselves are typically simple for humans to understand. This is due to the fact that the rules are frequently derived from expert conduct or knowledge, and the vocabulary used to encode the rules is intended to be intuitive to humans.

However, as the size of the production rule system expands, scalability may become a problem. A system with thousands of rules requires a lot of maintenance work to manage. It gets increasingly difficult to ensure the appropriate operation and coordination of rules as the number of rules rises. As a result, maintaining a complex production rule system can be time- and resource-consuming.

The Ontology Web Language (OWL), which McGuinness and Van Harmelen introduced in 2004 [5], is another noteworthy NLP approach. The Resource Description Framework (RDF) is an XML-based vocabulary that OWL enhances to offer a more complete set of capabilities for ontology representation. It makes it possible to define classes, relationships between classes, class properties, and restrictions on those relationships and properties.

The subject-predicate-object model is supported by RDF, the foundation of OWL, when making assertions about resources. RDF-based reasoning engines have been created to improve ontology classification and guarantee semantic consistency. Particularly OWL focuses on precisely describing static structures, mak-

ing it appropriate for representing declarative information. OWL has restrictions, though.

First off, OWL demands rigorous definitions, which makes it unsuitable for describing information with varying degrees of confidence. It works better to convey factual or objective knowledge. It is also challenging for OWL to express knowledge that is temporally dependent, making it challenging to capture data that is dynamic or has temporal dependencies.

These restrictions show that although OWL is an effective tool for capturing some forms of information, it may not always be the best option, especially when dealing with subjective or temporal aspects.

Network-based methods are in fact frequently employed in NLP, with Bayesian networks (sometimes referred to as belief networks) serving as a notable illustration. A framework for expressing joint probability distributions over connected hypotheses is provided by Bayesian networks, which were first described by Pearl in 1985. A directed acyclic graph (DAG), in which arcs reflect causal relationships between variables, is used to represent the variables in a Bayesian network.

Subjective levels of confidence can be represented using Bayesian networks. To determine the likelihood of events, they expressly take prior knowledge into account and pool the available evidence. It is necessary to understand the conditional probability Pr(P|parents(P)) for each variable P in order to compute the joint distribution of a belief network. However, figuring out these probabilities for each variable can be difficult, particularly for activities involving a lot of data processing. The statistical tables for such networks might be difficult to maintain and update.

Additionally, the expressiveness of Bayesian networks is constrained, comparable to that of propositional logic. Semantic networks are hence frequently chosen in NLP research. Semantic networks give you more options for expressing and capturing intricate connections between concepts and entities. They offer a graphic representation that makes knowledge and meaning easier to understand and more richly represented.

Developed by Sowa in 1987 [6], a semantic network is a graphical representation that shows knowledge as interconnected nodes and arcs. Semantic networks come in a variety of forms, such as assertional and definitional networks.

The "IsA" interactions between concepts and their subtypes are the main focus of definitional networks. These connections create generalizations that enable all of the subtypes of a supertype to inherit the properties specified for the supertype. The information is often taken as true in definitional networks.

On the other hand, assertional networks are used to make claims, and the data they contain is thought to be partially true. In assertional networks, contingent truth is more often determined by common sense than by default logic. In assertional networks, the propositions are supported by sufficient arguments, where the arguments support the proposition. For instance, "the sun is shining on the stone" and "whatever the sun shines on is warm" could both be adequate arguments for the assertion "the stone is warm."

Semantic networks give knowledge an organized and illustrative representation, making it easier to identify relationships and to reason about the data they include.

In the early 1960s, Simmons and Quillian's [7] research gave rise to the idea of semantic networks. In the late 1980s, Marvin Minsky expanded on this concept in his Society of Mind theory. According to Minsky, varied agents working together to do various cognitive tasks including remembering, comparing, generalizing, analogizing, and anticipating lead to the development of human intelligence. The artificial intelligence community was enthused by this idea of human cognition, which inspired the development of practical knowledge bases for NLP tasks.

A number of well-known initiatives, such as Cyc, WordNet, Thought-Treasure, and the Open Mind Common Sense project, were inspired by Minsky's notion. Doug Lenat created Cyc, a logic-based database of common sense information. WordNet is a global database of word senses that was developed by Christiane Fellbaum [8]. A system for interpreting stories was created by Erik Mueller and is called Thought-Treasure [9]. The Open Mind Common Sense project was a second-generation common-sense database where information is presented in nat-

ural language and provided by online volunteers rather than being manually created by skilled technologists.

A substantial quantity of common sense knowledge has been gathered as part of the Open Mind Common Sense project, and is now being applied to a variety of NLP tasks, including textual affect sensing, interpreting casual conversations, opinion mining, narrative telling, and others. By incorporating and utilizing common sense information in natural language processing tasks, these efforts have advanced NLP research.

Their technological specialties. It is true that the pace of progress in NLP has not always kept up with the quick changes in the digital landscape, even if there have been substantial achievements in NLP research, such as the creation of numerous models and techniques as previously described.

Natural language understanding's intricacy is one cause of this discrepancy. Due to the complexity and ambiguity of language, it is difficult for robots to effectively understand and produce writing that is human-like. In addition, when languages change over time, new slang, idioms, and cultural references are added, making NLP jobs much more challenging.

The enormous quantity of data and computing power needed for developing and deploying complex NLP models is another aspect. It gets more and harder to comprehend and interpret all the information accessible as the number of digital stuff expands exponentially.

It is crucial to remember that NLP research has advanced significantly in recent years. Machine translation, sentiment analysis, question answering, and text synthesis have all seen significant improvements thanks to the development of deep learning techniques, which were made possible by technology improvements and the availability of enormous datasets.

Additionally, combining NLP with other cutting-edge technologies like reinforcement learning, graph neural networks, and knowledge graphs has the potential to overcome some issues and advance the field of NLP research.

It is anticipated that NLP research will continue to grow at an accelerated rate

as we observe the ongoing technological evolution and the rising demand for natural language processing capabilities. The future of human-computer interaction and intelligent systems depends on researchers and practitioners actively investigating novel ideas and methodologies to enhance language understanding and generation.

It is true that a significant trend in the area is the move away from word-based approaches toward a more consistent exploitation of semantics in NLP systems. Although words are the basic building blocks of language, they frequently lack the depth and context necessary for proper meaning representation and deeper understanding.

NLP systems can capture a deeper grasp of language by focusing on multi-word expressions and the accompanying semantics and sentac. Idioms, collocations, and named entities are examples of multi-word formulations that have distinct meanings and nuances that are difficult for single words to adequately convey. NLP systems can comprehend language more complexly by taking into account the compositional character of language, in which words are combined to create new meanings.

Sentic and semantic integration is essential for decision-making and common sense in NLP. information the world, its people, events, and relationships, as well as being able to use logic and judgment to base judgments on this information, are all components of common sense knowledge. Sentics, which includes emotive information, gives language an additional layer of meaning by allowing systems to understand subtle emotional cues, individual viewpoints, and sentiment analysis.

Deep learning models, knowledge graphs, and other improvements in semantic representations have given NLP systems new ways to capture and use semantic data more efficiently. These methods make it possible to describe links, ideas, and hierarchies, which improves comprehension and inference abilities.

NLP research strives to advance the precision, contextual comprehension, and reasoning capacities of NLP systems along the "Semantics Curve," ultimately enabling more natural and intelligent interactions between humans and machines. Here on figure 2.1 for us given envisioned evolution of NLP research through three

different eras or curves.

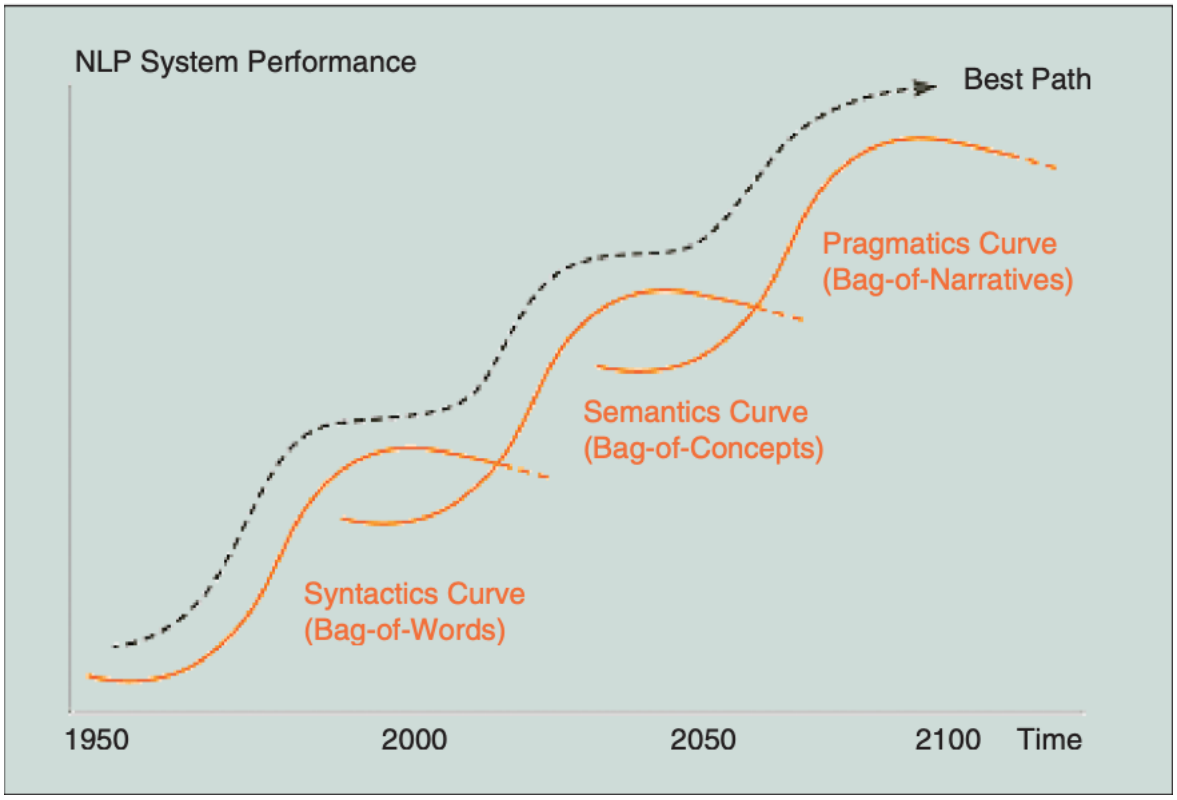

Figure 2.1: Envisioned evolution of NLP research through three different eras or curves.

2.2 Automatic tagging of text

The proliferation of user-generated content on the internet has led to the development of tagging systems that aim to annotate these data together metadata, typically in a like a form with passwords. These tagging systems serve the purpose of organizing, browsing, and searching the vast amount of content available. Various types of tagging systems have emerged, such as image tagging on platforms like Flickr, web page tagging on platforms like Del.ico.com, and tagging of social forms on platforms like Facebook. These systems have gained popularity and are widely used across the web.

Different tagging schemes are designed for various kinds of information. Multiple people can tag resources using social and collaborative tagging systems, allowing them to be shared throughout a group or community. On the other

hand, certain systems, like those used by sites like YouTube, simply let the content owner define the tags connected to their content. Our technology is designed to handle the latter.

Many individuals think that the power of tags comes from their capacity to let users freely select relevant tags for resources without being constrained by established lexicons or hierarchies. Without relying on established tag corpora, the dynamic nature of tagging systems allows content authors to easily adapt to new ideas and terminological changes. On the other hand, some contend that big tag of user generations collections, or folksonomies, will eventually converge on a common lingo that helps users find and browse material. The collaborative process that went into developing this vocabulary system allowed for individual contributors to naturally pick up new terminology and grow the domain language.

Unfortunately, tagging systems have the same difficulties as conventional free text information retrieval systems because they do not enforce set or restricted vocabularies for tag selection. Polysemy, synonymy, and level variation were cited as the three main issues with current tagging techniques by Golder et al [10] .

When a single tag in a tagging system has numerous meanings, this is known as polysemy. For instance, a blog entry with the tag "caterpillar" might discuss etymology or discuss building machinery.

Multiple tags that have the same meaning are referred to as synonyms. This may result from semantic similarities (such as "news" versus "current events") or morphological variation (such as "blog" against "blogs"). In blog post tagging systems, synonymy presents a special problem because authors must use their own discretion to choose an acceptable tag for their content and there is no guarantee that two users writing about the same subject would use the same tag.

Level variation, which happens when users tag information at various levels of abstraction, is the third issue that has been identified. Depending on the blogger's knowledge or needs, tags can be applied to content at different levels of generality or specificity. In order to add more specific information, an enthusiast can choose to tag a post with "Volkswagen MKIV 2001 Golf" as opposed to the more general "car" tag.

The lack of obvious functional motivations to assure consistent, reliable, and thorough tagging presents an additional issue for current blog tagging systems. When these tags are utilized in applications for collaboration, community involvement, clustering, and search, this becomes a problem.

Some writers use tags in their writings to increase their visibility to a wider audience. They use general category descriptions like "politics" or "shopping." On the other side, some bloggers use tags mainly for their own interpretation and organizing, including descriptive tags that have nothing to do with the content, like "random" or "toRead." Some people choose to use tags that are extremely detailed or specialized, such as "why I hate Best Buy" or "instructions for making grandma's apple pie," to accurately reflect the substance of their postings. It is difficult to develop uniform and standardized tagging norms across the blogging community due to the diversity of tagging techniques. Here on figure 2.2 shown the distribution of tags and their frequencies.

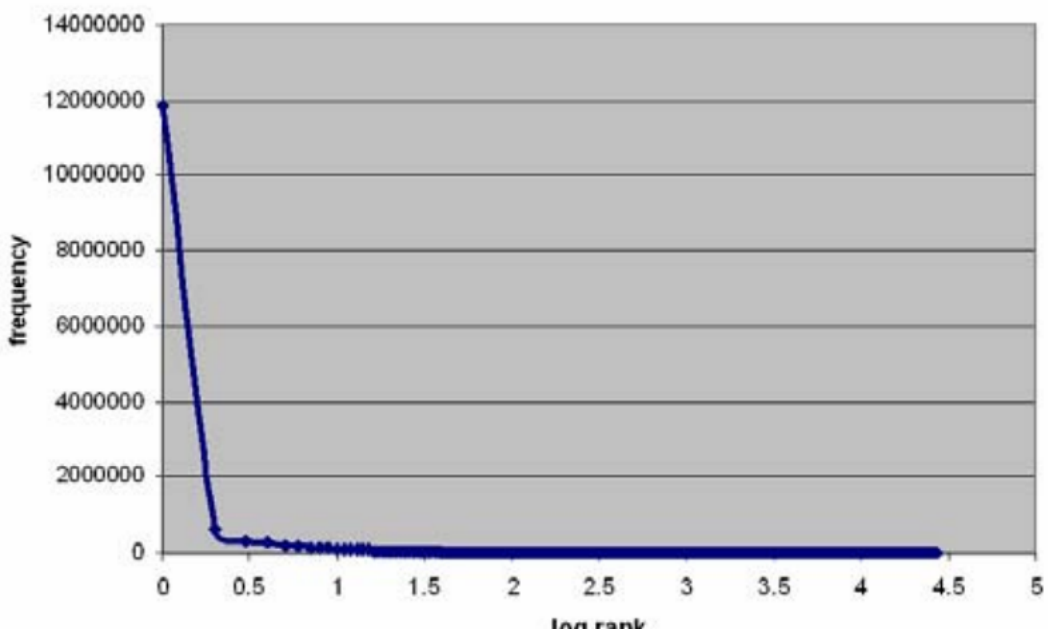

Figure 2.2: The distribution of tags and their frequencies.

The lack of a standardized or shared language has caused the blogosphere to experience a significant overpopulation of individual tags. As of a recent update, approximately 60 million blogs and around 11.5 million tags were being tracked by Technorati, one of the most well-known blog aggregators [11]. A sample of English blog data from a 16-day period in late 2006 was examined by Technorati, and it

was discovered that there were roughly 403,000 different tags. These tags had a median frequency of 8 and a mode of 1, with a mean frequency of 343.1. The tag "Weblog," which was used 6,695,762 times, was the most commonly used. Only 5.7% (88,212) of the tags happened more frequently than the average frequency (343.1), whereas about 22% (88,212) of the tags only appeared once. On figure 2.2 illustrates a sample distribution of tags and their frequencies, with the tags ordered by frequency and sampled every 15 tags using a logarithmic distribution.

The statistics unambiguously show that, while a major fraction of tags are used infrequently, following a "long tail" distribution, only a small percentage of tags are reused by bloggers [12] . Practically, due to their distinctive nature, highly infrequent tags, commonly referred to as "meta-noise," are unlikely to be beneficial for retrieval. For instance, it is extremely improbable that users will use the tags we will consider in chapter 4, which is a small random sample of tags appearing just once in our data, to browse or search for content.

Tagging procedures used by content providers' own tagging systems, particularly those without a collaborative component, are frequently uneven and peculiar. People do tag their posts, so it's not like they don't care to do it. The tagging effort, however, becomes difficult without knowledge of how other bloggers identify their content, leading to less relevant results for retrieval and browsing reasons. Therefore, systems that can propose relevant tags for content must be developed.

The goal of such a system is to enable users to see how other users are categorizing their information so that they can utilize the common terminology or add new tags as needed. The ultimate objective is to improve tagging system performance without compromising the strength or potential of folksonomies.

Our goal was to develop a system that helps users tag their own blog entries by offering a relevant set of tags, taking into account the condition of the tag space. We use a mediated recommendation system technique, in which the system recommends a set of pertinent tags but does not automatically apply them. Instead, the user can pick the tags from the suggested set that they think are appropriate. The chosen tags are then included in a bigger collection of associations between posts and tags after being added to the post. The system also provides a text box

where users can enter extra tags that weren't already suggested, allowing for the introduction and use of brand-new and developing tags.

This strategy is intriguing because it makes use of extensive data processing while requiring little human involvement in the suggestion-verification process. Additionally, it radically changes the tagging process, shifting it from generation to recognition and saving time and cognitive effort [13] [14].

Additionally, we give the tag space the chance to spontaneously converge on a consensus regarding tag choice by consistently offering suggestions to users. The users would gain insight into the kinds of tags used by other bloggers to describe related content, which would assist address issues like synonymy and level variation. By lowering the amount of idiosyncratic tags and lowering the meta-noise in the tag distribution, convergence would also improve recall.

In fact, a number of systems have been created to automatically generate relevant tags for blog entries. These tools are intended to help bloggers with the tagging procedure and increase the regularity and usefulness of tags. These systems scan blog post content using a variety of methods and algorithms, and then they provide pertinent tags that appropriately reflect the post's subjects or themes.

Systems for automatic tag generation use a variety of techniques, such as information retrieval methods, machine learning, and natural language processing (NLP). Using word frequency, semantic relationships, and context, NLP algorithms extract significant keywords and phrases from blog posts. Machine learning models are able to anticipate the right tags for new blog posts by learning patterns and correlations between content and tags from big datasets. By comparing a blog post's content to previously published posts and the tags assigned to them, information retrieval techniques can use similarity measures to recommend appropriate tags based on the content's resemblance to existing posts.

By providing bloggers with tag options, these automated tag creation systems lessen the mental strain involved in manual labeling. These automated systems produce consistent tags that could perhaps be more precise, increasing the overall effectiveness of the tagging system. It's important to note that the performance and accuracy of automatic tag generation systems can vary depending on the

specific algorithms, training data, and the nature of the blog content. Nonetheless, these systems represent valuable tools for bloggers seeking assistance in the tagging process and optimizing their content organization and discoverability.

One of the earliest programs created for automatic tag generation was TagIt [11]. It used Naive Bayes text classification techniques to decide what tag would be best for a brand-new blog article. Although the system produced encouraging results, it was not clear whether it could scale beyond the 330 tags in the training set or whether it could be evaluated against blog articles with multiple tags.

Using TFIDF (Term Frequency-Inverse Document Frequency) scoring, another system created by Brooks et al. [15] was able to automatically tag blog articles based on the top three phrases that were extracted from the post. This strategy produced clusters of posts that were more narrowly targeted, but it only gave unigram tags that actually existed in the blog post.

A system similar to ours is the collaborative filtering AutoTag system developed by Gilad Mishne [16] . AutoTag identifies similar tagged posts and suggests a set of associated tags to the user for selection. While our system employs a similar technique, we have improved its performance by introducing tag compression and case evaluation to filter and rank tag suggestions.

The collaborative filtering AutoTag system created by Gilad Mishne [16] is a system that is comparable to ours. AutoTag finds similar tagged content and offers the user a list of related tags to choose from. Despite using a similar methodology, our system performs better because to the addition of tag compression and case evaluation to sort and rank tag suggestions

These systems demonstrate many methods for automatically creating tags, each with advantages and disadvantages. The usage of collaborative filtering, TFIDF scoring, and machine learning algorithms highlights the variety of approaches taken to solve the problem of coming up with meaningful tags for blog entries.

We created a functional framework for blog tags to specify the task and direct the creation of our system. Our main objective was to make it possible for users to browse and retrieve posts based on their contextual ties to tags or groups of

tags. Currently, the tag space is used for retrieval and browsing, but due to its inconsistent use, a considerable but under-noticed component of the tagged blogosphere known as the "long tail" exists.

In order to solve this problem, we created a recommendation system that makes use of tags that have already been applied to content in order to suggest tags for new content. The system takes a post that isn't tagged, finds posts that are tagged and similar to it, gathers the tags that go with those posts, and then suggests a set of tags to the user. The number of times a tag appears in a prior post is one of many variables that affect the tags that are suggested. Instead of treating each unique tag as a separate symbol in order to increase the efficiency of our system, we searched for possibilities to automatically group tags that were morphologically related utilizing lossless compression methods. The system can use a higher number of tagged posts to provide recommendations by finding comparable groups of tags.

Our technology provides for the continued updating of content within the training system in order to react to the constant flow of fresh blog content and keep the data from getting out-of-date. The system can adapt to modifications in the blogosphere, such as the inclusion of new tags and changes in the meanings of current tags, thanks to its adaptability.

Chapter 3

Methodology

3.1 Future engineering

Natural Language Processing (NLP) refers to the field of computer science that deals with the automated processing of human languages. It encompasses algorithms that can analyze and understand human-generated text as input, as well as algorithms that can generate text that resembles natural language as output.

In this chapter, we present our approach for an autonomous tagging system. The data used to evaluate our method was sourced from "kaggle.com" and [17]. The following steps provide a detailed description of our method. The accompanying flowchart provides a concise overview of our proposed approach. On figure 3.1 clearly shown processing steps.

Preprocessing Steps:

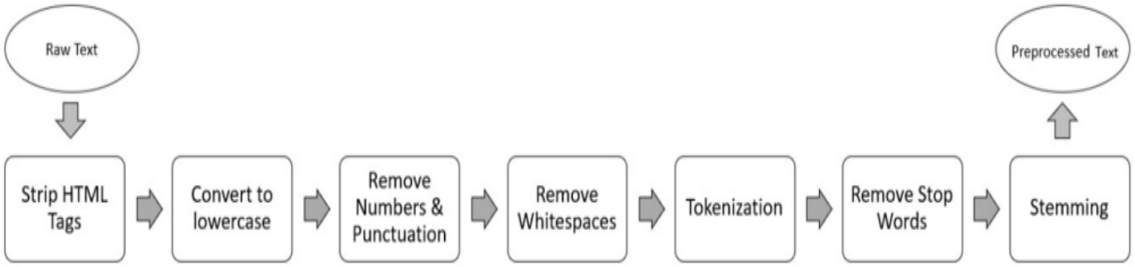

Figure 3.1: Preprocessing Steps.

3.1.1 Data Preprocessing

Certainly! Preprocessing of data is an essential step before training a model. Here are the different stages of text preprocessing typically performed to prepare the data for feature extraction and training:

1. Tokenization: The text is divided into individual tokens, such as words or subwords. This process helps break down the text into meaningful units.

2. Cleaning and normalization: This stage involves removing unwanted characters, such as punctuation marks or special symbols, and converting the text to a consistent format. It may also include steps like converting text to lowercase and handling contractions or abbreviations.

3. Stopword removal: Stopwords are commonly used words (e.g., "and," "the," "is") that do not carry significant meaning. Removing stopwords can reduce noise and improve processing efficiency.

4. Lemmatization or stemming: Lemmatization aims to reduce words to their base or root form (lemma), while stemming involves reducing words to their base form by removing prefixes or suffixes. These techniques help consolidate variations of words and improve generalization.

5. Handling numerical values or special entities: If the text contains numbers or special entities like URLs or email addresses, they can be replaced or treated in a specific way to avoid bias or unnecessary complexity.

6. Removing irrelevant or rare words: Words that occur very infrequently in the dataset may not contribute significantly to the model's learning. Removing these rare words can reduce noise and improve performance.

7. Handling spelling errors or typos: Depending on the quality of the data, it may be necessary to correct spelling errors or typos to ensure consistency and improve accuracy.

8. Text normalization: This step involves applying additional techniques such as removing excessive whitespace, handling repeated characters, or converting accented characters to their base form.

These preprocessing stages help clean and transform the text data into a format that is more suitable for feature extraction and training machine learning models.

3.1.2 Feature Extraction

The process of feature extraction is a crucial step in building an automatic tagging system. In this context, the feature extraction takes place after the data preprocessing step. Here's an overview of the feature extraction process described in your statement:

Tokenization: The questions in the database are converted into individual words, also known as tokenization. This step breaks down the questions into meaningful units.

Stopword removal: Insignificant words, such as "in," "as," "the," "and," "a," and "an," are removed as they do not carry much information and are commonly used in the English language.

Stemming: The Porter Stemmer algorithm is used to reduce words to their base form, known as stemming. This step helps consolidate variations of words and improve generalization.

Extracting top hundred tags: The top hundred tags, based on their frequency of occurrence in the database, are selected. These tags are the most frequently appearing ones in the Stack Overflow questions.

Selecting positive and negative examples: For each of the hundred tags, a balanced dataset is created by selecting a thousand positive examples (questions with the respective tag) and eight hundred negative examples (questions without the respective tag). This balance is important to ensure the SVM algorithm's performance, as it tends to perform poorly on unbalanced data.

Extracting frequent words: From the thousand positive examples for each tag, the top hundred frequently occurring words in the questions are extracted. These words serve as features for the respective tag. Dimensionality of the feature vector: As there may be overlapping words among the hundred tags, the worst-case scenario is that the dimension of the feature vector will be (n * 10,000), where

n is the number of data rows used to train the model. This dimension reflects the number of distinct words across all tags.

By following this feature extraction process, a feature vector is constructed for each question, representing the occurrence of the top hundred frequent words associated with the respective tags.

3.1.3 Prediction Models

In your project, you utilized classifiers from the scikit-learn library in Python, specifically the k-Nearest Neighbors (KNN) and Random Forest (RF) algorithms, for the classification task on your dataset. Let's focus on the KNN algorithm.

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm primarily used for classification tasks. It works based on the assumption that data points with similar features are close to each other in the feature space and form clusters [18]. When classifying a new data point, KNN finds its 'k' nearest neighbors, where 'k' is a parameter set for the model . The new data point is assigned the class label that appears most frequently among those 'k' nearest neighbors. For instance, if k=5 and the 5 nearest neighbors for a data point are labeled 0, 0, 1, 0, and 1, the class label 0 will be assigned to the point.

To measure the similarity or "closeness" between two data points, the KNN algorithm utilizes a distance measure in the feature space. The most common distance measure used is the Euclidean distance, which is calculated in a multi-dimensional space, where each feature acts as a dimension. In your dataset [19], the frequently occurring words for each tag were used as features. Consequently, similarly classified questions tend to have similar feature values, as frequently appearing words are more likely to appear in positive examples and less likely to appear in negative examples.

For your application, you chose a parameter value of k=5 for the KNN model [20]. This value indicates that the algorithm considers the five nearest neighbors when classifying a new data point. Based on your results, KNN provided satisfactory performance and computational efficiency.

In Table-1 of your project, you presented the percentage accuracy of KNN on the top five tags (sorted by frequency) for both the train and test data. As anticipated, JavaScript emerged as the most frequent tag, followed by Java.

It's worth noting that this explanation focuses on the KNN algorithm specifically and provides an overview of how it functions in your project.

In addition to the K-Nearest Neighbors algorithm, you also utilized the Random Forest Classifier for your classification task. Here's an explanation of the Random Forest Classifier and its performance in your project:

The Random Forest Classifier is an ensemble machine learning method that leverages multiple decision trees to make predictions or classify data. In your project, you used the Random Forest implementation provided by Python's scikit-learn library [21], which efficiently builds the random forest for your dataset.

To optimize the performance of the Random Forest Classifier, you tuned the maximum depth parameter within a range of 3 to 8. After experimentation, you found that setting the maximum depth to 5 yielded the best results. This parameter determines the maximum depth of each decision tree in the random forest.

The Random Forest Classifier [22] achieved an accuracy of 72% in your classification task.

When building the random forest, you tested multiple hyperparameters and identified the following as the most promising:

n_estimators = 150: This hyperparameter determines the number of trees in the random forest. By setting it to 150, you specified that the forest should consist of 150 decision trees. max_depth = 5: This hyperparameter determines the maximum depth of each decision tree. A depth of 5 was found to provide optimal performance for your task. It's important to note that random forests go beyond simply averaging the results of individual decision trees. They employ two key concepts during the forest construction:

Random sampling of training dataset: Each decision tree in the random forest is built using a randomly selected subset of the training data. This random

sampling ensures diversity among the trees. Random feature selection for node splitting: When creating each node in a decision tree, a random subset of features is considered for determining the best split. This further enhances the diversity and robustness of the random forest. Overall, the Random Forest Classifier [23], with the selected hyperparameters, demonstrated a promising performance in your project with an accuracy of 72%.

3.2 System analysis

3.2.1 Existing system

The existing system involves users searching for information on the internet, but sometimes they struggle to find the desired information or face difficulties in locating specific content. The problem is exacerbated by the fact that the system doesn't provide alternative solutions to users. The current methods employed for tagging content are manual tagging and semi-automatic tagging.

In manual tagging, questions are tagged with relevant knowledge units by experts or specialists in the subject matter. However, this process can be time-consuming and costly due to the large number of questions in a question bank, which is constantly updated. It requires significant human effort to manually tag all the questions.

On the other hand, semi-automatic tagging involves analyzing the content and generating tags, which then require further processing by users. Human assistance is necessary in this process. However, there is a high probability of questions being mismatched with incorrect or irrelevant tags, leading to inaccurate or unreliable results.

Overall, the limitations of the existing system include the difficulty in finding desired information, the absence of alternative solutions, the time and cost associated with manual tagging, and the potential for mismatches in semi-automatic tagging. These challenges highlight the need for an improved system that can effectively address these issues and provide users with accurate and relevant information.

The problem with the existing system can be defined as follows:

1. Time-consuming and labor-intensive manual tagging: The process of manually tagging content requires a significant amount of time and effort from experts or specialists. This manual attention can be costly and may slow down the overall system.

2. Inefficiencies in semi-automatic tagging: The semi-automatic tagging approach, which involves analyzing content and generating tags, still requires further processing by users. This additional processing adds complexity and increases the overall time and effort needed to tag the content accurately.

3. Difficulty in providing requested user query questions: The system may struggle to effectively retrieve and present the requested user query questions. This can lead to a frustrating user experience and hinder the user's ability to find the desired information.

4. High cost and time requirements: Both manual tagging and semi-automatic tagging processes can be resource-intensive, resulting in high costs and time-consuming operations. These factors contribute to inefficiencies and hinder the overall productivity of the system.

In summary, the existing system suffers from the drawbacks of time-consuming manual tagging, the need for additional processing in semi-automatic tagging, limitations in providing relevant user query questions, and high costs and time requirements. Addressing these issues is crucial for improving the efficiency and effectiveness of the system.

The proposed system aims to address the limitations of the existing system by introducing an automatic tagging approach using Natural Language Processing (NLP) and machine learning techniques. The system leverages the power of NLP to analyze and understand human language in a smart and effective manner.

Instead of relying on manual or semi-automatic tagging methods [24], the proposed system automates the process of tagging questions. It overcomes the challenges of manual tagging by providing a more efficient and scalable solution for daily updates of knowledge. This ensures that the system can keep up with the

evolving information landscape and provide the most up-to-date and relevant answers to users.

The system utilizes medical datasets collected from the Medinet library, which are stored in CSV file formats. These datasets contain questions and answers from the medical health domain. Through automatic tagging, the system assigns appropriate tags to each question and answer pair.

When a user searches for a query, the system forwards the query to the Medinet library [25], which contains 700 medical domain files stored in the "tc2011" API. If the requested file is not found, the system redirects the query to PDF box, where relevant PDF files containing information about the domain or diseases related to the question are provided. Lucene indexing is used to extract and retrieve the PDF files efficiently.

In cases where the answer is not directly related to the query, the system forwards the query to an expert, such as a doctor, who can provide clarification and respond to the query. This ensures that users receive accurate and reliable information.

Additionally, the system offers quizzes for students who want to assess their knowledge in the medical field. It provides domain scores and overall scores to help students gauge their understanding. Feedback is also provided to assist students in improving their knowledge.

Advantages of the proposed system include:

Elimination of human involvement in the tagging process: The automatic tagging feature removes the need for manual assistance, saving time and resources. Standard and consistent results: The system ensures that tagging is performed consistently, leading to more reliable and standardized outcomes. Cost-effectiveness: The automation of tagging reduces the cost associated with manual or semi-automatic methods. Easy retrieval of tagged answers: The tagging process enhances the understanding of information, making it easier for users to retrieve the answers they are seeking. Provision of knowledge for students: The system supports learning by providing access to medical knowledge and offering quizzes

for students to test their understanding. Overall, the proposed system offers significant advantages over the existing system, providing an efficient, accurate, and cost-effective solution for tagging questions and delivering relevant information to users.

The system begins with the registration of the Admin, who provides their details (Name and password). Once a valid admin logs in, they gain access to the system. The Admin is responsible for various tasks such as data cleaning, CSV file management, resource analysis, and Natural Language Processing (NLP) cleaning.

Users need to register by providing their details, and upon successful login (using their Name and Password), they can enter the system. Users can select domain categories from the available options. Upon selecting a category, a list of related domains, along with information and question answers, is displayed. In case the requested information or question answer is unavailable, PDFs related to that domain are provided. If users are unsatisfied with the provided answers, they can ask questions to the designated Expert (doctor) who can provide clear answers.

Experts (doctors) need to register by providing their Name, Expert ID (a unique identifier), and their specific domain category. After logging in with valid credentials, experts receive notifications about user questions. They analyze the questions and provide relevant answers, which are then forwarded to the user's profile. Users can view the answers at any time.

Additionally, students can register and proceed to the quiz page upon login. They are presented with a medical domain quiz, where they answer questions related to the chosen domain. After completing the quiz, students receive an overall score, domain score, and feedback to help improve their knowledge in the chosen domain.

I hope this paraphrased version accurately captures the essence of the original description. Let me know if you need any further assistance!

Our system incorporates the following features:

Automatic Question Tagging: We introduce models that facilitate automatic question tagging. This feature is crucial for effective question management, despite being an area that has been largely overlooked. Our proposed mechanism efficiently captures vital information from questions, allowing for improved question text and short text mining.

Scalability: Our system is designed to handle large volumes of questions and effectively manage them. Whether it's a small number of questions or a massive database, our system can handle the scalability requirements.

Flexibility: The mechanism we propose for capturing information from questions is adaptable and can be extended to other tasks beyond question tagging. It can be applied to various text mining activities, enabling versatility and usability across different domains. Efficiency: Our models are designed to optimize the processing time for question tagging and related tasks. We prioritize efficiency to ensure quick and accurate results, enabling smooth user experience and seamless system performance.

Integration: Our system can be integrated with existing question management systems or utilized as a standalone solution. It offers compatibility and can seamlessly merge with other components or technologies already in use.

User-Friendly Interface: We emphasize a user-friendly interface, ensuring that users can easily interact with the system. Intuitive controls, clear navigation, and helpful prompts enhance the overall user experience.

Continuous Improvement: We are committed to ongoing enhancement and refinement of our system. As new techniques and approaches emerge, we strive to integrate them to provide the latest advancements in question management and text mining.

By incorporating these features, our system aims to address the limitations and gaps in question management while providing a robust and efficient solution for various text mining tasks.

On figure 3.2 shown overall system architecture.

The system utilizes the following modules:

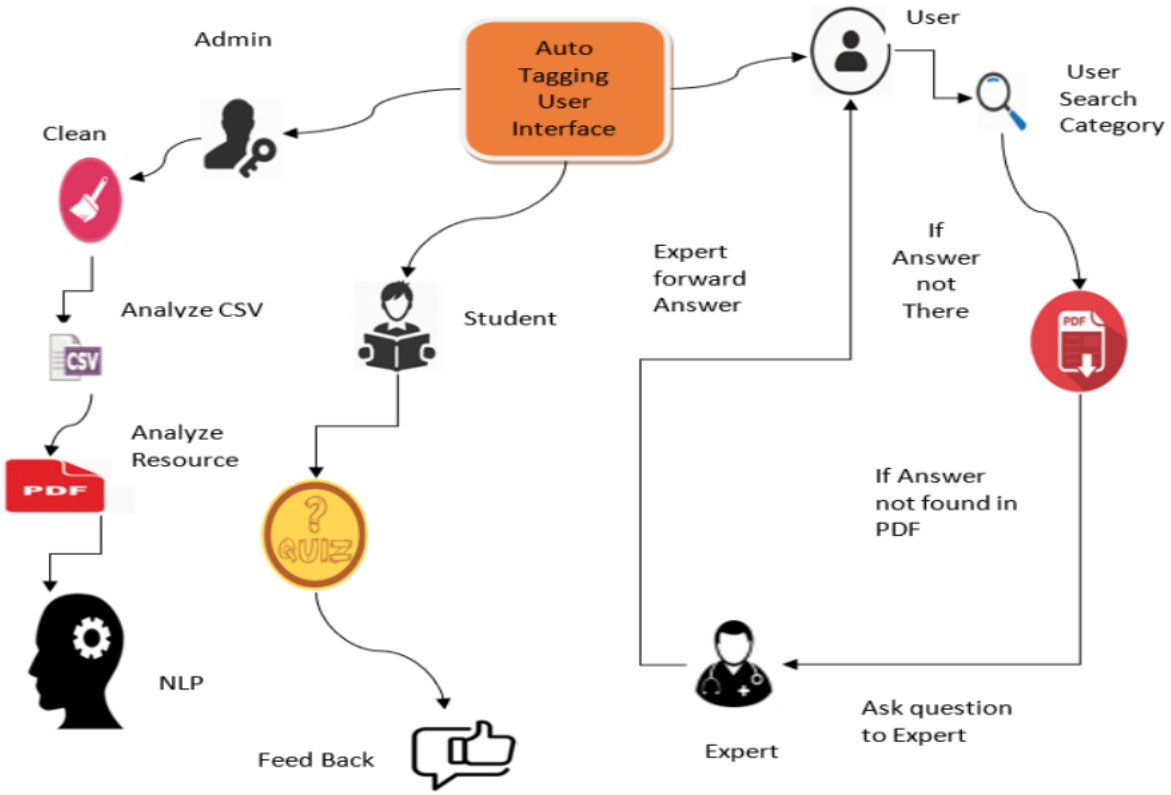

Figure 3.2: System Architecture.

Admin Preprocessing: This module focuses on the tasks and responsibilities of the system administrator. It involves activities such as data cleaning, CSV file management, analyzing resources, and cleaning Natural Language Processing (NLP) data. The admin ensures the smooth operation of the system.

Auto Tagging Questions: This module is responsible for automatically tagging questions with relevant keywords or categories. It employs machine learning or natural language processing techniques to analyze the question text and assign appropriate tags. Auto tagging helps in efficient question management and retrieval.

Expert's Answering Process: This module handles the interaction between users and experts (doctors). Experts register with their unique identification (Expert ID) and specific domain category. When a user asks a question, the system notifies the relevant expert. The expert analyzes the question and provides a comprehensive answer, which is then forwarded to the user's profile.

Student Assessment: This module focuses on assessing students' knowledge in

a specific domain, particularly in the medical field. Students register and access the quiz page upon login. They are presented with a set of questions related to the chosen domain. After completing the quiz, the system calculates the overall score, domain score, and provides feedback to help students improve their knowledge.

These modules work together to facilitate the smooth functioning of the system. The Admin Preprocessing module ensures data integrity and system maintenance. The Auto Tagging Questions module enhances question organization and retrieval. The Expert's Answering Process module connects users with domain experts for accurate answers. Finally, the Student Assessment module assesses students' knowledge through quizzes, providing scores and feedback for improvement.

Sure! Here's an explanation of each module:

Admin Preprocessing: This module involves the registration process for the admin. The admin provides their name and password during registration. Once the admin is validated, they gain access to the system. The admin is responsible for various tasks, including data cleaning, adding CSV files, analyzing resources, performing NLP (Natural Language Processing) operations, and cleaning NLP data. Essentially, the admin handles the overall operation and maintenance of the system.

Auto Tagging Questions:

In this module, users register by providing their details. Upon successful login using their name and password, users enter the system. They have the option to choose categories of domains presented on the page. By clicking on the categories, a list of available domains is displayed. Users can select a domain and view related information, questions, and answers. If the requested information or question answer is not available, the system provides PDFs associated with that domain. Users also have the option to ask questions to the designated expert (doctor) who can provide clear answers.

Expert's Answering Process:

In this module, experts (doctors) need to register by providing their name, unique expert ID, and their specific domain/category. After logging in with valid

credentials, experts receive notifications about user questions. They analyze the questions and provide related answers, which are then forwarded to the user's profile. Users can view the answers at any time.

Student Assessment:

In this module, students register and upon login, they are directed to the quiz page. The quiz focuses on the medical domain. Students answer questions related to the chosen domain. After completing the quiz, students receive an overall score for the test, as well as a domain score. They also receive feedback to help them improve their knowledge in the chosen domain.

These modules collectively form the system, enabling admin management, question tagging and retrieval, expert answering, and student assessment in the medical domain.

Chapter 4

Results

4.1 Quantity of data

The data is quantitative and it will be represented in some vector or numeric format. At first we wait for some words in data, but for machine learning analysis, the words will be represented in vector or digit format. We chose statistical analysis for analyzing data from experiments.

The data have 50.000 rows, and have about 8 million words in the data. Including the title of problem, body and related tags. Problem body column contains a detailed description of each problem, which can be used to extract features for problem classification and prediction The problem tags column contains information about the tags associated with each problem. These tags can be used to train a model for automatic tagging of algorithmic problems. The tags is well distributed, and we do not need use extra sampling methods to avoid that.

In the pre-processing stage, several essential steps are taken to prepare the text data for further analysis. Firstly, special characters are removed from both the title and body of the text. This helps eliminate any unwanted symbols or punctuation that may hinder the accuracy of subsequent processing steps. Next, stop words are removed, which are commonly occurring words in a language that typically do not contribute much meaning to the text analysis. Removing stop words helps reduce noise and focus on the more significant content words. Ad-

ditionally, HTML tags are eliminated, ensuring that any remaining markup or formatting elements from web pages do not interfere with the subsequent analysis. To ensure consistency and avoid issues with word matching, all characters in the text are converted to lowercase. Lastly, the words are lemmatized, which involves reducing them to their base or dictionary form. This step helps capture the core meaning of the words and reduces variations, such as different tenses or plural forms, to enhance the accuracy of subsequent analysis tasks. By performing these pre-processing steps, the text data is cleansed and standardized, ready for various natural language processing techniques and machine learning algorithms. The code is located in Appendix A.

In the future engineering steps, we employ the CountVectorizer to convert our text documents into a matrix of token counts. This process allows us to represent the textual data in a numerical form, enabling further analysis. Additionally, we utilize the tf-idf transformer to transform the count matrix into a normalized tf-idf representation by formula 4.1.1. This transformation helps capture the importance of each token in the document collection by taking into account both the term frequency (tf) by formula 4.1.2 and the inverse document frequency (idf) by formula 4.1.3. By incorporating these techniques, we enhance the effectiveness and depth of our text analysis and modeling approaches.

$$tf\ idf(t, d, D) = tf(t, D) * idf(t, D) \tag{4.1.1}$$

$$tf\ (t, d) = log(1 + freq(t, d)) \tag{4.1.2}$$

$$idf(t, D) = log(\frac{N}{count(d \in D : t \in d)}) \tag{4.1.3}$$

4.2 Main results of work

4.2.1 Naive Bayes Classifier

So, our work is to discriminate tag based on problem description.

The Naive Bayes Classifier perfectly fits for baseline, to get a starting point of our results. A Naive Bayes classifier is a probabilistic machine learning model for classification with indicating formula 4.2.1. The classifier's crux is based on the Bayes theorem.

Bayes Theorem:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)} \tag{4.2.1}$$

We are using MultinomialNB classifier, that is one of the two classic naive Bayes variants used in text classification, and it implements the naive Bayes algorithm for multinomially distributed data.

Naive Bayes algorithms are commonly used in sentiment analysis, spam filtering, recommendation systems, and other applications. They are quick and simple to implement, but their main limitation is that predictors must be independent. In most real-world scenarios, the predictors are dependent, which reduces the classifier's performance.

We fit 74% of accuracy with Nayev Bayes in table 4.1. The code is located in Appendix B.

4.2.2 Support Vector Machine

Another basic approach is the support vector machine. With minimal compute power, the support vector machine achieves great accuracy. SVM can be applied to both regression and classification problems. The SVM algorithm's main goal is to determine the best hyperplane in an N-dimensional space that can split data points into different classes in the feature space.

We fit 79% of accuracy in table 4.1, that is more by 5% over Nayev Bayes.

4.2.3 Logistic Regression

Logistic Regression is a statistical model used in binary classification tasks to predict the likelihood of an instance belonging to a specific class. It can also

be enhanced to handle multi-class classification by employing approaches such as one-versus-rest or softmax regression.

Logistic regression is designed to describe the relationship between input variables (features) and the probability of a binary outcome. It assumes a linear relationship between the features and the outcome log-odds. A logistic function, often known as the sigmoid function (4.2.2), is then used to convert the log-odds into probabilities.

We fit 78% of accuracy with Logistic Regression in table 4.1.

Logistic regression function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.2.2}$$

4.2.4 Word2vec and Log Regression

Word2Vec is a set of models meant to generate word embeddings. These models are made up of rudimentary neural networks with two layers, and their goal is to learn how to rebuild words' surrounding linguistic context.

A vector is a mathematical construct that represents both magnitude and direction and aids in determining the position of one point in space relative to another. If we can mathematically manipulate the vectorized forms of words, it opens up possibilities for comparing words in a multi-dimensional space.

Behind the scene, why word2vec is good enough to use, because with one-hot encoding we calculate the numeric format of words, but loose the meaning of words.

We fit 64% of accuracy with Word2Vec and Log regression in table 4.1.

4.2.5 Doc2vec and Log Regression

The same concept like in word2vec but instead of tokenizing words, we will tokenize the sentences. And doc2vec is actually based on word2vec with some improvements using meaning of sentences.

	accuracy score
Multinomial Naive Bayes	0.74
Linear Support Vector Machine	0.79
Logistic Regression	0.78
Word2vec and Log Regression	0.64
Doc2vec and Log Regression	0.80

Table 4.1: Accuracy score of models.

	precision	recall	f1-score	support
Multinomial Naive Bayes	0.75	0.74	0.74	12000
Linear Support Vector Machine	0.79	0.79	0.78	12000
Logistic Regression	0.78	0.78	0.78	12000
Word2vec and Log Regression	0.63	0.64	0.64	12000
Doc2vec and Log Regression	0.80	0.80	0.80	12000

Table 4.2: Results metrics of models.

We achieved 80% of accuracy with Doc2vec and Log Regression in table 4.1. And all calculated metrics results of precision, recall and f1-score in table 4.2

Chapter 5

Conclusion

The present study delved into the domain of multi-class text classification, specifically examining the performance of machine learning approaches. Our findings emphasize the existing gaps in the capabilities of machine learning techniques and underscore the need for further advancements in the form of deep learning models.

Furthermore, we conducted extensive experimentation with diverse text representation architectures, leading us to identify LinearSVC as the optimal choice in terms of performance. These results offer valuable insights into enhancing the effectiveness and efficiency of text classification tasks.

The study also shed light on the pervasive challenge of working with limited data, a recurring issue in the field of machine learning. By encountering and navigating these obstacles firsthand, we gained a deeper understanding of the practical implications and potential strategies for addressing this challenge in future research.

Appendix A

Pre-processing

```
def clean_text(text):
    text = BeautifulSoup(text, "lxml").text
    text = text.lower()
    text = re.compile('[/(){}\[\]\|@,;]').sub(' ', text)
    text = re.compile('[^0-9a-z #+_]').sub('', text)
    text = ' '.join(word for word in text.split() if word not in set(
   stopwords.words('english')))
    return text


df['posts'] = df['posts'].apply(clean_text)
```

Appendix B

Text Classification of models

```
# Splitting the data into training and testing sets
X = df.posts
y = df.tags
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
    random_state=42)
# Creating a pipeline for Naive Bayes classifier
nb = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB()),
])
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)
# Creating a pipeline for SGD classifier
sgd = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', SGDClassifier(loss='hinge', penalty='l2', alpha=1e-3,
    random_state=42, max_iter=3)),
])
sgd.fit(X_train, y_train)
y_pred_sgd = sgd.predict(X_test)
# Creating a pipeline for Logistic Regression classifier
logreg = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', LogisticRegression(n_jobs=1, C=1e5)),
])
logreg.fit(X_train, y_train)
y_pred_logreg = logreg.predict(X_test)
```

Bibliography

[1] Carlos Basto. Extending the abstraction of personality types based on mbti with machine learning and natural language processing. arXiv preprint arXiv:2105.11798, 2021.

[2] Sahraoui Dhelim, Nyothiri Aung, Mohammed Amine Bouras, Huansheng Ning, and Erik Cambria. A survey on personality-aware recommendation systems. Artificial Intelligence Review.

[3] Kurt Shuster, Samuel Humeau, Hexiang Hu, Antoine Bordes, and Jason Weston. Engaging image captioning via personality. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12516–12526, 2019.

[4] Ana Carolina ES Lima and Leandro Nunes de Castro. Tecla: A temperament and psychological type prediction framework from twitter data. Plos one, 14 (3):e0212844, 2019.

[5] Deborah L McGuinness, Frank Van Harmelen, et al. Owl web ontology language overview. W3C recommendation, 10(10):2004, 2004.

[6] John F Sowa. Principles of semantic networks: Explorations in the representation of knowledge. Morgan Kaufmann, 2014.

[7] Robert F Simmons et al. Semantic networks: Their computation and use for understanding English sentences. Department of Computer Sciences and Computer-Assisted Instruction Laboratory . . . , 1972.

[8] Christiane Fellbaum, Musa Alkhalifa, W Black, Sabri Elkateb, Adam Pease,

H Rodriguez, and PTJM Vossen. Introducing the arabic wordnet project. In Third Global Wordnet Conference, 2006.

[9] Erik T Mueller. Natural language processing with thought treasure, 1998.

[10] George W Furnas, Caterina Fake, Luis von Ahn, Joshua Schachter, Scott Golder, Kevin Fox, Marc Davis, Cameron Marlow, and Mor Naaman. Why do tagging systems work? In CHI'06 extended abstracts on Human factors in computing systems, pages 36–39, 2006.

[11] Sanjay Sood, Sara Owsley, Kristian J Hammond, and Larry Birnbaum. Tagassist: Automatic tag suggestion for blog posts. In ICWSM. Citeseer, 2007.

[12] Divyam Madaan, Jinwoo Shin, and Sung Ju Hwang. Learning to generate noise for multi-attack robustness. In International Conference on Machine Learning, pages 7279–7289. PMLR, 2021.

[13] Daniel G Goldstein and Gerd Gigerenzer. The recognition heuristic: How ignorance makes us smart. In Simple heuristics that make us smart, pages 37–58. Oxford University Press, 1999.

[14] Endel Tulving. Elements of episodic memory oxford univ, 1983.

[15] Christopher H Brooks and Nancy Montanez. Improved annotation of the blogosphere via autotagging and hierarchical clustering. In Proceedings of the 15th international conference on World Wide Web, pages 625–632, 2006.

[16] Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In Proceedings of the 15th international conference on World Wide Web, pages 953–954, 2006.

[17] Ankita Mangal and Nishant Kumar. Using big data to enhance the bosch production line performance: A kaggle challenge. In 2016 IEEE international conference on big data (big data), pages 2029–2035. IEEE, 2016.

[18] Leif E Peterson. K-nearest neighbor. Scholarpedia, 4(2):1883, 2009.

[19] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Scharwächter,

Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset. In CVPR Workshop on the Future of Datasets in Vision, volume 2. sn, 2015.

[20] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings, pages 986–996. Springer, 2003.

[21] Oliver Kramer and Oliver Kramer. Scikit-learn. Machine learning for evolution strategies, pages 45–53, 2016.

[22] Mahesh Pal. Random forest classifier for remote sensing classification. International journal of remote sensing, 26(1):217–222, 2005.

[23] Mariana Belgiu and Lucian Drăguţ. Random forest in remote sensing: A review of applications and future directions. ISPRS journal of photogrammetry and remote sensing, 114:24–31, 2016.

[24] Wen-Chi Chou, Richard Tzong-Han Tsai, Ying-Shan Su, Wei Ku, Ting-Yi Sung, and Wen-Lian Hsu. A semi-automatic method for annotating a biomedical proposition bank. In Proceedings of the workshop on frontiers in linguistically annotated corpora 2006, pages 5–12, 2006.

[25] M Kathiravan, A Irumporai, S Sreesubha, and M Madhurani. Auto question tagging for health care using machine learning technique. Advances in Parallel Computing Technologies and Applications, 40:18, 2021.