

A
Summer Internship Report
On
Exploring and Developing solutions using LLMs

(IT346 – Summer Internship - I)

Prepared by
Kshitiz Pandya (21IT100)

Under the Supervision of
Prof. Rajnik Kataria

Submitted to
Charotar University of Science & Technology (CHARUSAT)
for the Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology (B.Tech.)
for Semester 5

Submitted at



**SMT. KUNDANBEN DINSHA PATEL DEPARTMENT OF
INFORMATION TECHNOLOGY**

Chandubhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT
At: Changa, Dist: Anand, Pin: 388421.
August, 2023



Accredited with Grade A+ by NAAC
Accredited with Grade A by KCG

CERTIFICATE

This is to certify that the report entitled “**Exploring and Developing Solutions using LLMs**” is a bonafied work carried out by **Kshitiz Pandya (21IT100)** under the guidance and supervision of **Prof. Rajnik Kataria & Mr. Apurva Nagavenkar** for the subject **Summer Internship – I (IT346)** of 5th Semester of Bachelor of Technology in **Department of Information** at Chandubhai S. Patel Institute of Technology (CSPIT), Faculty of Technology & Engineering (FTE) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred by the examiner(s).

Under the supervision of,

Prof. Rajnik Kataria
Assistant Professor
Smt. Kundanben Dinsha Patel Department of
Information Technology
CSPIT, FTE, CHARUSAT, Changa, Gujarat

Mr. Apurva Nagavenkar
Principal Data Scientist
Trinka.AI
Crimson Interactive Pvt. Ltd.

Dr. Parth Shah
Head of Department (IT)
CHARUSAT, Changa, Gujarat.

Chandubhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT

At: Changa, Ta. Petlad, Dist. Anand, Pin: 388421. Gujarat.

ACKNOWLEDGEMENT

Every work that one completes successfully stands on the constant encouragement, good will and support of the people around. I hereby avail this opportunity to express gratitude to number of people who extended their valuable time, full support and cooperation in developing the project.

I express deep sense of gratitude towards our **HoD, Dr. Parth Shah** for the support during the whole session of study and development. Given the opportunity, I also want to thank **Dr. Priyanka Patel** and **Prof. Jalpesh Vasa** for constant support and guidance throughout the time being. It is because of them, that I was prompted to do hard work and adopting new technologies.

I am immensely grateful to **Dr. Sourish Dasgputa, Founder of RAx (Faculty at DA-IICT)** to provide me this wonderful opportunity and at the same time and also thankful to all my mentors at the company. for their at-most patience and highly friendly nature who helped me understand and forge a lot of basics. The camaraderie and shared enthusiasm among the whole Data Science and Research Team is wonderful. Moreover, I would like to express my heartfelt gratitude to the entire **Crimson Interactive Team** for their constant support, guidance, and trust in my abilities. Also, this journey would not have been a success, without constant support and efforts of my parents **Mr. Devang Pandya** and **Mrs. Swati Pandya**.

I am sincerely thankful to all who helped me complete the project in one way or the other. They altogether provided us favorable environment, and without them it would not have been possible to achieve my goal.

ABSTRACT

As a member of Data Science and Research Department, I worked actively with the department officials to **Produce Products and Solutions using LLMs** so that we can solve various problems and difficulties in the industry of Scholarly Documentation.

The main focus of our Data Science and Research Department is to **leverage Large Language Models (LLMs)** to develop innovative products and solutions that address the challenges and complexities in the industry of Scholarly Documentation.

With the advancements in **Natural Language Processing (NLP)** and the capabilities of LLMs like GPT-3.5 and GPT-4, we aim to revolutionize the way scholarly information is processed, analyzed, and disseminated.

Our team collaborated closely with department officials and stakeholders to identify key problem areas and pain points within the scholarly documentation domain.

Table of Contents

Acknowledgement	iii
Abstract	iv
Chapter 1 Overview of the Organization.....	1
1.1 Company Description.....	1
1.1.1 Company Values	1
1.2 Locations and Scope of Work/Domain	2
1.2.1 Locations	2
1.2.2 Scope of Work/Domain	3
1.3 Organizational Hierarchy, Awards and Valued Clients	3
1.3.1 Organizational Hierarchy	3
1.3.2 Awards	3
1.3.3 Valued Clients	4
Chapter 2 Learning Period	5
2.1 Linux Fundamental	5
2.2 Basic Understanding about AWS services.....	5
2.3 Essential Python and Asynchronous Python	6
2.3.1 Essential Python	6
2.3.2 Asynchronous Python.....	8
2.4 Data Engineering.....	9
2.5 Recommendation Engines.....	10
2.5.1 Lucene in Action	10
2.5.2 Elasticsearch Gothrough.....	12
2.6 Natural Language Processing (NLP) Fundamentals	14
2.6.1 Classical NLP	14
2.6.2 Neural Network Based NLP	15
Chapter 3 Implementation / Work.....	17
3.1 Competitor Analysis.....	17
3.2 OpenAI models capability to simplify sentences	19

3.3 OpenAI models capability to remove jargons from sentences.....	19
3.4 Scripting on larger data	20
3.5 Domain based classification and simplification.....	23
3.6 Dataset search for finetuning GPT for defined tasks	24
3.7 Grammar Correction Scripting.....	25
3.8 Scripting of 3 major projects in Trinka - AI.....	26
Conclusion	28
References	29

List of Figures

Fig. 1.2.1 Locations around the globe.....	2
Fig. 1.3.1 Organization structure and branches	3
Fig. 1.3.2 Awards acquired so far	3
Fig. 1.3.3 Values Clients of the Company	3
Fig. 1.3.2 Awards acquired so far	4
Fig. 3.1.1 Interface of Scispace.....	17
Fig. 3.2.1 Implementation code for different models for simplification	19
Fig. 3.7.1 Input format of API	25
Fig. 3.7.1 Output format of API.....	25
Fig. 3.7.2 Example input provided.....	26
Fig. 3.7.2 Output of the example	26

List of Tables

Table 3.1	Simplicity - OVERALL.....	21
Table 3.2	Simplicity – Domain – YES (57s)	21
Table 3.3	Simplicity – Domain – NO (45s).....	21
Table 3.4	Adequacy – OVERALL.....	22
Table 3.5	Adequacy – Domain – YES (57s).....	22
Table 3.6	Adequacy – Domain – NO (45s)	22
Table 3.7	Fluency – OVERALL.....	23
Table 3.8	Fluency – Domain – YES (57s).....	23
Table 3.9	Fluency – Domain – NO (45s).....	23




CHAPTER 1: OVERVIEW OF THE ORGANIZATION

1.1: COMPANY DESCRIPTION

Crimson offers a robust ecosystem of services with cutting-edge AI and learning products for researchers, publishers, societies, universities, and government research bodies worldwide. With a global presence, including 9 international offices, we cater to the communication needs of the scientific community and corporates. We operate globally, with regional teams supporting clients locally. Our events act as open forums, bringing together key research stakeholders like government policymakers, marquee publishers and universities, Nobel laureates, and researchers. Our team comprises an enthusiastic and passionate lot of people fuelled by our open culture to learn, share, and grow with each other, carving a niche and achieving great heights for Crimson and themselves.

1.1.1. Company Values

Our value system is the core of Crimson. It defines the way we work, think, and act and is a strong influence on our success. Our values encourage a clear balance between our working style and the plans we've charted out for our clients and ourselves.

- **Integrity** 
We are honest to our work, our peers, and our clients. We build truthful relationships with fellow Crimsonites and do the right thing always, even when no one is watching. We work to give our clients the best possible assistance through our services, while keeping in mind and considering various limitations they could have.
- **Commitment** 
We are a large team of professionals working together toward meeting our commitment to our clients. We are true to our word and go to any extent to fulfil what we promise to ourselves, peers, and our clients. Our efforts do not decide our commitment; rather, our commitment decides our efforts! For us, a commitment is carved in stone and has to be met.
- **Respect** 
We are a diverse team of people sharing utmost respect for each other's cultures, personalities, and individualities. We respectfully accept different opinions and suggestions. Our respect for our clients goes beyond boundaries in our unwavering and tireless efforts to fulfil their requirements and offer them the best service.

- **Simplicity**



Our philosophy to anything we do is defined by simplicity. Our ideas may be different but simple. Our processes are simple, our expectations are clear, and our approach to serve our clients is transparent. Our simplicity makes things practical and executable, setting us apart, without much complexity.

- **Kaizen**



“Kaizen” means “continuous improvement” and this is exactly what we live by. We are constantly looking for means to make the good better and make the better the best. This value helps us adapt to change comfortably and easily and still be relevant in what we offer and how. Creative thinking and innovation are encouraged and rewarded here.

1.2: LOCATIONS AND SCOPE OF WORK / DOMAIN

1.2.1. Locations

Crimson is a multinational corporation with 10 offices in key knowledge centers worldwide, including New Jersey, London, Seoul, Beijing, Shanghai, Tokyo, Istanbul, Bogota, Hyderabad and Mumbai. With clients in more than 125 countries and offices in 4 continents, we take pride in our extensive global network of employees, clients, and collaborators.



Fig. 1.2.1 Locations around the globe

1.2.2. Scope of Work / Domain

- Artificial Intelligence / Machine Learning
- Cutting – edge NLP research
- Hardcore Engineering
- Language Polishing
- Translation
- Publication Support and research

1.3: ORGANIZATIONAL HIERARCHY, AWARDS AND VALUED CLIENTS

1.3.1. Organizational Hierarchy

Crimson's brand portfolio spans across different centers of excellence catering to specific, unique requirements, each providing its own essence and expertise.



Fig. 1.3.1 Organization structure and branches

1.3.2. Awards

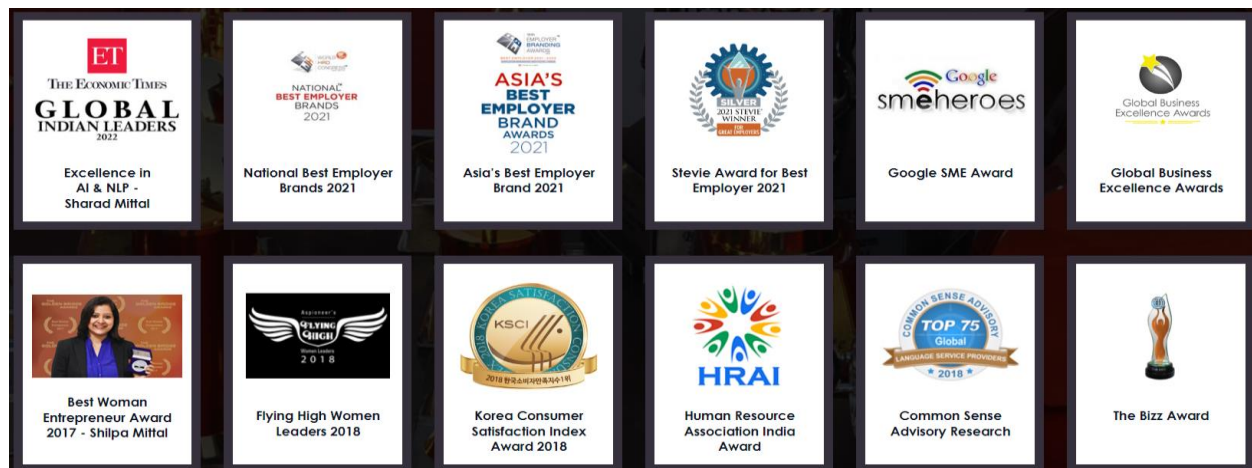


Fig. 1.3.2 Awards acquired so far

1.3.3. Values Clients



Fig. 1.3.3 Values Clients of the Company

CHAPTER 2: LEARNING PERIOD

2.1: LINUX FUNDAMENTALS ^[1]

During the initial phase of my internship, my primary focus was on acquiring and enhancing my skills in Linux, an extensively used operating system in the industry. Understanding Linux is crucial as it simplifies deployment and scripting processes.

The main topics assigned to me for exploration were as follows:

a) Understanding Linux File Permissions:

- I learned about file permissions and their significance in Linux.
- Some essential commands like **ls**, **cd**, and **mkdir** were covered, helping me navigate and manipulate directories and files.

b) Managing File Systems:

- I delved into managing file systems effectively.
- I gained insights into essential commands like **chmod**, **chown**, **df**, and **du**, which enable me to control access permissions, change ownership, and monitor disk usage.

c) Bash Shell Commands for Searching, Sorting, and Compressing Data:

- I familiarized myself with various Bash shell commands that are useful for searching, sorting, and compressing data.
- Commands such as **ps**, **top**, **htop**, **kill**, and **netstat** provided me with valuable insights into process management and network monitoring.

d) Sed and Gawk:

- I learned about stream text editing with **sed** and text processing with **awk**.
- Commands like **head**, **tail**, **grep**, **cut**, **sort**, **uniq**, and **sed** were covered, allowing me to perform powerful text manipulation tasks.

e) Regular Expressions:

- I delved into regular expressions, which are essential for pattern matching and text processing in Linux.
- I learned how to apply regular expressions in various contexts, such as standard input/output/error, pipes, permissions, and user input handling.

Mastering these Linux topics has equipped me with valuable skills and knowledge that will prove instrumental in my future endeavors in the world of IT and software development. Understanding Linux thoroughly has laid a strong foundation for me to excel in various tasks related to system administration, development, and automation.

2.2: BASIC UNDERSTANDING ABOUT AWS SERVICES

As part of my internship, I was assigned the task of learning the fundamentals of Amazon Web Services (AWS). During this phase, I gained a basic understanding of some key AWS services that are widely used in cloud computing. Below are short notes on each of these services:

a) S3 (Simple Storage Service):

S3 is a scalable object storage service provided by AWS. It allows you to store and retrieve any amount of data at any time from the web. Data is stored in "buckets," and each bucket can contain multiple objects (files). S3 provides high durability and availability with a 99.999999999% (11 nines) durability guarantee. It is commonly used for hosting static websites, storing backups, and serving large media files.

b) EC2 (Elastic Compute Cloud):

EC2 is a web service that provides resizable compute capacity in the cloud. It allows users to rent virtual machines (instances) on which they can run applications. EC2 instances are available in various types, each optimized for specific use cases such as general computing, memory-intensive tasks, or GPU processing. Users have full control over these instances, including the operating system, software, and network configurations.

c) API Gateway:

API Gateway is a fully managed service that enables the creation, deployment, and management of APIs at scale. It acts as a front door for applications to access data, business logic, or functionality from your backend services. API Gateway supports RESTful and WebSocket APIs and allows for integration with AWS Lambda, EC2, or other backend services. It offers features like request and response transformations, caching, throttling, and authentication, making it a powerful tool for building robust APIs.

d) RDS (Relational Database Service):

RDS is a managed database service that simplifies the setup, operation, and scaling of relational databases. It supports popular database engines like MySQL, PostgreSQL, Oracle, SQL Server, and Amazon Aurora. RDS automates routine database tasks such as backups, software patching, and hardware provisioning, reducing administrative overhead. It provides high availability and fault tolerance by replicating databases across multiple availability zones. These are the foundational AWS services that I have learned during my internship, and they serve as a strong starting point for building a deeper understanding of cloud computing and AWS as a whole.

2.3: ESSENTIAL PYTHON AND ASYNCHRONOUS PYTHON ^[2]

2.3.1. Essential Python

I revisited and solidified the fundamentals of Python, which included the following essential topics:

a) Python Data Model:

- The Python data model refers to the way objects, data types, and operations work in Python. It defines how Python objects behave, including their creation, representation, and interaction.
- Python objects are based on classes, and every object has a unique identity, type, and value.
- Understanding the data model allows developers to work effectively with Python's built-in data types and create custom classes with the desired behavior.

b) Python Data Structures:

- Python offers several built-in data structures to organize and store data efficiently.
- Arrays: Lists in Python act as dynamic arrays, allowing for efficient storage and retrieval of elements.
- Sequences: Sequences include lists, tuples, and strings, providing ordered collections of items with indexing and slicing capabilities.
- Dictionaries: Dictionaries are key-value pairs, providing fast access to values based on their keys.
- Sets: Sets are unordered collections of unique elements, allowing for set operations like union, intersection, and difference.
- Text vs. Bytes: Python differentiates between text (Unicode) and bytes (binary data). Understanding this difference is essential when working with different encodings and data formats.

c) Control Flow:

- Control flow constructs in Python allow developers to manage the flow of execution in their programs.
- Conditional Statements: Python uses **if**, **elif**, and **else** statements to execute different code blocks based on certain conditions.
- Loops: Python provides **for** and **while** loops to iterate over sequences or execute code repeatedly as long as a specific condition is met.
- Exception Handling: **try**, **except**, **finally** blocks enable developers to handle exceptions and prevent program crashes when unexpected errors occur.

d) Iterables, Iterators, and Generators:

- Iterables: Any object that can be looped over, such as lists, tuples, and strings, is an iterable. They can be used with loops and comprehensions.
- Iterators: Iterators are objects that implement two methods, `__iter__()` and `__next__()`, allowing sequential access to elements. Iterators help conserve memory as they don't load the entire data into memory at once.
- Generators: Generators are a type of iterator that generates values on-the-fly using the **yield** keyword. They are useful for working with large datasets without storing them entirely in memory.

e) Concurrency with Asyncio:

- Python's **asyncio** library enables asynchronous programming, which allows handling multiple tasks concurrently without blocking the execution.
- **async** and **await** keywords are used to define asynchronous functions that can pause their execution and allow other tasks to run.
- By using asynchronous programming, developers can write more responsive and scalable applications, especially when dealing with I/O-bound operations.

By revisiting and mastering these fundamental concepts, I have gained a solid understanding of Python's core principles, making me a more competent and versatile Python developer. These skills are essential for building robust and efficient software solutions in a wide range of domains, from web development to data science and beyond.

2.3.2. Asynchronous Python ^[5]

I focused on learning about asynchronous programming in Python. Asynchronous programming allows for concurrent execution of tasks without blocking the main thread, making it particularly useful for handling I/O-bound operations and improving the performance of applications. The following points summarize my key takeaways from the provided documentation:

a) Introduction to Asynchronous Programming:

- Asynchronous programming enables the execution of multiple tasks concurrently, allowing for better utilization of system resources.
- It is particularly useful for tasks that involve waiting for I/O operations, such as reading and writing files or making network requests.

b) Understanding `async` and `await` ^[6]:

- Python's **`asyncio`** library provides support for writing asynchronous code using the **`async`** and **`await`** keywords.
- Functions defined with **`async def`** are asynchronous functions that can be paused and resumed during their execution.
- **`await`** is used within asynchronous functions to pause execution until the awaited task completes.

c) Event Loops and Coroutines:

- Asynchronous programming in Python revolves around event loops, which manage the execution of asynchronous tasks.
- Coroutines are special functions that yield control back to the event loop using **`await`**, allowing other tasks to execute while waiting for I/O operations to complete.

d) Async I/O and Non-Blocking Operations ^[7]:

- Python's **`asyncio`** supports asynchronous I/O operations for reading and writing data without blocking the main thread.
- Non-blocking operations enable applications to perform other tasks while waiting for I/O responses, leading to increased efficiency.

e) `asyncio` Module Features:

- The **`asyncio`** module provides various features, including **`asyncio.Task`**, **`asyncio.gather`**, and **`asyncio.wait`**, to manage and synchronize multiple asynchronous tasks.
- **`asyncio.Queue`** facilitates communication between coroutines, allowing for data exchange in a thread-safe manner.

f) Understanding async Context Managers:

- Asynchronous context managers are used with the **async with** statement and provide a way to manage resources asynchronously.
- They ensure proper resource cleanup and release when the asynchronous block is exited.

g) Concurrency with asyncio and Threads:

- **asyncio** can work seamlessly with threads, combining the benefits of both asynchronous and multithreaded programming.
- However, care must be taken to avoid mixing blocking I/O calls with asynchronous code to prevent performance issues.

In conclusion, my exploration of asynchronous Python using the provided documentation has expanded my understanding of concurrent programming and its benefits in I/O-bound scenarios. Asynchronous programming in Python, with the help of **asyncio**, offers an efficient approach to handling tasks concurrently and enhancing the performance of modern applications.

2.4: DATA ENGINEERING

I delved into data engineering, with a focus on web scraping using Scrapy. Web scraping is the process of extracting data from websites, and Scrapy is a powerful Python library that facilitates this task efficiently. Here are the key points from my experience:

a) Scrapy Tutorial Series ^[8]:

- I followed the Scrapy tutorial series provided by AccordBox. The tutorial comprehensively covered the basics of Scrapy, including setting up the project, creating spiders, and extracting data from websites.
- The tutorial also introduced concepts such as XPath and CSS selectors, which are essential for navigating and extracting data from HTML documents.

b) Podcast Recommendation Project:

- I worked with the "podcast_recommendation" project available on GitHub, created by one of my mentor.
- The project showcased a practical implementation of web scraping using Scrapy to extract podcast data from various sources.

c) Running Scrapy Locally:

- I successfully set up the Scrapy project and executed the spiders on my local machine.

- By running Scrapy locally, I gained hands-on experience in inspecting the scraped data and fine-tuning the scraping process.

d) TED Data Scraping ^[9]:

- As part of the task, I performed web scraping to collect TED Talk data using different search queries.
- I defined custom spiders in Scrapy to target specific TED Talk pages, extracted information such as titles, speakers, durations, and descriptions.

e) Challenges and Solutions:

- During the task, I encountered challenges related to handling website structures, dynamic content, and rate limiting to avoid overloading servers.
- To overcome these challenges, I applied techniques like user-agent rotation, handling pagination, and using `sleep()` to introduce delays between requests.

f) Data Processing and Storage:

- After scraping the TED Talk data, I processed and stored it in various formats such as CSV or JSON.
- Proper data processing and storage are crucial for preparing the scraped data for further analysis or integration into other systems.

2.5: RECOMMENDATION ENGINES

Recommendation engines are data-driven systems that provide personalized suggestions to users. They are widely used in various domains, including e-commerce, content streaming, and social media platforms. The primary goal of a recommendation engine is to enhance user experience by presenting relevant items, products, or content, thereby increasing user engagement and satisfaction. Recommendation engines utilize different algorithms, such as collaborative filtering, content-based filtering, and hybrid approaches, to generate recommendations based on user behavior, preferences, and item characteristics. These engines play a vital role in driving user engagement and increasing business revenue by promoting relevant and personalized content to users.

2.5.1. Lucene in Action ^[3]

Chapter 1 of the book "Lucene in Action" introduces the core components of a search application and provides an overview of Lucene, a widely used open-source search engine library written in Java. The chapter highlights the fundamental building blocks of a search application and how Lucene fits into the search ecosystem.

a) Understanding Search Applications:

- The chapter explains the key components of a search application, including data indexing, searching, and relevance ranking.
- It emphasizes the importance of efficient indexing and retrieval of data to provide fast and relevant search results to users.

b) Introduction to Lucene:

- Lucene is introduced as a powerful and flexible search engine library that enables developers to implement search functionality in their applications.
- The chapter provides an overview of Lucene's architecture and its core features, including indexing, searching, and scoring.

c) Indexing Documents with Lucene:

- The process of indexing documents, converting text data into a format suitable for efficient searching, is explained in detail.
- Lucene's indexing mechanism, which involves analyzing and tokenizing text, is discussed to create an inverted index for rapid lookup.

d) Searching with Lucene:

- The chapter explores the search process in Lucene, where user queries are matched against the indexed data to retrieve relevant documents.
- It covers the usage of Lucene's query classes and explains how scoring is used to rank search results based on relevance.

e) Scoring and Ranking:

- The concept of scoring is vital for search applications, as it determines the relevance of search results based on user queries.
- Lucene's scoring model, which combines term frequency and inverse document frequency, is introduced as a key factor in ranking search results.

In conclusion, **Chapter 1 of "Lucene in Action"** serves as a comprehensive introduction to the components of a search application and Lucene's role in facilitating efficient searching and indexing. Understanding the core principles of search applications and Lucene's capabilities will be beneficial for developers looking to build powerful and accurate search functionalities in their applications.

2.5.2. Elasticsearch Go-through

1. Elasticsearch:

- Elasticsearch is a distributed search and analytics engine based on Apache Lucene.
- It provides fast and scalable full-text search, real-time data indexing, and data analysis capabilities.
- Elasticsearch is commonly used for building powerful search applications, log monitoring systems, and data analytics platforms.

2. Kibana Local Setup ^[10]:

- Kibana is an open-source data visualization platform that works with Elasticsearch.
- Setting up Kibana locally allows users to explore, analyze, and visualize data stored in Elasticsearch.
- It provides a user-friendly interface for creating dashboards, charts, and graphs.

3. Chapter 15 - Iterables, Iterators, Generators ^[11]:

- This chapter covers Python's iterable objects, iterators, and generators.
- Iterables are objects that can be looped over, like lists and dictionaries.
- Iterators are objects that provide sequential access to elements of an iterable using the `__iter__()` and `__next__()` methods.
- Generators are a type of iterator that use `yield` to produce values on-the-fly, conserving memory.

4. DSL Queries - Simple & Complex :

- In Elasticsearch, the DSL (Domain Specific Language) is used to define queries.
- Simple DSL queries involve basic match, term, or range queries.
- Complex DSL queries use aggregations, filters, and other advanced features to perform more sophisticated searches.

5. Mapping ^[12]:

- Mapping in Elasticsearch defines the schema or structure of the data in an index.
- It specifies the data types and how the data should be indexed and searched.

6. Indexes^[13]:

- In Elasticsearch, an index is a collection of documents that share similar characteristics.
- It serves as the main unit of data organization and search.

7. Shards and Replicas^[14]:

- Elasticsearch distributes data across multiple nodes using shards.
- Shards allow for horizontal scaling and improve performance.
- Replicas are additional copies of shards, providing fault tolerance and data redundancy.

8. TF-IDF Fundamentals^[15]:

- TF-IDF (Term Frequency-Inverse Document Frequency) is a numerical representation of the importance of a term in a document relative to a collection of documents.
- It is commonly used in information retrieval and text mining to rank the relevance of search results.

9. BM25 Algorithm^[16]:

- BM25 is a ranking algorithm commonly used in search engines.
- It improves upon TF-IDF by considering term frequency, document frequency, and document length for ranking search results.

10. Explore Key Index Structures^[17]:

- This topic likely involved investigating specific index structures used in Elasticsearch, such as "new_literature," "light reads," "references," "newsfeed," and "publication_venues."

11. Data Ingestion Pipeline^[18]:

- A data ingestion pipeline refers to a series of processes for collecting, preparing, and ingesting data into a storage system, like Elasticsearch.
- It ensures that data is efficiently and reliably transferred into the system for further analysis.

12. Data Cleaning (NLP Pre-processing) Module^[19]:

- NLP (Natural Language Processing) pre-processing involves cleaning and transforming text data before analysis.

2.6: NATURAL LANGUAGE PROCESSING (NLP) FUNDAMENTALS ^[4]

Natural Language Processing (NLP) is a subfield of artificial intelligence (AI) and computational linguistics that focuses on the interaction between computers and human language. It involves developing algorithms and models that enable computers to understand, interpret, and generate human language in a way that is both meaningful and useful.

2.6.1. Classical NLP ^[4]

Topics covered under this were as follows:

a) **Tokenization:**

- Tokenization involves breaking down a text into smaller units called tokens, such as words or subwords.
- Understanding tokenization is crucial for further NLP tasks, as it defines the basic building blocks for language processing.

b) **Part-of-Speech (POS) Tagging:**

- POS tagging involves assigning grammatical labels (nouns, verbs, adjectives, etc.) to each word in a sentence.
- This is essential for understanding the grammatical structure of sentences and extracting valuable information from text.

c) **Named Entity Recognition (NER):**

- NER aims to identify entities like names of people, organizations, locations, etc., in a text.
- It is a critical component in information extraction and knowledge graph construction.

d) **Parsing and Dependency Parsing:**

- Parsing involves analyzing the syntactic structure of sentences to understand their grammatical relationships.
- Dependency parsing establishes the grammatical dependency relationships between words in a sentence.

e) **Language Models and n-grams:**

- Language models help estimate the probability of a sequence of words in a sentence.
- N-grams represent contiguous sequences of n words and are used in various NLP tasks, including language modeling and information retrieval.

f) Feature Engineering:

- Classical NLP often relies on handcrafted features to represent text for machine learning models.
- Feature engineering plays a vital role in enhancing the performance of traditional NLP algorithms.

2.6.2. Neural Network Based NLP ^[4]**a) Word Embeddings:**

- Word embeddings are dense vector representations that capture semantic relationships between words.
- Techniques like Word2Vec, GloVe, and FastText are used to generate word embeddings.

b) Recurrent Neural Networks (RNNs):

- RNNs are a class of neural networks that process sequential data, making them suitable for natural language processing tasks.
- They have a memory component that allows them to maintain information about previous inputs.

c) Long Short-Term Memory (LSTM):

- LSTM is a variant of RNNs designed to mitigate the vanishing gradient problem, enabling better modeling of long-range dependencies in text.

d) Transformer Architecture ^[14]:

- The Transformer architecture, especially the attention mechanism, revolutionized NLP tasks by enabling parallel processing of words in a sentence.
- The popular model BERT (Bidirectional Encoder Representations from Transformers) uses the Transformer architecture.

e) Sequence-to-Sequence Models:

- Sequence-to-sequence models are used for tasks like machine translation and text summarization.
- They use an encoder-decoder architecture, with RNNs or Transformers, to handle input and generate output sequences.

f) Transfer Learning and Pre-trained Models:

- Transfer learning involves using pre-trained language models like BERT and GPT to perform downstream NLP tasks with limited labeled data.
- These pre-trained models capture rich language representations and can be fine-tuned for specific tasks.

Both Classical and Neural Network-based NLP approaches have their strengths and weaknesses. Classical NLP techniques are well-established and interpretable, while NN-based methods often achieve state-of-the-art performance with large amounts of data. Understanding both paradigms is crucial for building robust and effective NLP applications.

CHAPTER 3: IMPLEMENTATION / WORK

3.1: COMPETITOR ANALYSIS ^[15]

Scispace is an online platform that provides a comprehensive database of scientific research articles and publications. During my internship, I was assigned the task of conducting a competitor analysis on Scispace to understand its unique features and advantages in the market and find out that the same task can be achieved by openAI models.

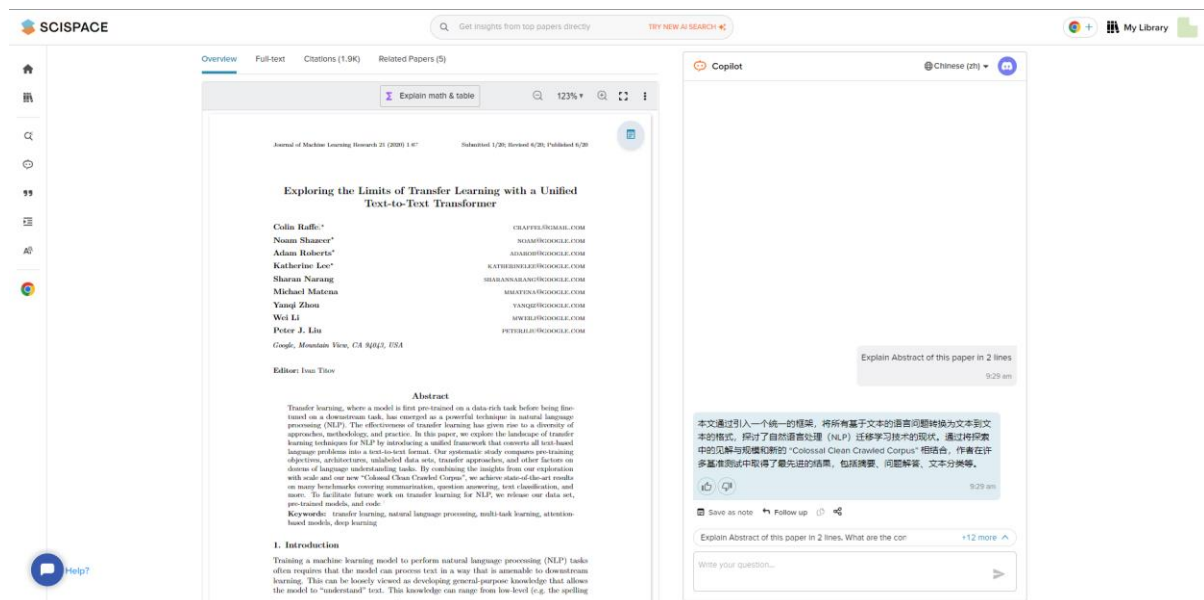


Fig. 3.1.1 Interface of Scispace

Key area of interest found were as follows:

a) Topic Highlighting:

- Scispace excels in highlighting topics in research articles using advanced natural language processing techniques.
- OpenAI's general-purpose language model might lack the specialized focus and precision in highlighting scientific topics.

b) Author Name Highlighting:

- Scispace prominently highlights the names of authors, adding a personal touch and improving author identification.
- OpenAI may not have the same level of emphasis on author names, potentially making it harder for users to identify researchers.

c) Headings and Subheadings:

- Scispace effectively highlights headings and subheadings, enhancing the organization and navigability of articles.
- OpenAI may not prioritize headings in the same way, making it less conducive to easy article scanning.

d) Text Highlighting:

- Scispace skillfully highlights important text passages, ensuring readers focus on critical content.
- OpenAI's highlighting might not be as specialized, possibly leading to less effective emphasis on vital information.

e) Follow-up Questions:

- Scispace uses follow-up question prompts to encourage deeper engagement and critical thinking.
- OpenAI may not have this specific feature designed for scientific research, potentially missing out on fostering active reader participation.

f) Math and Table Formatting:

- Scispace handles complex mathematical content and tables with precision, ensuring clarity for researchers.
- OpenAI's general-purpose nature might not offer the same level of formatting finesse for specialized content like equations and tables.

g) References Organization:

- Scispace likely has a well-organized system for handling references and citations within articles.
- OpenAI may not be as specialized in handling references in a scholarly context.

h) Comparing with OpenAI:

- Scispace's specialized focus on scientific research highlights makes it excel in the domain-specific tasks relevant to researchers.
- OpenAI's strength lies in its versatility across various applications, but it might not match Scispace's precision in scientific content highlighting.

3.2: OPENAI MODELS CAPABILITY TO SIMPLIFY SENTENCES

The objective was to prepare **prompts for simplifying the scientific sentences into common language and comparing compatibility of OpenAI playground different models versus ChatGPT. Identification of best prompt was to be found for the task.**

For checking different models of openAI a simple script was written to show the outputs the query given –

```
import openai
openai.api_key = "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"
x = openai.Edit.create(
    model="text-davinci-edit-001",
    input="XYZ....",
    instruction="simplify the input"
)
text = x['choices'][0]['text']
print(text)
```

Fig. 3.2.1 Implementation code for different models for simplification

The prompt for OpenAI as well as ChatGPT was tested on raw dataset provided by the organization and found out the conclusion that, it is clear that while performing the simplification, chatGPT is giving better results than openAI in accordance to the point that openAI is giving the same output as the input prompt without any changes.

3.3: OPENAI MODELS CAPABILITY TO REMOVE JARGONS FROM SENTENCES

The main objective here was simplifying the scientific sentences by doing following:

- **Identifying the Jargons/Scientific terms from the sentence.**
- **Using the Jargons to simplify the text.**

The necessary steps taken to complete the assigned task were as follows:

- Necessary inputs from various domains were taken.
- Inputs were given as prompt to openAI and chatGPT.
- Instruction prompt was specified - for example, simplify the text by replacing following words with simpler words, i.e., ABC, PQR, XYZ.
- Output was fetched and given to python script for finding the diff.

The final conclusion gained was - chatGPT is performing better than openAI. It might be because of its model is fine tuned to performed certain task.

3.4: SCRIPTING ON LARGER DATA

The experiment defined was - **Simplification Task on Life Science domain on provided dataset - using ChatGPT.**

The objective here was, Exploration of simplification task using chatGPT on ‘Life Science’ domain by considering the text’s core concept and subject area.

Data given was **5 documents** shared by the language team of the industry, of which .txt files were provided which had the selected sentences which had a lot of jargons which should be simplified.

The results achieved was displayed in a formatted .csv file of which directory structure is defined below:

Directory Structure

- Document name - Name of the document from which the sentence is extracted.
- Subject_area1, subject_area2, subject_area3 - Top 3 entries of each section of subject area identification API.
- Core_concepts - top 10 extracted concepts from concept extraction API.
- Sentence - Actual sentence worked on.
- Jargons - Jargons found from the sentence.
- Output (jargons elimination) - output chatgpt produced after eliminating jargons.
- diff (jargons elimination) - diff of the sentence and its respective jargons eliminated output.

Explanation of the task performed:

- This sentence was solely about **simplifying the sentences based on the given information of background of the text i.e., its subject area and core concepts.**
- Using the subject area and core concepts, prompts were designed to consider these parameters for simplification of the sentences.
- The final decided **prompt design** was then used for every sentences to generate our satisfactory output.

NOTE: There is no relation between the tasks - “Jargon elimination” and “Simplification considering Subject Area and Core Concepts”. There outputs are solely different and there is no relation or cross use between them.

The **final analysis** of the report provided by me was given by the **language team** of the industry after final assessment of the outputs.

Table 3.1 Simplicity - OVERALL

0	text is difficult to understand (or little to no simplification done)	7	
1	text is somewhat easy to understand (some words are not familiar to me)	17	
2	text is easy to read and understand	78	76.5%

Table 3.2 Simplicity – Domain – YES (57s)

0	text is difficult to understand (or little to no simplification done)	4	
1	text is somewhat easy to understand (some words are not familiar to me)	16	15.7%
2	text is easy to read and understand	37	36.3%

Table 3.3 Simplicity – Domain – NO (45s)

0	text is difficult to understand (or little to no simplification done)	3	
1	text is somewhat easy to understand (some words are not familiar to me)	1	1.0%
2	text is easy to read and understand	41	40.2%

Table 3.4 Adequacy – OVERALL

0	the meaning of the sentence is very different from the original.	14	
1	the sentence has a slightly different meaning from the original; some critical information or no information is missing.	44	43.1%
2	the sentence means the same as the original sentence; some noncritical information or not information may be missing.	44	43.1%

Table 3.5 Adequacy – Domain – YES (57s)

0	the meaning of the sentence is very different from the original.	8	
1	the sentence has a slightly different meaning from the original; some critical information or no information is missing.	30	29.4%
2	the sentence means the same as the original sentence; some noncritical information or no information may be missing.	19	18.6%

Table 3.6 Adequacy – Domain – NO (45s)

0	the meaning of the sentence is very different from the original.	6	
1	the sentence has a slightly different meaning from the original; some critical information or no information is missing.	14	13.7%
2	the sentence means the same as the original sentence; some noncritical information or not.	25	24.5%

Table 3.7 Fluency – OVERALL

0	Ungrammatical	7	
1	Few minor grammatical errors or unidiomatic phrases	17	
2	Grammatical & fluent	78	76.5%

Table 3.8 Fluency – Domain – YES (57s)

0	Ungrammatical	4	
1	Few minor grammatical errors or unidiomatic phrases	14	
2	Grammatical & fluent	39	38.2%

Table 3.9 Fluency – Domain – NO (45s)

0	Ungrammatical	3	
1	Few minor grammatical errors or unidiomatic phrases	3	
2	Grammatical & fluent	39	38.2%

3.5: DOMAIN BASED CLASSIFICATION AND SIMPLIFICATION

Problem statement: The thought is, every scholarly documents is not understandable by every person of different qualification background. For example, sometimes by reading a biological Scholarly document, an engineering background person can make up a solution for the described problem. But this task can be done by the individual if and only if the scholarly document is totally understood by the developer. The idea is to perform the simplification tasks on particular domain texts by giving the background of the person who wants it simplified. E.g. prompt - Simplify this text for a Computer Engineer: “In terms of the number of isolates from the three units examined, the uro-endoscopy unit recorded the highest followed by the general surgical theater and the GI endoscopy.”

Data given was **5 documents** shared by the language team of the industry, of which .txt files were provided which had the selected sentences which had a lot of jargons which should be simplified.

Explanation of the task (Sheet was prepared of which explanation is below):

1. In the first sheet, we have taken the 3 sentences, from the old data used for analysis, i.e., from `simplification_task_files.zip`, which are medical related data. The task performed in this sheet and its output is in accordance to the prompt to *simplify the given medical text for a 10 year old kid*.
2. In the second sheet, we have taken the 3 sentences, from the old data used for analysis, i.e., from `simplification_task_files.zip`, which are medical related data. The task performed in this sheet and its output is in accordance to the prompt to *simplify the given medical text for a student who has engineering background and has no knowledge of the domain of the text*.
3. In the third sheet, we have taken the 3 sentences, from the new paper i.e., from `Generative_Adversarial_Networks_for_Astronomical_I.docx`, which is engineering research paper. The task performed in this sheet and its output is in accordance to the prompt to *simplify the given engineering text for a 10 year old kid*.
4. In the third sheet, we have taken the 3 sentences, from the new paper i.e., from `Generative_Adversarial_Networks_for_Astronomical_I.docx`, which is engineering research paper. The task performed in this sheet and its output is in accordance to the prompt to *simplify the given engineering related text for a student who has medical background and has no knowledge of the domain of the text*.
5. Sheet 2 is under process to do the task of simplification for large amount of text.

3.6: DATASET SEARCH FOR FINETUNING GPT FOR DEFINED TASKS

Task was to search for the dataset available, which can help us improve and fine tune the GPT model on simplification tasks.

The findings were as follows:

1. Simple English Wikipedia: Simple English Wikipedia is a version of Wikipedia that uses simple English words and grammar to make the articles easier to understand. You can download the latest dump of Simple English Wikipedia articles from here: ^[16]
2. Newsela: Newsela is a website that offers news articles at different reading levels, including a simplified version for learners. You can request access to their API for research purposes here: ^[17].
3. WikiHow: WikiHow is a website that provides how-to guides on a wide range of topics. You can download their dataset of simplified summaries here: ^[18].

4. ASSET: ASSET is a dataset that contains parallel sentences in English and six other languages that have been simplified for language learners. You can request access to their dataset for research purposes here: ^[19].

5. TURKCORPUS: TURKCORPUS is a Turkish corpus that contains parallel sentences in Turkish and English, with the English sentences simplified for language learners. You can access their dataset here: ^[20].

3.7: GRAMMAR CORRECTION SCRIPTING

OpenAI released its new model, GPT-3.5-turbo. Using that model the task was to do grammar correction on white paper data provided.

The task was to create a FastAPI to perform the task iteratively on the dataset provided of which –

Input Format was:

```
{
  "sentence_info_list": [
    {"sentence": "I have book", "lang_type": "US", "file_id": null}
  ]
}
```

Fig. 3.7.1 Input format of API

```
{
  "result": [
    {
      "original_sentence": "I have book",
      "processed_sentence": "I have book",
      "output_sentence": "I have a book",
      "lang_type": "US",
      "file_id": null
    }
  ],
}
```

Fig. 3.7.2 Output format of API

The task was completed with flying colors. The image proof of work achieved is below:

Input given:

```
{
  "sentence_info_list": [
    {
      "sentence": "To find the effectiveness of hydroxychloroquine alone and adjuvant with azithromycin in mild to severe Covide-19 pneumonia patients admitted to Coronavirus cell/ward of Ayub Teaching hospital, Abbottabad Pakistan",
      "lang_type": "US",
      "file_id": null
    }
  ]
}
```

Fig. 3.7.2 Example input provided

```
{
  "result": [
    {
      "original_sentence": "To find the effectiveness of hydroxychloroquine alone and adjuvant with azithromycin in mild to severe Covide-19 pneumonia patients admitted to Coronavirus cell/ward of Ayub Teaching hospital, Abbottabad Pakistan",
      "processed_sentence": "To find the effectiveness of hydroxychloroquine alone and adjuvant with azithromycin in mild to severe Covide-19 pneumonia patients admitted to Coronavirus cell/ward of Ayub Teaching hospital, Abbottabad Pakistan",
      "output_sentence": "To determine the effectiveness of hydroxychloroquine alone and in combination with azithromycin on mild to severe COVID-19 pneumonia patients admitted to the Coronavirus unit at Ayub Teaching Hospital in Abbottabad, Pakistan.",
      "lang_type": "US",
      "file_id": null
    }
  ],
  "status": true
}
```

Fig. 3.7.2 Output of the example

3.8: SCRIPTING OF 3 MAJOR PROJECTS IN TRINKA – AI

Trinka is a pioneer writing assistant made for academic and formal writing. Leveraging cutting-edge AI, computational linguistics, and NLP technologies, it helps students, professionals, and academicians across the globe write better and publish more confidently.

During the last few months of my internship, I had the opportunity to work on three major projects for the company. I am excited to share a brief note about my experience, although I must mention that due to restrictions outlined in my offer letter, I am not allowed to disclose detailed information regarding the projects.

- 1. Concept Extraction of Scholarly Documents:** The concept extraction project involved working with scholarly documents and utilizing natural language processing techniques to

identify and extract key concepts and ideas. It was a challenging and intellectually stimulating project that allowed me to delve deep into the world of academic literature and learn about various domains and research areas. Through this project, I honed my skills in information retrieval, text mining, and data analysis, which proved to be invaluable for my professional growth.

2. **Subject Area Extractor of Scholarly Documents:** In the subject area extractor project, I was tasked with developing a system that could automatically categorize scholarly documents into specific subject areas. This involved building and fine-tuning machine learning models to accurately classify research papers into relevant domains. The project required a combination of data preprocessing, feature engineering, and model evaluation. Working on this project not only improved my technical expertise but also gave me insights into the challenges and complexities of document classification in the academic context.
3. **Fine-tuning of Whisper Model for English Speech Recognition Deployment:** The final project I worked on was related to fine-tuning the Whisper model for English speech recognition deployment. This cutting-edge endeavor involved working with state-of-the-art deep learning models and speech data to improve the accuracy and performance of the speech recognition system. The project allowed me to gain hands-on experience in fine-tuning large-scale models, dealing with real-world speech datasets, and deploying models for practical applications.

CONCLUSION

In conclusion, my internship experience has been an exceptional learning journey, allowing me to acquire a diverse skill set and knowledge base across various domains. From the outset, I was exposed to a wide range of technologies and tools, enabling me to gain expertise in deep Python programming, AWS services, Linux, Lucene, and Elastic Search. These foundational skills formed the bedrock of my technical proficiency throughout the internship.

Moreover, the projects I worked on provided invaluable opportunities to delve deeper into cutting-edge areas of artificial intelligence and machine learning. I had the privilege of exploring and implementing deep learning algorithms, enhancing my understanding of neural networks and their applications. Navigating through these complex algorithms and models taught me problem-solving skills and creative thinking, essential traits in the field of AI.

One of the most exciting aspects of the internship was delving into natural language processing. The exposure to both basic and neural language processing techniques allowed me to analyze and comprehend textual data, a skill that is increasingly critical in today's data-driven world.

Working on the concept extraction and subject area extractor projects introduced me to the fascinating realm of academic literature analysis. The projects challenged me to design robust systems capable of processing vast amounts of scholarly documents, and through this experience, I honed my abilities in information retrieval and data analysis.

The fine-tuning of the Whisper model for English speech recognition deployment was a defining moment during my internship. Tackling this advanced project exposed me to state-of-the-art deep learning models and the complexities of working with real-world speech datasets. The practical deployment of the model offered a hands-on experience that prepared me for real-world applications of AI in speech recognition.

Overall, this internship has been a transformative period, empowering me with the technical expertise, problem-solving skills, and domain-specific knowledge required to excel in the field of artificial intelligence. The exposure to diverse projects and cutting-edge technologies has fueled my passion for AI, and I am eager to apply these newfound skills to future endeavors in the industry. I am grateful for the invaluable guidance and mentorship received from the team, making this internship a truly enriching and rewarding experience.

REFERENCES

- [1] Richard Blum, Christine Bresnahan Linux Command Line and Shell Scripting Bible, Wiley, year of publication, 2015
- [2] Luciano Ramalho, Fluent Python, Clear, Concise, and Effective Programming, O’Riley, year of publication, 2015
- [3] Otis Gospodnetic, Erik Hatcher, Lucene In Action, Manning, year of publication, 2005
- [4] Hobson Lane, Cole Howard, Hannes Max Hapke, Forwarded by Dr. Arwen Griffioen Manning, year of publication, 2019
- [5] <https://realpython.com/async-io-python/>
- [6] <https://realpython.com/python-async-features/>
- [7] <https://www.python.org/dev/peps/pep-3156/>
- [8] <https://docs.scrapy.org/en/latest/intro/tutorial.html>
- [9] <https://www.accordbox.com/blog/scrapy-tutorial-series-web-scraping-using-python/>
- [10] <https://coralogix.com/blog/42-elasticsearch-query-examples-hands-on-tutorial/>
- [11] <https://coralogix.com/blog/42-elasticsearch-query-examples-hands-on-tutorial/>
- [12] <https://www.elastic.co/guide/en/elasticsearch/reference/7.7/mapping.html>
- [13] <https://www.elastic.co/guide/en/elasticsearch/reference/7.7/indices.html>
- [14] <https://course.spacy.io/en/>
- [15] <https://derwen.ai/docs/ptr/sample/>
- [16] <https://dumps.wikimedia.org/simplewiki/latest/>
- [17] <https://newsela.com/data>
- [18] <https://github.com/mahnazkoupae/WikiHow-Dataset>
- [19] <https://github.com/roeeaharoni/unsupervised-simplification>
- [20] <https://github.com/coltekin/TURKCORPUS>