MSc in Business Analytics

Skills that matter: A Natural Language Processing approach to Extraction of Job Requirement from Job Ads

Author: Saloni Wadhwa

Supervisor: Shahin Ashkiani

Acknowledgements

.

I am very thankful to my supervisor, Dr. Shahin Ashkiani, for providing guidance and his valuable knowledge throughout this project, but through my MSc degree, as well. Moreover, the biggest thanks to my family and friends for their unlimited support and encouragement during these months.

Using suggestions from a variety of blogs and existing whitepapers, the task was made practicable. In the References section, all of the blogs and articles are cited. I recognise the employment portal, from which I obtained sample job advertisements for the model's training.

The information was solely used for my own study and in accordance with the terms of the website's privacy agreement.

I'd also want to thank the wonderful Python packages Keras, Pandas, Nltk, Spacy, Sklearn, and others. These are fantastic libraries that allowed me to conduct my research.

Table of Contents

Abstract

A dynamic employment market, driven by a variety of variables such as globalization and growing populations, needs constant monitoring. As job postings/ads become more available, the digitalization of the job market has provided researchers with the chance to better analyze employment market demands. Such posts, on the other hand, are posted in unstructured language and require additional processing to determine the essential abilities. Given the massive amount of information contained in job descriptions, it might be difficult for a candidate to assess and extract the essential abilities required.

As a result, current information extraction algorithms can't retrieve contextual elements. As a result, they fall short in extracting domain-specific information entities, which has an impact on relevant skill extraction from job descriptions.

The aim of this project is to extract relevant technical skills and predict new related skills from job descriptions using Natural Language Processing, Machine Learning and Deep Learning techniques. In this study, we systematically reviewed 634 job postings on the topic. We evaluated and classified the prior work aiming to identify the skill bases used for analyzing job market needs; the skill identification methods; extracting domain-specific skills; evaluating and improving the performance of models and comparing the output of various models.

Various tests on data from job listings are used to test the suggested technique. The findings demonstrate that the proposed method enhances data retrieved from job descriptions and can assist users in extracting appropriate domain-specific skills from job advertisements.

Chapter 1: Introduction

1.1. Affect of digitalization on employment market-an overview

The advent of digital transformation and the expansion of the internet has had a significant influence on our lives, affecting everything from how we interact on a daily basis to how we seek for future careers. The digital revolution makes huge volumes of organized and unstructured data, such as videos, pictures, and text, available, paving the way for big data analytics. Entities may obtain a clear and full knowledge of their company when such data is properly and efficiently gathered, processed, and analyzed, resulting in increased efficiency and decreased costs.

As a result of the digitalization of the employment market, there are now a plethora of websites dedicated to recruiting and job posting. Recruiters can post job opportunities directly on websites known as job boards or job portals, which candidates can quickly access. The digital transformation is not only altering employment, from job destruction to job creation, but it is also enabling for improved job market needs to be recognized. This is accomplished by monitoring the massive job advertising that are placed on the internet on a daily basis in an attempt to detect employment market trends.

Therefore, extracting job skill is a subproblem in the information extraction area that focuses on finding specific sections of text in user profiles that may be matched to job posting criteria. Job skills serve as a connection between job applications, user resumes, and company job advertisements. Identifying skills in new job ads is a critical challenge that can bridge this gap and offer a roadmap for job searchers and employers.

1.2. Traditional approach for extracting skills

Extraction of job skills is a difficult task that is frequently handled using traditional approaches such as matching against a taxonomy skills dictionary. This method, however, does not apply to new and emerging talents. Dictionary update is time-consuming and laborious, and it takes domain specialists a long time to discover the proper abilities that correspond to a certain sector. Furthermore, talent extraction needs careful thought; for example, the phrase «Java» might refer to either an Indonesian island or an object-oriented programming language. Furthermore, CVs and JDs frequently contain unstructured content, which may include tables, bulleted lists, and other elements.

While general-purpose search engines have advanced significantly, job search services have progressed more slowly. All of these, as well as other aspects, need research expenditure for automated job search engines to be dependable and enhance their effectiveness. As a result, having an automated framework that can extract skills from JDs and CVs is critical for analyzing and reducing the skills mismatch for job search services. The generated data can be utilized for labour market analysis and in job matching and recommendation systems to better match people to jobs and minimize unemployment.

In this project I have tried out a combination of various machine and deep learning techniques to extract relevant skills from unstructured text of text corpus. I have observed significant improvements in model performance by using CNN model for our problem.

1.3. Aims and objectives of the current project

The high-level primary goal of this project is two fold:

1. To expand on the job skill extraction and analysis work by Sharma (2019) performance of the various Machine Learning and Deep Learning models used to detect the Data analytics job skills on the new data.
2. To conduct extensive research and analysis in order to examine the process followed to extract relevant technical skills for Data Analytics role.

The objectives of the research project include:

1. To conduct a quantitative survey on the Alumni of MSc in Business Analytics from Aston Business School and conduct a research study to identify the skills required from the job title that most of the alumni pursue after completing their postgraduation based on the survey.
2. To extract the relevant data from the unstructured text of Job description.
3. To generate Bag of Words holding information about the common words from Data Analytics job description.
4. Extract the technical skills for Data Analyst role from reed website (online job portal)
5. To identify the relevant technical skills holding semantic information from the words identified through initial simple model.
6. To identify the process to distinguish and label the data as skill vs non-skill.
7. To identify the most appropriate model after conducting comparison between Machine Learning and Deep Learning models which would detect the related technical skills on a new dataset.
8. To discuss the practical implications of objectives 2 to 7.

1.4. Thesis Structure

The rest of the master thesis is organized as follows: In Chapter 2, related work from the area of skills extraction and its analysis are described. The theoretical background used in this work and the proposed approach are explained in Chapter 3 respectively. The result of the proposed method is presented in Chapter 4 followed by conclusions and limitations in chapter 6.

Chapter 2: Literature Review

This section covers the most common approaches for extracting entities from JDs and CVs, such as skills. It examines a variety of industry-standard fetching techniques and tactics, as well as past common study for extracting the skills. This section also identifies the major flaws in the present solutions. Furthermore, material from papers, journals, websites, and books is evaluated and compared to come up with solutions to the research questions. This information sets the groundwork for comprehending, planning, and developing the new model of extracting skills that this project seeks to execute.

2.1. Skills Folksonomy/Taxonomy and Graph-Based Approaches

Methods that rely on the creation of a folksonomy or taxonomy of entities are one of the most prevalent ways for skills extraction. Both phrases refer to controlled dictionaries, although they differ in numerous ways, as shown in Table 1:

Taxonomy	**Folksonomy**
Hierarchical - Parent/child & sibling relationship	Flat - No levels, no order, no explicit relationship
Exclusive - The same item cannot be in few distinct categories	Not Exclusive - An item can be associated with many tags
Top-down - Established by experts	Bottom-up - Created by users

Table 1: The comparison of taxonomy and folksonomy.

- ESCO is one of the skills taxonomy-based extraction systems. It's a bilingual system for categorizing European skills, competencies, qualifications, and occupations. It functions as a dictionary, with the goal of providing semantic interoperability between labour markets, education, and training programmes. Unfortunately, no information on the processes and methodologies used to create the ESCO taxonomies is accessible. This lexicon, on the other hand, is open to the public and can be beneficial in constructing a taxonomy for this study.

- LinkedIn is now the world's leading job search system. LinkedIn's team created a large skills extraction framework in 2014. The framework was built on top of a built folksonomy of skills and expertise, and it served as an inference for a recommender system. The phases in the folksonomy-building system were discovery, disambiguation, and deduplication. The first stage in the discovery process was based on the fact that most users generate a list of comma-

separated talents in their profile's specialty section. This feature was used to retrieve entities that might be utilized as prospective skills.

The goal of the second phase of disambiguation was to eliminate uncertainty in skill phrases that had various meanings depending on context. The disambiguating skills problem was solved via clustering based on co-occurring words. Some abilities were labelled with several senses that belonged to distinct industrial groups. The final stage was to get rid of conceptual overlaps like "Python development" and "Python programming." Researchers used a crowdsourcing technique that included LinkedIn members to tackle this challenge. Users were asked to associate a skill phrase with the most appropriate Wikipedia page from a selection of possibilities. A skills inference component was included in this skills extraction system, which used profile data like business, title, and industry as features. The generated feature set was used to train the Nave Bayes classifier. The following studies, on the other hand, proved the superiority of the graph-based model over the Nave Bayes classifier.

- A new technique to skill taxonomy creation was presented in a recent study by Phuong et al. (2018) This study covered the discovery, disambiguation, and deduplication phases, but with different implementation techniques than LinkedIn. They gathered skill-related material from applicant resumes and job descriptions accessible on the Career Builder website in order to create a taxonomy system. Domain knowledge separated the gathered text data by punctuation marks and cleansed it of noise such as stop words, extra adverbs, and other specified keywords. The cleaning step essentially eliminated terms that have no or very little semantic significance in a created skill taxonomy. The researchers used the Wikipedia API for normalization and deduplication. Validation, which employed the Standard Occupational Classification (SOC) method to validate the provided Wikipedia category tags, was another essential step in constructing a taxonomy of skills.

  The Google Search API was used to address the issue of word sense disambiguation (WSD) in this study. If a skill phrase had many meanings, for example, the algorithm picked the one with the greatest Google Search relevance ranking. Nonetheless, this method exposed the flaw of not taking semantic context into account. As a consequence, the researchers created the Skill Tagging system, which is detailed in Figure 1. 39,000 raw terms were mapped to 26,000 normalized skill entities in this skill taxonomy.

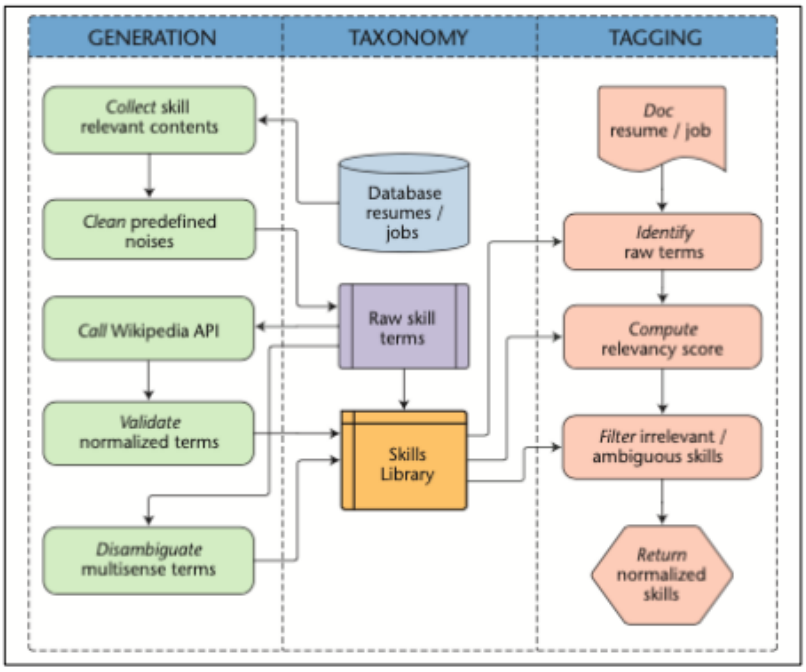


Figure 1 : Architecture of the SKILL System

2.2. Machine Learning Approach

This section covers the most sophisticated Machine Learning (ML) techniques for extracting skills, such as natural language processing (NLP), deep learning networks, and so on.

Sharma (2019) evaluated several ways to extracting relevant abilities from free written material based on unsupervised and self-supervised learning strategies.

The models were trained on a limited dataset of job ads in the Data science area before being expanded to other cross categories. The strategies used in the first two models were as follows:

• Topic modelling is an unsupervised method for retrieving abstract concepts. This approach demonstrated a thorough knowledge of the situation. The retrieved key words, on the other hand, did not match the relevant set of abilities specified in the problem description.

• Word2Vec is a self-supervised neural network that can recognize words in circumstances that are similar. This strategy was used to build on top of Topic modelling. The Word2Vec model was trained using key words extracted using the top modelling approach. When it came to recognizing skills, Word2Vec performed admirably. Word2Vec, on the other hand, extracted a lot of noise, and distinguishing useful skills from noise was a time-consuming task.

Supervised learning is used in the other two models presented in the study. The training dataset was manually labelled and consisted primarily of noun phrases as abilities. Here's a more detailed comparison of the two approaches:

- The first was a straightforward word embedding-based classifier with a convolutional layer that was trained on the labelled data. The job descriptions yielded a lot of valuable skills using this method. The accuracy of the test was 0.6803.

- The second was a mix of word embedding and long short-term memory (LSTM), which increased the skills classifier's accuracy while also extracting a large number of keywords. With a test accuracy of 0.7658, this method produced the best results. In comparison to other models, the presence of noise was also reduced.

By training on a limited dataset, the LSTM and Word embedding models were able to produce acceptable results. However, whereas model training primarily utilized noun phrases, many job advertisements define necessary abilities using verb phrases and other grammatical structures. As a result, the original dataset must be supplemented with a larger number of labelled instances, and only then can a new model be trained.

Gugnani et al. (2020) demonstrated a framework for extracting skills based on multiple natural language processing (NLP) approaches. The framework for extracting skills was made up of four primary submodules:

- Named Entity Recognition (NER) is a technique for identifying keywords and concepts, as well as extracting entities such as people's names, organization's names, places, time expressions, quantities, monetary values, and percentages. Researchers used NER to extract abilities from Job Descriptions and were able to get excellent results. To extract skills as entities, they used Watson NLU 3 services. The collected list of skills was then categorized as a set of "Probable Skills." Word2Vec was used to assign relevance ratings to the processed abilities at this phase.
- The technique of identifying each word in a text as a corresponding part of speech is known as Part of Speech (PoS) Tagger. Five domain experts went through hundreds of JDs by hand, labelling words and phrases as abilities. They observed that abilities differed depending on not just the employment industry, but also the subjective judgement of the individual who labelled them. To identify parts of speech, the JDs data was additionally run via the Stanford Core NLP Parser and PoS Tagging. They developed criteria and patterns based on their observations for identifying prospective or new skill terms that were not included in the skill dictionary or taxonomy. For example, if a phrase has a comma-separated list of nouns, and only a few of the nouns are skills, the other nouns should very certainly be skills. To detect skill-terms, the system was designed using similar rules.
- In a nutshell, Word2Vec (W2V) is used to express words as vectors. W2V uses the input data, which is usually a huge corpus of text, to generate a vector space with several hundred dimensions. In the vector space, each unique word is assigned to a matching vector. Words from frequent contexts are usually clustered together in the vector space. Because the W2V approach tokenizes text with white spaces, a single-word skill is usually straightforward to extract. When the skill is a term like "hard working," "web development," and so on, the issue emerges. Researchers proposed that a skill phrase be represented by a vector, which is an average of the various vectors that make up the skill phase. In addition, a skill dictionary was utilized to compare every possible skill-term in the embedded W2V space. Users' feedback mechanisms were also used by the researchers to acquire new skills and enhance performance. The model was developed using a text corpus containing 1.1 million JDs from more than 50 distinct categories.
- To designate a word or phrase as a skill, a skill dictionary was required. Researchers used the same technique suggested by Gugnani et al. (2018) to construct this skill dictionary. Public resources such as Onet, Hope, and Wikipedia were used to gather

skills. The words were then verified by a group of three specialists. In all, 53,293 soft and hard skills in various categories were included in the skill dictionary.

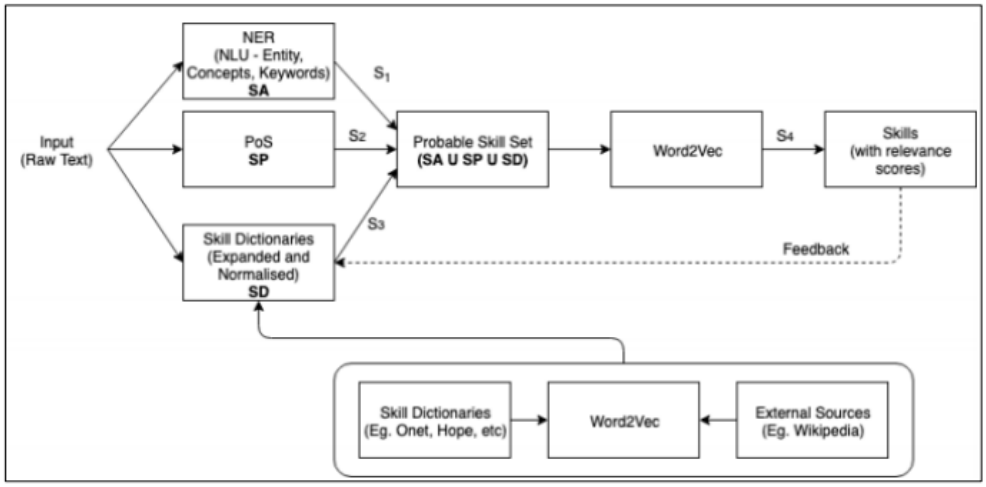


Figure 2: Skill identification Flow Diagram

Figure 2 depicts the modules of the skill extraction system. The first three modules take a raw text as input and extract a group of phrases from it, assigning them a module-specific "score." Furthermore, the sum of the ratings indicates how probable a phrase is to
be a skill.

2.3. Limitations

There are a variety of methods for extracting abilities from a free written text, but this thesis focuses on the most sophisticated and successful ones. This thesis continues the development of this issue toward a specified emphasis while considering meeting the research objectives.

Firstly, in order to overcome deduplication and disambiguation difficulties, a team of specialists and users must be involved in the development of the folksonomy/taxonomy. Unfortunately, it was not feasible to enlist the help of other persons for this thesis.

Furthermore, the investigated ML methods are based on supervised learning, which necessitates the labelling of the dataset for the training model, which takes vast domain-expertise, time and additional resources. As a consequence, the created model should be trained partly based on the experience and intuition and produce acceptable results.

On top of everything else, the availability of computational resources is critical. In reality, the analysis of large amounts of textual data and advanced machine learning algorithms like LSTM and CNN sometimes necessitate the use of powerful GPUs or multicore CPU computers. As a result, one laptop's performance may be insufficient and time-consuming to train and evaluate the built skills extraction model.

2.4. Summary

Based on the limitations stated, this section examines several techniques for skill extraction and determines whether or not they can be utilized for this project and why. The outcomes are shown in Table 2. Different techniques to skill extraction are examined.

Approach name	Conclusion
Skills folksonomy/taxonomy and Graph-Based Approaches	
Bastian et al. LinkedIn, LinkedIn Skills: LargeScale Topic Extraction and Inference	To begin with, the job postings of Reed website consists of unstructured texts within which skills need to extracted from JDs, for which no template for extracting entities can be created. Second, to reduce semantic duplication, the model employed a crowd-sourced technique which was not feasible to study dure insufficient resources.
Phuong et al. Large-Scale Occupational Skills Normalization for Online Recruitment	Because it ignores the semantic background, this method has an obvious flaw. Furthermore, this technique is difficult to apply, because creating a taxonomy requires a significant amount of human resources.
Machine Learning Approach	
Sharma (2019) Job Skills extraction with LSTM and Word Embeddings	The method, which combines word embedding and LSTM techniques, produced the greatest results in terms of extracting skills from unstructured texts. This technique is also reasonably simple to utilize and might be employed in this research.
Gugnani et al. Implicit Skills Extraction Using Document Embedding and Its Use in Job Recommendation [9]	One of the modules uses natural language processing (NER) to find keywords and extract items as talents. This method might be beneficial in this master's thesis.

Table 2. Analysis of different approaches to skill extraction

The model proposed by Sharma (2019), which combines Word embedding and LSTM, is the most suited method for this research. The suggested methodology enables a neural network to be trained on a minimal dataset and achieve good results. A similar strategy was created in this master's thesis, however the NLP component uses a more modern language processing technology called CNN. The challenge of extracting skills from unstructured text may also be considered a NER problem. Gugnani et al. (2020) employed this paradigm in their research.

Chapter 3: Research Methodology

This section is the research's blueprint, and it seeks to address a research topic.

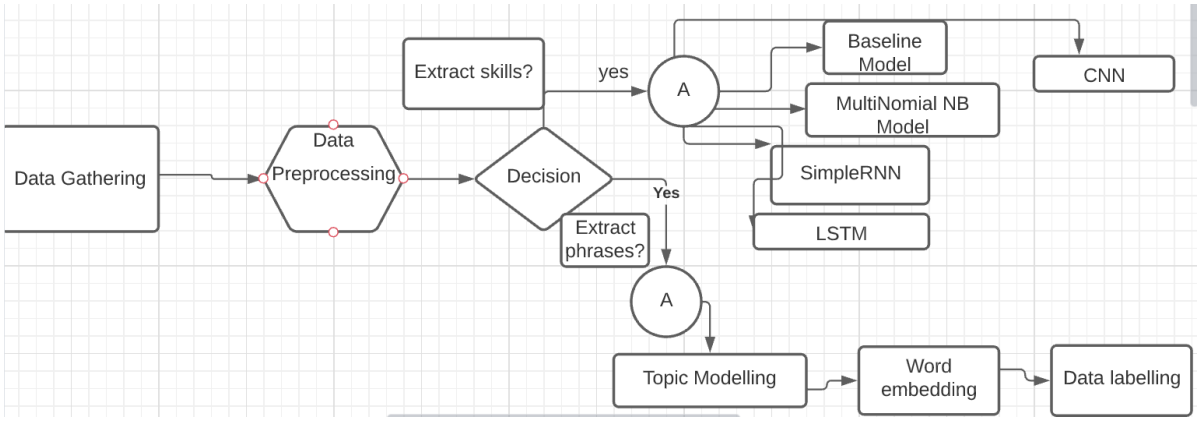


Figure A: Depicting high-level overview of the methodology

3.1. Gathering Information through online survey

A qualitative method in the form of conducting online survey through Survey Monkey was used to gather the information which was helpful to define the scope of the research question. 91 graduates who completed their postgraduation in the year 2018 to 2020 pursuing MSc in Business Analytics from Aston Business School were contacted via email and they were requested to participate in the survey.

A questionnaire was designed for the survey which consisted of the following questions:

Q1. Name

Q2. Where did you get your first graduate role after completing MSc in Business Analytics?

Q3. What was the job title of your first graduate role?

Q4. In which year did you graduate after completion of the Masters's degree?

For Q2, the options provided for the graduates to select from were – United Kingdom and Overseas while the options provided for Question 4 was 2018, 2019 and 2020.

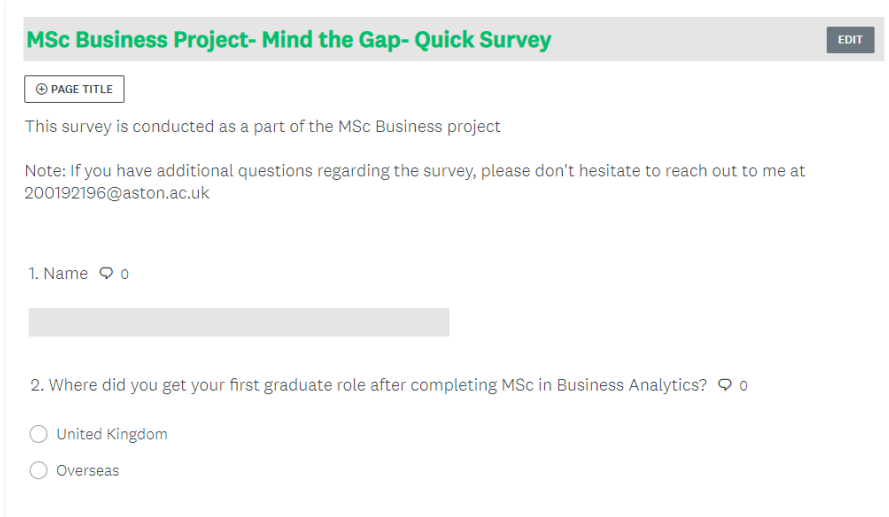


Figure 3: Survey screenshot 1

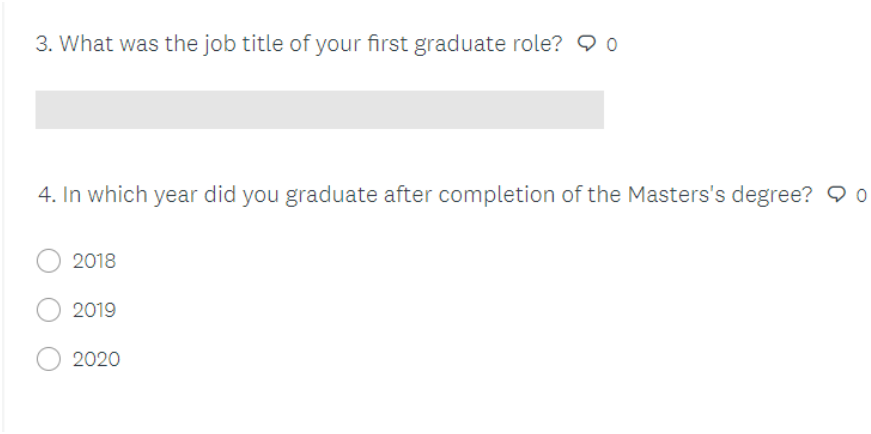


Figure 4: Survey screenshot 2

Figure 3 and 4 are the screenshots obtained for the survey conducted on Survey Monkey website

Based on the survey conducted, as per the invitation sent to 91 postgraduates only 13 students participated in the survey.

The following results were collected on the basis of the responses:

- Figure 5 depicts that 61.54% students(8 out of 13 students) got their first graduate role in Overseas while the rest i.e 38.46% students got their first graduate role in the United Kingdom.

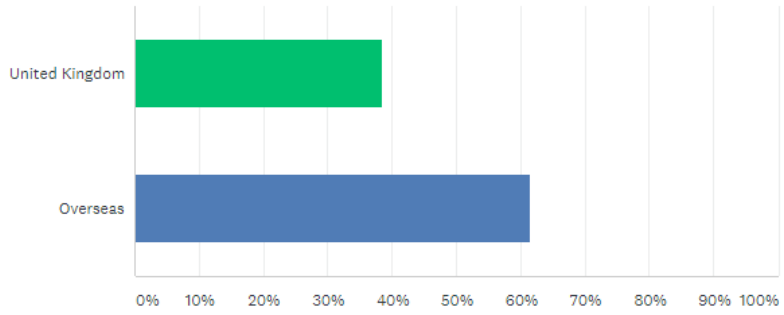
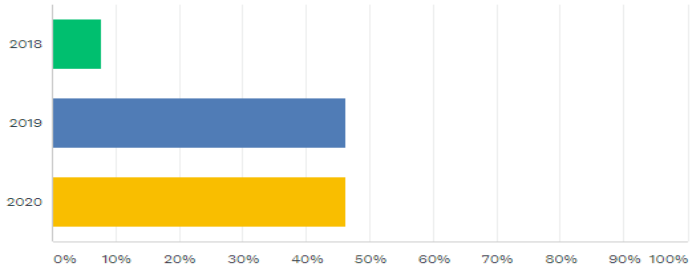


Figure 5: Graduate role in UK vs Overseas

- Figure 6 depicts that equal percentage of the students i.e 46.15% (6 number of students each) who graduated in the year 2019 and 2020 participated in the survey while only 1 student who graduated in 2018 participated in the survey.



ANSWER CHOICES	RESPONSES	
2018	7.69%	1
2019	46.15%	6
2020	46.15%	6
TOTAL		13

Figure 6: Number of students graduating 2018-2020

- Around 31% (highest percentage) of the postgraduates worked as a Data Analyst as their first gradate role after completion of their postgraduation studies.

3.2 Materials & Dataset

3.2.1. Software used

❖ **Python**- It is the main software used in this project for creating machine learning algorithms, data visualization, data manipulation and analysis, and other data-related tasks. It also features a variety of libraries, such as TensorFlow and Keras, that allow programmers to develop data analysis and machine learning applications more rapidly and efficiently. Moreover, programming is done in version 3.7.12 for this project.

Python's popularity stems from the following factors:

- It features a straightforward syntax that resembles normal English, making it easier to read and comprehend. This speeds up the development of projects as well as the improvement of existing ones.
- It's adaptable. Python can be used for a variety of purposes, including web development and machine learning.
- It's user-friendly, making it popular among new programmers.
- It's free to use and distribute, even for commercial purposes, because it's open source.
- Python's library of modules and libraries—code bundles generated by third-party users to extend Python's capabilities—is large and increasing.
- Python has a vibrant community that contributes to the library of modules and libraries and serves as a resource for other programmers.

❖ **Google Colaboratory** - This project makes use of Colab notebooks, which are cloud-based Jupyter notebooks that are tightly connected with Google Drive, making them simple to set up, access, and share. Python code was created and executed in the browser using Colab, which is particularly well suited to machine learning, deep learning, and data analysis tasks. Colab notebooks run code on Google's cloud servers, allowing to take advantage of Google hardware, such as GPUs and TPUs, regardless of the machine's capabilities.

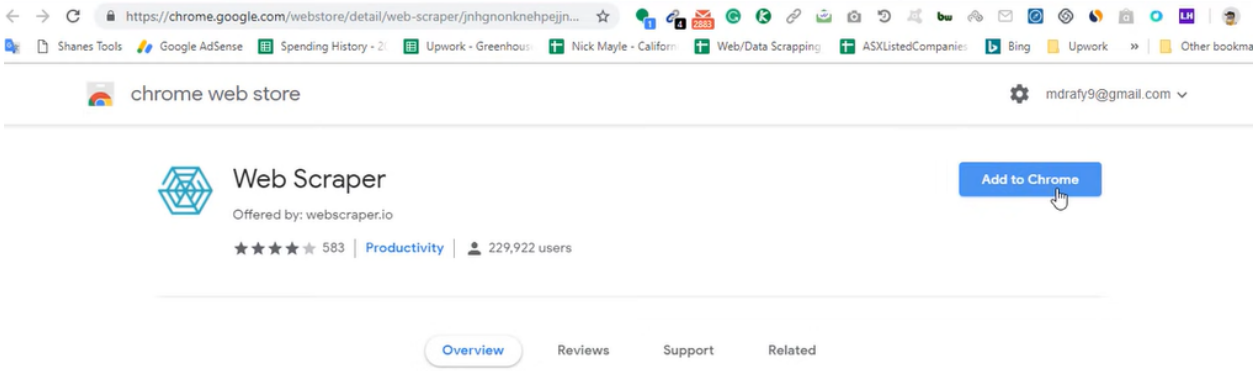
- **Web Scraper –** This chrome extraction was utilized for data gathering since it is a web data extraction tool with a simple point-and-click interface that allows anyone to extract thousands of information from a website in just a few minutes. Web Scraper employs a selector-based structure that instructs the scraper on how to navigate the target site and what data to gather. Furthermore, the data extraction is done through the browser and does not require any software to be installed on the machine.

3.2.2. Source of dataset

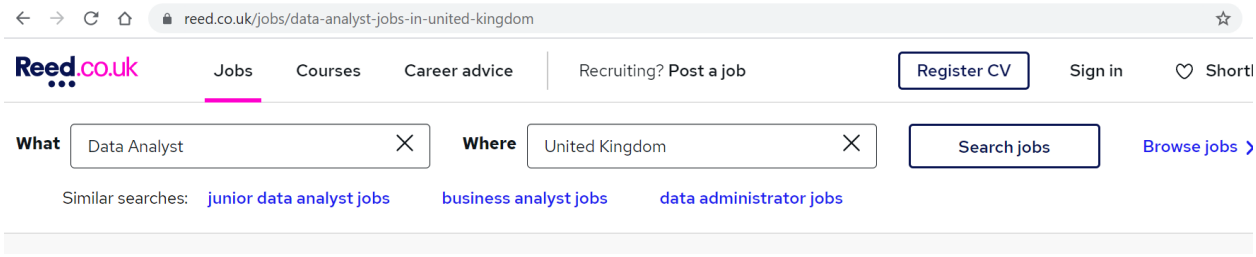
On the basis of the survey, the job postings were collected for Data Analyst role with the help of Web scrapper chrome extension from Reed website which is UK's leading job site connecting people to a world of opportunities online. This extension was used in order to scrape the HTML text from the web page into csv file which would be later used in the analysis.

3.2.3. Detailed process for Data Collection

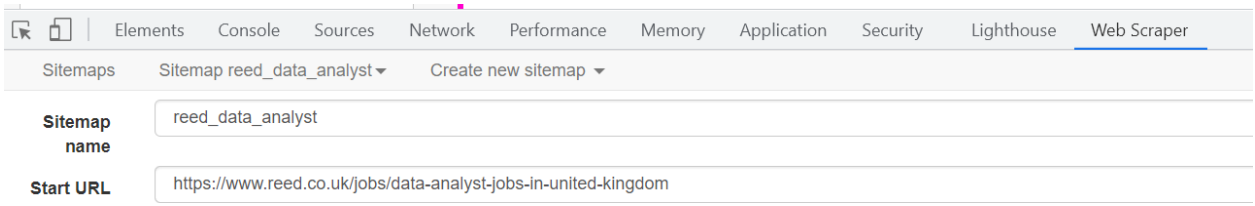
- Add Web scrapper from Google chrome extension



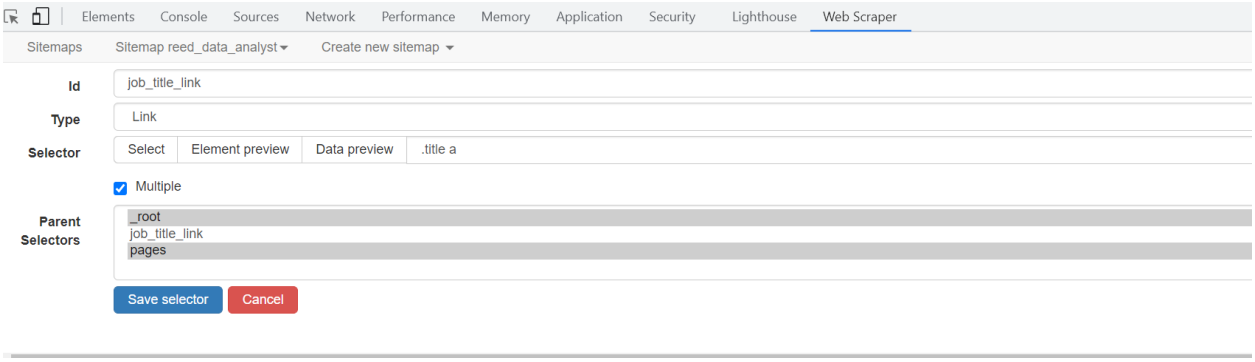
- Open Reed website and type the job title and the location.



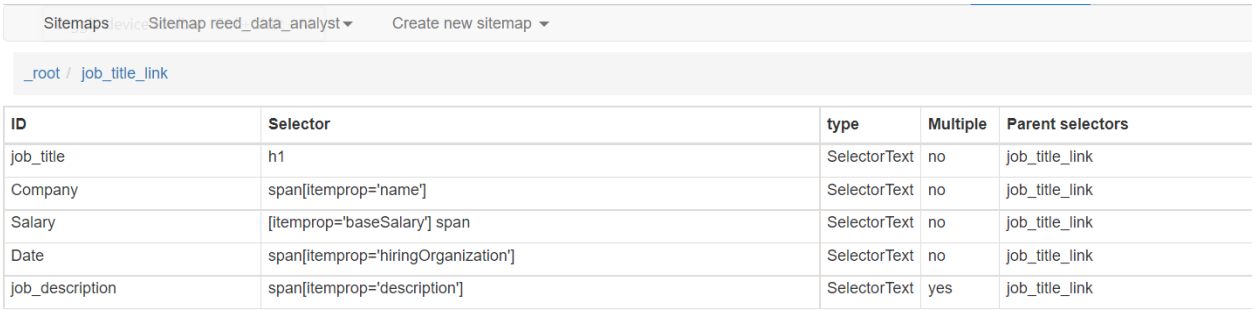
- Inspect anywhere from the webpage and create the sitemap through web extension by mentioning the sitemap name and home url of the webpage from where the information needs to be scrapped.



- Choose multiple option while creating the selector and select multiple job titles from the home page. Select type as link because clicking on the it would open a new webpage containing the detailed information about the particular job posting.

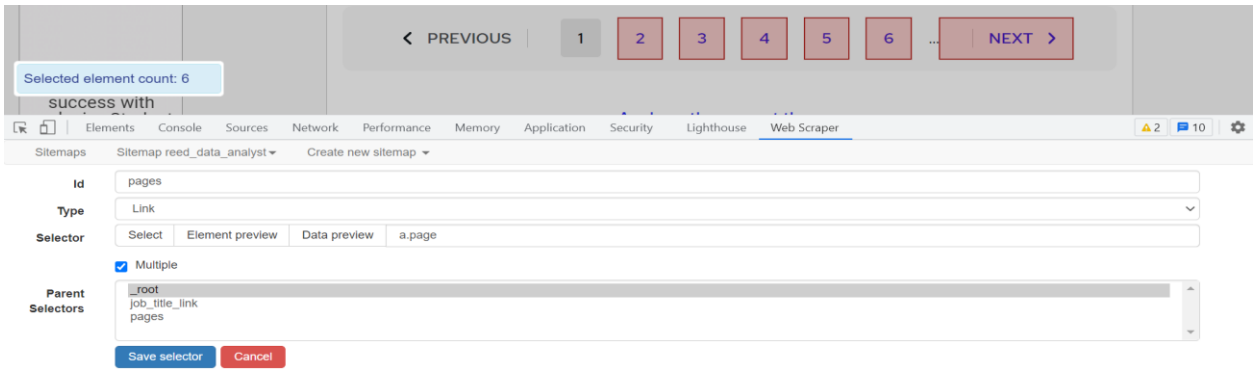


- Open the saved selector and create multiple selectors under it from the detailed job description page. Inspect the text where we can find the job title, company name, Salary, date and Job description.

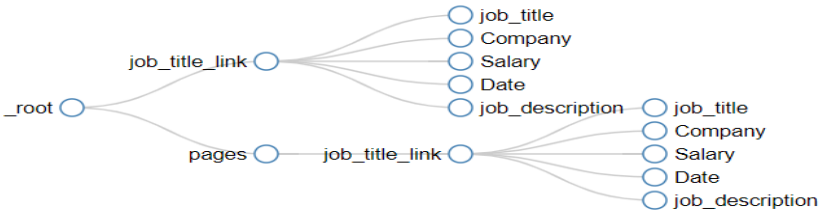


ID	Selector	type	Multiple	Parent selectors
job_title	h1	SelectorText	no	job_title_link
Company	span[itemprop='name']	SelectorText	no	job_title_link
Salary	[itemprop='baseSalary'] span	SelectorText	no	job_title_link
Date	span[itemprop='hiringOrganization']	SelectorText	no	job_title_link
job_description	span[itemprop='description']	SelectorText	yes	job_title_link

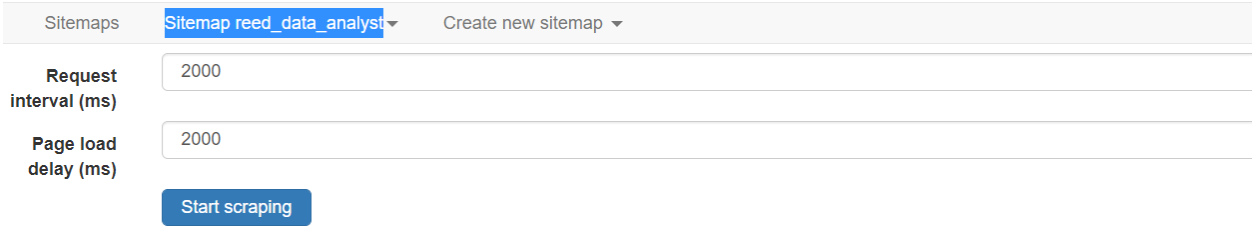
- After creating multiple selectors, we need to create the last selector which is to navigate to all the selected pages and extract the data in a loop from the root or main location of the sitemap.



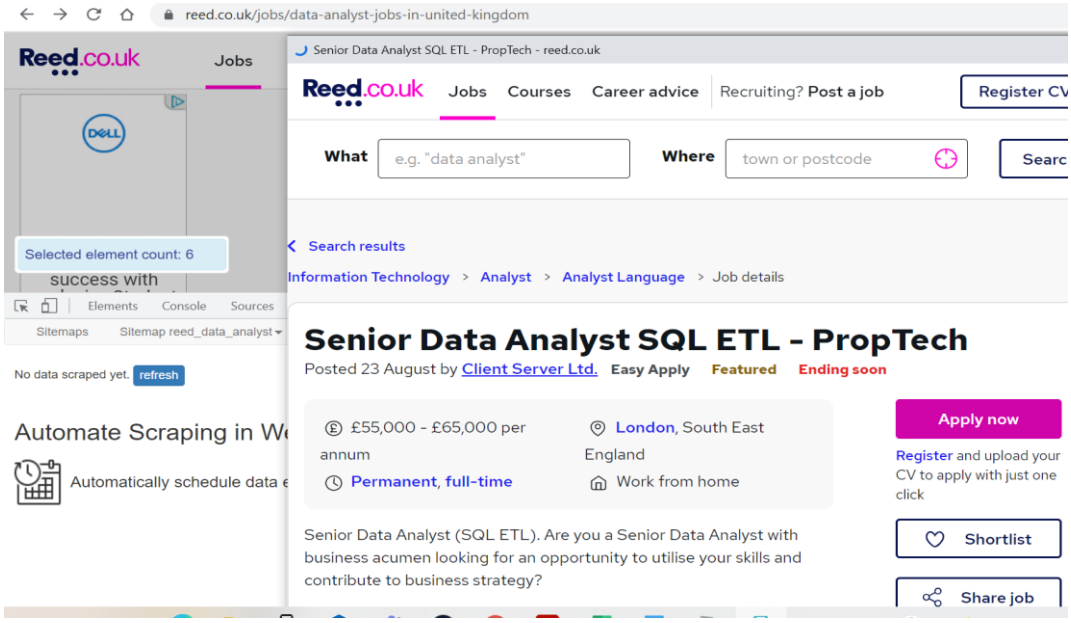
- In order to check the selector graph, navigate to Sitemap Name-> Selector graph



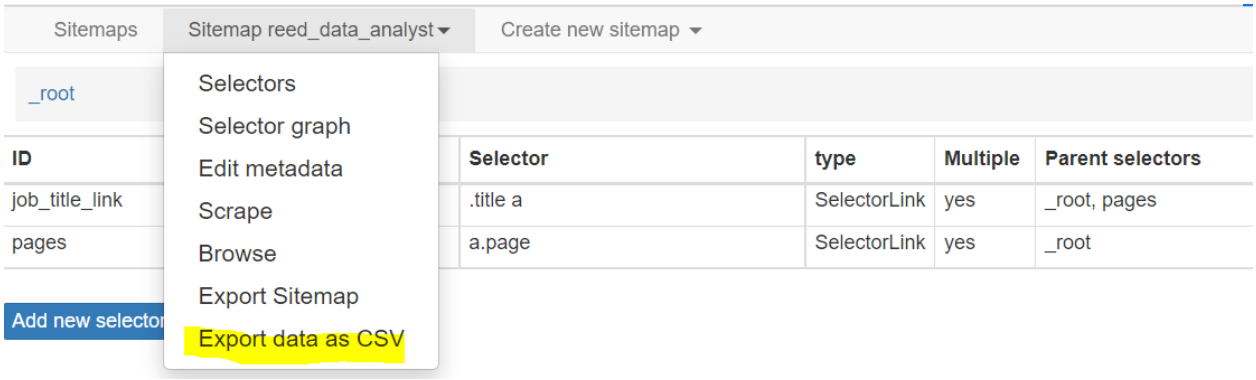
- Once the selector graph seems to be correct, then click on scrape from the dropdown of main selector name and then click on **start scraping** button.



- A new pop-up window will appear which would navigate through all the web pages containing the detailed job description and scrape the data.



- The pop-up window will close once the scrapping of the job data is done. The final step would be to select export data as CSV from the Sitemap name



3.2.4 Dataset Understanding

From the obtained CSV file, 634 number of records were extracted from the webscrapper tool.

The records obtained consisted of 9 different columns which are explained as follows:

1. Web scrapper order- The id to distinguish the job posting
2. Job title link href- The link of the detailed job posting webpage.
3. Job title- The title of the job posting
4. Company- The name of the company for respective job posting.
5. Salary – The salary of the job
6. Date- the date when the job was posted on the website.
7. Job Description- The description of the job in detail.
8. Pages- the page number where the job posting was present in the Index page.
9. Pages-href- The link of the index of the page where the job was posted.

web-scraper-order	job_title_link-href	job_title	Company	Salary	Date	job_description	pages	pages-href
1628454831-1436	https://www.reed.co.uk/jobs/data-qu	Data Quality Analyst	Profiles Creative	£36,000 - £38,000 per annum	Posted 20 July by	A leading online retail & groceries brand is undergoing consider		
1628455695-1752	https://www.reed.co.uk/jobs/data-an	Data Analyst	Profectus Recruitment	£30,000 - £40,000 per annum	Posted 13 July by	Commercial Data AnalystLon	2	https://www.reed.c
1628455409-1639	https://www.reed.co.uk/jobs/senior-d	Senior Data Analyst	Data Idols	Salary negotiable	Posted 28 June by	Senior Data Analyst Data Idols are working with one of the larg		
1628455063-1527	https://www.reed.co.uk/jobs/market-	Market Data Analyst	James Frank Associates	£25,000 per annum	Posted 5 July by	Our client, a well-established and hugely-successful business is		
1628454193-1193	https://www.reed.co.uk/jobs/custome	Customer Service Data Analyst	Focus Resourcing	Salary negotiable	Posted 30 July by	4-6 month contract to cover sabbatical Working for our highly		
1628455483-1667	https://www.reed.co.uk/jobs/data-an	Data Analyst	Pontoon	Salary negotiable	Posted 3 days ago by	My Corporate Travel Manage	5	https://www.reed.c
1628454727-1398	https://www.reed.co.uk/jobs/data-an	Data Analyst - Fintech	Davies Resourcing	£35,000 - £50,000 per annum	Posted 5 July by	Our client is a multi-award winning FinTech company on a missi		
1628455109-1545	https://www.reed.co.uk/jobs/crm-dat	CRM Data Analyst	Experis LTD	£37,000 - £42,000 per annum	Posted 16 July by	CRM Data Analyst Job Category: MarketingJob Level: Specialist,		
1628455299-1595	https://www.reed.co.uk/jobs/marketi	Marketing Data Analyst	Experis LTD	£37,000 - £42,000 per annum	Posted 6 days ago by	CRM Data Analyst Job Category: MarketingJob Level: Specialist,		
1628455675-1745	https://www.reed.co.uk/jobs/data-an	Data Analyst	Harnham	£30,000 - £40,000 per annum	Posted 26 July by	Data AnalystLeading football	3	https://www.reed.c
1628455037-1517	https://www.reed.co.uk/jobs/data-qu	Data Quality Analyst	Proactive Appointments	£40,000 - £45,000 per annum	Posted 6 July by	Data Quality Analyst Our client who are one of the longest stan		

As per the research objective, we need to extract the relevant skills from the job description which is unstructured text, so we will consider only job description column out of the 9 columns for our analysis. Also, appropriate methods were implemented in order to clean the text and convey sensible information.

3.3 Data Preprocessing and simplified representation

First and foremost step for Data Preprocessing involves to load the data which was done uploading the obtained dataset in Google Colab and displaying some the records as shown in Figure 7

	web-scraper-order	web-scraper-start-url	job_title_link	job_title_link-href	job_title	Company	Salary	Date	job_description	pages	pages-href
0	1628454831-1436	https://www.reed.co.uk/jobs/data-analyst-jobs-...	Data Quality Analyst	https://www.reed.co.uk/jobs/data-quality-analy...	Data Quality Analyst	Profiles Creative	£36,000 - £38,000 per annum, negotiable, inc b...	Posted 20 July by \n\n ...	A leading online retail & groceries brand is u...	NaN	NaN
1	1628455695-1752	https://www.reed.co.uk/jobs/data-analyst-jobs-...	Data Analyst	https://www.reed.co.uk/jobs/data-analyst/43351...	Data Analyst	Profectus Recruitment	£30,000 - £40,000 per annum	Posted 13 July by \n\n ...	Commercial Data Analyst\nLondon - Part Remote ...	2.0	https://www.reed.co.uk/jobs/data-analyst-jobs-...
2	1628455409-1639	https://www.reed.co.uk/jobs/data-analyst-jobs-...	Senior Data Analyst	https://www.reed.co.uk/jobs/senior-data-analys...	Senior Data Analyst	Data Idols	Salary negotiable	Posted 28 June by \n\n ...	Senior Data Analyst Data Idols are working wi...	NaN	NaN
3	1628455063-1527	https://www.reed.co.uk/jobs/data-analyst-jobs-...	Market Data Analyst	https://www.reed.co.uk/jobs/market-data-analys...	Market Data Analyst	James Frank Associates	£25,000 per annum	Posted 5 July by \n\n J...	Our client, a well-established and hugely-succ...	NaN	NaN
4	1628454193-1193	https://www.reed.co.uk/jobs/data-analyst-jobs-...	Customer Service Data Analyst	https://www.reed.co.uk/jobs/customer-service-d...	Customer Service Data Analyst	Focus Resourcing	Salary negotiable	Posted 30 July by \n\n ...	4-6 month contract to cover sabbatical Workin...	NaN	NaN

Figure 7: Displaying the records

After loading the data, we need to import required libraries in python. Figure 8 is the output on inspecting the first record of the job description

```
df['job_description'][0]
```

```
'A leading online retail & groceries brand is undergoing considerable investment in their Insight & Analytics teams across the business. \n\nThis is a fantastic opportunity to join a data-driven e-commerce business at the forefront o
f technological innovation. Working in an entrepreneurial start-up environment with the backing of a huge brand.\n\nThe Role:\n\n*The main purpose of the role is to Drive Data quality improvement across the business.\n*Support the da
ta governance programme (and associated data governance council) through detailed data analysis.\n*Builds tools and dashboards to monitor quality and provide alerts to all data users.\n*Manage, monitor, assess, resolve data quality i
ssues with EDW, Data Marts and reporting solutions.\n*Answer business questions about data, be the bridge between data and Insight.\n*Assess and profile new data sources to provide input to integration design.\n*Collaborate with busi
ness users and analysts to capture business rules and data quality rules.\n*Maintain enterprise data model in line with data quality deliverables.\n\nThe Candidate:\n\n*Minimum 2 years experience working as a Data Quality Analyst.\n*
Experience of creating and implementing data quality dashboards and monitoring techniques.\n*Experience of data quality profiling, and creating data quality rules agreed with the business stakeholders.\n*Good Data Modelling experienc
e.\n*Good SQL and/or PL/SQL experience.\n*MS Excel/GSheets at an advanced level.\n*SQL, with table design and creation as a minimum standard, with the ability to manipulate fields, insert, delete and move data.\n*Knowledge of Data in
tegration, reconfiguration and manipulation\n*Experience working with cloud platform - especially GCP\n\nProfiles Creative is acting as an Employment Agency in relation to this vacancy.<'
```

Figure 8: Inspecting the first record of the job description

From the output, it is evident that the data is not clean as it contains unwanted characters like *, \n, etc., numbers and contain different case types (wherein the first letter of the word is generally capitalized) so data-preprocessing steps needs to be done in order to structure the data.

The data pre-processing involves the following steps:

1. Iterating through each of the record of the job description.
2. Breaking down the obtained job description into sentences through sentence tokenizer
3. Iterating through each of the sentences and breaking it down into list of words through word tokenizer
4. Looping through the list of words and converting into lowercase
5. Adding to the array if the words from step 4 are not stop_words(explained later) and if the obtained word contains all the characters as alphabet characters(a-z)

Introduction to Stop words

In data pre-processing steps, we would be removing the stop words as these are the most common words in any language (articles, prepositions, pronouns, conjunctions, and so on), and they don't add anything to the text. Stop words in English include "the," "a," "an," "so," and "what.".

Importance of removing stop words

In any human language, there are plenty of stop words. We remove the low-level information from our text by deleting these terms, allowing us to focus more on the crucial information. In other words, we can say that removing such phrases has no negative impact on the model we are training for our task.

Because there are fewer tokens involved in the training, removing stop words reduces the dataset size and hence reduces training time.

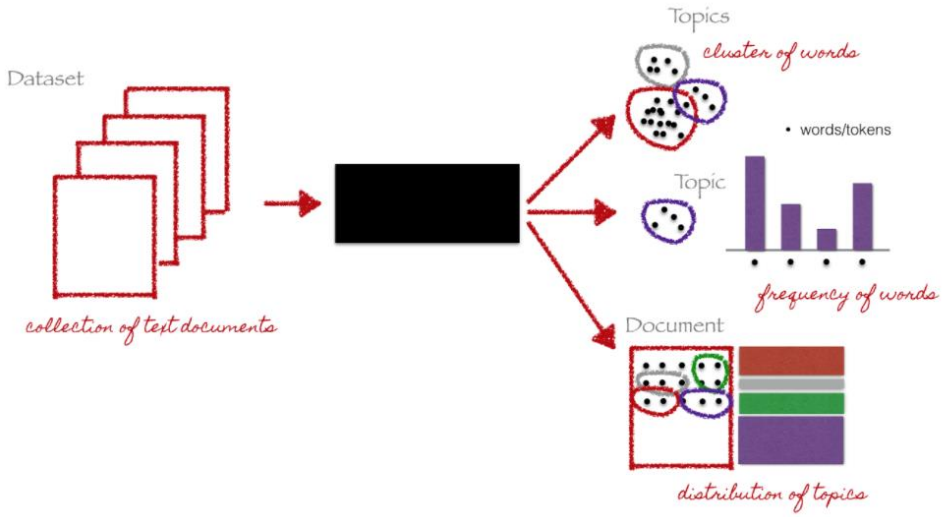
Once we got the resultant array of words after the data-preprocessing steps, we have generated a word cloud to get a high-level of understanding of the words which might play an important role in our analysis.

From the word cloud, it is evident that the words data, analyst, business, client, team, project etc. are few of the words which occur frequently in all the records of the job description. Also, bigger the size of the word shown in the word cloud, it has more frequency of occurrence in the records.

3.4 Implementing Methodology

Implementing models to Identify phrases from Job Ads

3.4.1.1 Topic Modelling



Through Topic modelling, we were able identify hidden subjects or the abstract topics from enormous amounts of text. The Latent Dirichlet Allocation (LDA) technique is a common topic modelling algorithm that has great implementations in Python's Gensim package. The problem is determining how to extract high-quality themes that are distinct, and significant. The method to identify the hidden subject largely depends on the text preparation quality and the approach for determining the ideal number of subjects.

Latent Dirichlet Allocation (LDA) algorithm: It takes a proportional approach to topic modelling, treating each document as a collection of topics. And each topic as a collection of keywords, proportionately distributed. Once we give the algorithm the number of topics, it simply rearranges the topic distribution within the documents and the keyword distribution inside the topics to achieve a desirable topic-keywords distribution composition.

Steps to generate Topics and Features

1. ***Converting text to numbers through Bag of Words model***-

The raw text is incomprehensible to machines, unlike humans. Numbers are all machines can see. Machine learning, for example, is a statistical technique that can only deal with numbers. As a result, we must transform our words into numerical values.

To translate text into numerical representation, there are a variety of methods. Two of the most widely utilized approaches are the Bag of Words Model and the Word Embedding Model. To transform our text to numbers in this article, we'll use the bag of words concept.

The following script uses the bag of words model to convert text documents into corresponding numerical features:

```
from sklearn.feature_extraction.text import CountVectorizer
cv= CountVectorizer(max_df=0.75,min_df=2,stop_words='english')
dtm=cv.fit_transform(df['job_description'])
dtm
```

The script above uses CountVectorizer class. We have passed three parameters to the class i.e max_df, min_df and stop_words.

- **Min_df**:
  The first parameter, min_df has been set to 2. This is the minimum number of documents that should include this feature. As a result, we only list words that appear in at least two documents.

- **Max_df:**
  The value of the second parameter, max_df, is set to 0.75, where the fraction equates to a percentage. In this case, 0.75 suggests that we should only consider terms that appear in a maximum of 75% of all documents. Words that appear in nearly every document are usually unsuitable for classification since they provide no distinctive information about the document.

- **Stop_words**
  The third parameter i.e stop_words ='english' is designated to remove the stop words in English language. The explanation of this parameter has been explained in the data-preprocessing steps.

```
<634x4821 sparse matrix of type '<class 'numpy.int64'>'
        with 81688 stored elements in Compressed Sparse Row format>
```

Figure 9: Sparse Matrix

After apply Countervectorizer class, we would get an output of sparse matrix (Figure 9) with 634 number of rows and 4821 number of columns(after getting converted into numerical representation). This means from each record from 634 records, in total 4821 features would be generated

2. Applying Latent Dirichlet Allocation(LDA) algorithm to the Bag of Words model-

```
from sklearn.decomposition import LatentDirichletAllocation
LDA=LatentDirichletAllocation(n_components=10,random_state=42)
LDA.fit(dtm)
```

Figure 9: LDA model

- After performing step 1, we select the number of topics = 10 and set the random state as 42(Figure 9)
- Apply the model obtained from Step 1 to LDA function.

```
LatentDirichletAllocation(batch_size=128, doc_topic_prior=None,
                          evaluate_every=-1, learning_decay=0.7,
                          learning_method='batch', learning_offset=10.0,
                          max_doc_update_iter=100, max_iter=10,
                          mean_change_tol=0.001, n_components=10, n_jobs=None,
                          perp_tol=0.1, random_state=42, topic_word_prior=None,
                          total_samples=1000000.0, verbose=0)
```

Figure 10: LDA model depicting the parameters for training

After running the mentioned LDA model, we get the information about some of the supplied and default parameters that would influence the output of the model. The important parameters to take in account would be learning_method='batch', batch_size i.e number of documents/records in each learning iteration(128 in our case) would be processed in batches of 128 records at a time, max_iter which is the total number of iterations to generate the relevant features and, n_components as described above.

Check the randomized text at a particular feature index

```
cv.get_feature_names()[2550]
```

```
'levelproficiency'
```

Figure 11: Get feature name

Figure 11. shows that at feature number 2550, we obtain the text as levelproficiency.

Generating top 10 features for randomized text holding any value between 0 to 4821

```
discrepancyresolve
mind
promote
syndicate
starts
guidelines
utility
sought
ordinator
eye
```

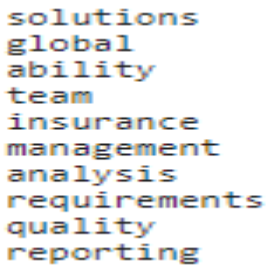
Figure 12: extracting top 10 features

Figure 12 output is obtained when we try to get the value corresponding to the random feature number 10 times

On getting the output of shape of LDA_components, it was observed that there is total 10 records having 4821 features in total.

On inspecting the shape of the first topic, it was observed that it has 4821 features and after inspecting the values, we observe that the feature index it holds are not sorted in order or the order where we can get the features holding much importance to determine the topics.

For example an array a holds value as 10,500,200 which is not sorted numerically, so in such case we need to sort the array in order to get the array in descending order which would be generating the index holding value as 500, 200 and then 10.



On inspecting the features for topic 1, the list of words were obtained as shown in Figure 13. So, in order to get top 10 features for topic 1, it can be claimed that solutions play a major role in depicting the topic 1 while reporting plays least role in defining topic 1.

Figure 13: Top 10 features in Topic 1

Generating top 15 features for the provided number of topics(10), the following results shown in Figure 14 can be obtained

```
THE TOP 10 WORDS FOR TOPIC #0
['reports', 'join', 'sql', 'company', 'governance', 'solutions', 'global', 'ability', 'team', 'insurance', 'management', 'analysis', 'requirements', 'quality', 'reporting']
THE TOP 10 WORDS FOR TOPIC #1
['remote', 'insights', 'll', 'market', 'making', 'opportunity', 'looking', 'new', 'power', 'sql', 'team', 'bi', 'product', 'work', '000']
THE TOP 10 WORDS FOR TOPIC #2
['sql', 'ability', 'people', 'requirements', 'ensure', 'mi', 'using', 'management', 'master', 'team', 'group', 'support', 'analysis', 'work', 'able']
THE TOP 10 WORDS FOR TOPIC #3
['employment', 'excel', 'analysis', 'customer', 'recruitment', 'client', 'company', 'looking', 'management', 'support', 'job', 'team', 'work', 'reporting', 'new']
THE TOP 10 WORDS FOR TOPIC #4
['new', 'processes', 'reports', 'requirements', 'company', 'knowledge', 'client', 'ability', 'excel', 'strong', 'senior', 'work', 'sql', 'analysis', 'team']
THE TOP 10 WORDS FOR TOPIC #5
['warehouse', 'financial', 'quality', 'azure', 'key', 'development', 'right', 'internal', 'management', 'ensure', 'product', 'work', 'delivery', 'project', 'team']
THE TOP 10 WORDS FOR TOPIC #6
['equivalent', 'including', 'analytics', 'analytical', 'key', 'client', 'analysis', 'governance', 'development', 'using', 'level', 'training', 'apprentice', 'work', 'apprenticeship']
THE TOP 10 WORDS FOR TOPIC #7
['required', 'systems', 'colleagues', 'analysis', 'provide', 'people', 'risk', 'ability', 'including', 'team', 'information', 'hr', 'work', 'management', 'support']
THE TOP 10 WORDS FOR TOPIC #8
['sql', 'excel', 'using', 'tools', 'support', 'key', 'work', 'analytical', 'knowledge', 'team', 'management', 'quality', 'reporting', 'information', 'analysis']
THE TOP 10 WORDS FOR TOPIC #9
['client', '000', 'customer', 'team', 'tools', 'apply', 'sql', 'analysis', 'google', 'company', 'insights', 'digital', 'insight', 'analytics', 'marketing']
```

Figure 14: Topic Modelling result showing 10 Topics and 15 features

The above result from Figure 14 could be interpreted as Topic 1 has generated 15 features as reports, join, sql, company, solutions, global, ability, team, insurance, management, analysis, requirements, quality and reporting wherein reports have a major probability or contribution in influencing the topic 1 while reporting holds the least probability in influencing the topic 1.

Similarly, the top 15 features can be interpreted for other 9 topics and also, we can interpret the words which have highest or the lowest probability in determining the respective topic.

The technical skills that can be extracted from the above model(fig 14) is sql, excel, reporting, insights, powerbi, development, azure, warehouse.

A similar LDA model was run with 7 topics and sorting it according to the 10 features in each topic, we observed that the model with 10 topics and 15 features extracted in each topic outperformed the former one as we could discover new technical skills such as azure and warehouse etc. which are more relevant to our research objective whereas in the former model we obtained only one technical skill which is visualization.

Labelling the records of the dataset according to the maximum probability of occurrence of particular topic number out of 10 topics(0-9)

- The LDA model obtained is transformed and we inspect the topic number probability to determine a particular topic for first record

```
topic_results[0]
```

```
array([0.59790254, 0.0007695 , 0.00076941, 0.1468137 , 0.00076948,
       0.00076936, 0.00076944, 0.0007694 , 0.24989769, 0.00076949])
```

Figure 15: LDA model result for first record

- On rounding off the array to 0 decimal places, we get:

```
array([0.6 , 0.  , 0.  , 0.15, 0.  , 0.  , 0.  , 0.  , 0.25, 0.  ])
```

Figure 16: Rounding off to zero decimal places

- Then, we use argmax function , to return the topic number having maximum probability of occurrence for the first record.

```
topic_results[0].argmax()
```

```
0
```

Figure 17: getting topic number having max probability

- This shows that the first record out of 634 number of records, has predominant occurrence of topic number 0
- On applying the transformation on whole LDA model, we obtain the following frequency distribution of topics on overall dataset of 634 number of records

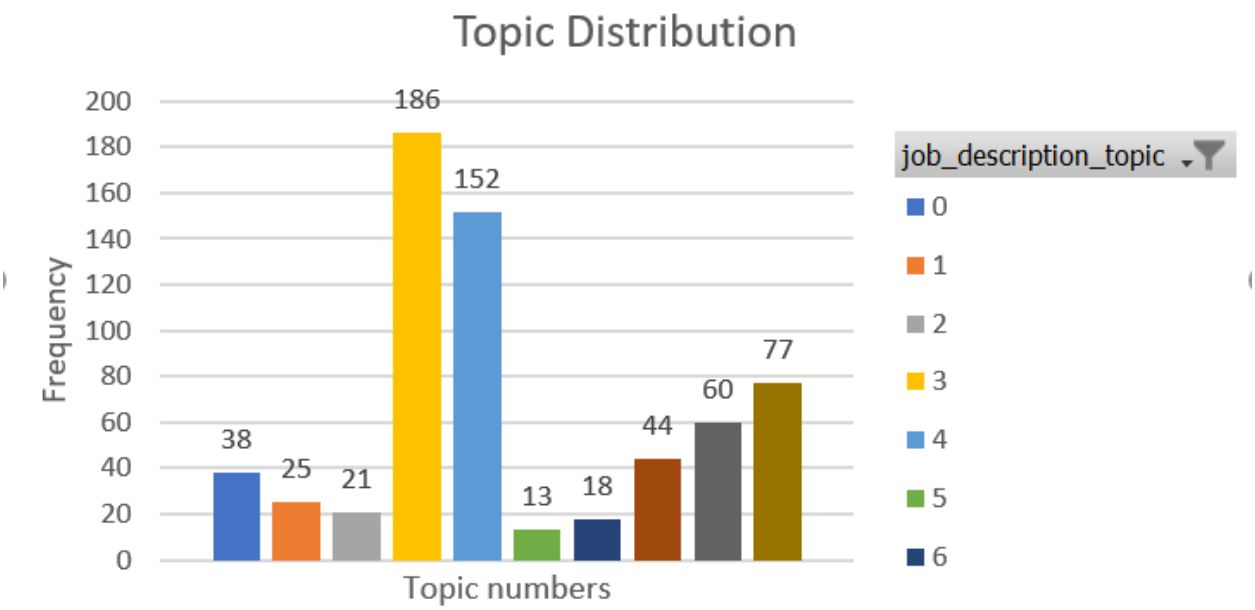


Figure 18: Topic distribution in records

From the above bar chart as shown in Figure 18 , it can be depicted that Topic number 3 has maximum occurrence which is 186 times from total of 634 records. So it can be also inferred that features for this topic that is employment, excel, analysis, customer, recruitment, client, company, looking, management, support, job, team, work, reporting, new are the words which majorly occur in our dataset.

Also, in terms of least occurrence of particular topic in our dataset, we can claim that Topic number 5 holds the least frequency which is 13 from total of 634 records. So it can be also inferred that features for this topic that is warehouse, financial, quality, azure, key, development, right, internal, management, ensure, product, work, delivery, project, team are the words which least occur in our dataset.

Topic modelling correctly recognized the context, as well as relevant terms such as data, insights, reporting, excel, sql, and so on. Topic modelling, on the other hand, did not give the entire collection of relevant skill sets that we were looking for in our problem specification.

3.4.1.2 Word Embedding

It's a self-supervised neural network that can recognize keywords in comparable contexts and extract related skills and keywords for any given collection of terms. The goal is to build on top of the Topic modelling framework.

Topic modelling has been used to extract basic keywords, and Word2Vec has been used to extract all skills and keywords used in the same context.

Theoretical concept overview

One of the most common ways to express document vocabulary is by word embedding. It can recognize a word's context in a document, semantic and grammatical similarity, and relationships with other words.

- **Why Do Computers Fail When It Comes To Word Processing?**

Words are difficult for computers to cope with. In reality, they are unable to do so without the use of workarounds. Natural language is just beyond the capabilities of computers. They are unable to comprehend concepts such as word definitions, adjectives, and synonyms.

They can, however, comprehend numbers. Computers are designed to do just that: compute. They are swift and efficient in their computations. As a result, in order for a computer to understand natural language, words must be translated into a numerical representation. The computer would be unable to cope with words if this were not the case. Words can be converted to numbers in a variety of ways, including one-hot encodings and embeddings.

A distributed representation of a word is utilized in word2vec. Consider a vector with hundreds of dimensions (say 1000). Each word is represented by a weighted distribution of those components. As a result, rather of a one-to-one mapping between a vector element and a word, the representation of a word is distributed throughout all vector elements, and each vector element contributes to the definition of many words.

For instance, the Figure 19 can be represented on naming the dimensions in a hypothetical word vector:

Figure 19: Naming the dimensions in Word vector

From Figure 19, we can infer that the results that are obtained seems to be reasonable. This can be proved as the word “Royalty” is more related to King, Queen and Princess while it is least related to Woman. Similarly, the word “Masculinity” is most related to word King as compared with Quenn, Woman and Princess which makes sense.

In some abstract sense, such a vector comes to represent the 'meaning' of a word. It is feasible to learn word vectors that can represent the meaningful connections between words in an expressive way by analyzing a huge corpus.

Thus, word embeddings being more concise and precise, are also capable to store contextual connections between words.

- **Practical Implementation of Word2vec Model**

  Using genism library, we can train our model using Word2Vec

  Hyperparameters of this model

- **Window_context:** The maximum interval between a target word and the words that surround it. The default size of the window is 5 but here 30 is provided to the model.
- **min_count:** The minimal number of words to take into account while training the model; words with fewer than this number of occurrences will be disregarded. The value of min count is set to 5 by default but here 2 is provided to the model.
- **Feature_size:** The number of dimensions of the embeddings which is set to 100 by default.
- **Sample:** The threshold for determining whichever higher-frequency words are downsampled at random.

Computing Similarity

The similarity has been computed for the technical skills obtained through Topic Modelling approach which is *insights, reporting, sql, bi, excel, analysis, powerbi, visualization, azure, tools and development*.

For a given technical skill, the 10 most related word would be returned through this model.

Example: Computing similar words for reporting(figure 20)

```
'reporting': [('data', 0.9999885559082031),
 ('management', 0.9999827742576599),
 ('team', 0.9999824166297913),
 ('working', 0.9999822974205017),
 ('requirements', 0.9999812841415405),
 ('ability', 0.9999808073043823),
 ('experience', 0.9999803900718689),
 ('business', 0.9999798536300659),
 ('analysis', 0.9999797344207764),
 ('skills', 0.9999791383743286)],
```

For Reporting(Figure 20), the word 'data' is most similar which can be observed through its highest probability while 'skills' is least similar.

Figure 20: Depicting related words for Reporting

Example: Computing similar words for Excel(Figure 21)

```
'excel': [('data', 0.999984860420227),
 ('experience', 0.9999837875366211),
 ('role', 0.999982476234436),
 ('skills', 0.999981164932251),
 ('team', 0.9999808073043823),
 ('business', 0.9999791383743286),
 ('working', 0.9999776482582092),
 ('apply', 0.9999776482582092),
 ('analytical', 0.9999770522117615),
 ('analyst', 0.9999765753746033)],
```

For Excel(Figure 21), the word 'data' is most similar which can be observed through its highest probability while 'analyst' is least similar. This also conveys that Excel is required as an experience, considered as of the skills and preferred for the analytical job role.

Figure 21: Depicting related words for Excel

Example: Computing similar words for sql(Figure 22)

```
'sql': [('information', 0.9999741315841675),
 ('data', 0.9999715685844421),
 ('team', 0.9999699592590332),
 ('business', 0.9999691843986511),
 ('experience', 0.9999675750732422),
 ('skills', 0.9999675154685974),
 ('role', 0.9999664425849915),
 ('reports', 0.9999659657478333),
 ('working', 0.9999654293060303),
 ('analysis', 0.9999638795852661)],
```

For Sql(Figure 22), the word 'information' is most similar which can be observed through its highest probability while 'analysis' is least similar. We can further interpret that sql can be used to interpret some information and is also related to other words like data, skills, reports etc. which conveys sensible information for meeting the research objective.

Figure 22: Depicting related words for Sql

3.4.1.3 Understanding Grammer

Every corpus of text has its own language style and grammar. Grammar has been examined to uncover patterns in sentences, which can subsequently be utilized to construct phrases that characterize our skills.

- **Parts of Speech (POS) Tagging**

  Through POS tagging approach, we were able to understand the POS tags of the technical skills identified from the Topic Modelling and Word2vec model. Therefore, relevant phrases from the corpus were extracted after identification of the POS tags.

  Parts of Speech Tagging (POS Tagging) is a method of marking up words in text format for a specific section of a speech based on its definition and context. It is in charge of reading text in a language and assigning a token (Parts of Speech) to each word. Grammatical tagging is another name for it.

- **Why Is POS Tagging Beneficial?**
  POS tagging proves to be beneficial, especially if one has numerous POS tags for words or tokens. Depending on the context, the term "google" can be used as both a noun and a verb. It is critical to recognize this distinction while processing natural language. Through the use of Spacy library, it can determine the right POS tag for a word based on the context (words around it) using machine learning algorithms.

  Example:

- ***Examining the first record of the job description***

```
A leading online retail & groceries brand is undergoing considerable investment in their Insight & Analytics teams across the business.

This is a fantastic opportunity to join a data-driven e-commerce business at the forefront of technological innovation. Working in an entrepreneurial start-up environment with the backing of a huge brand.

The Role:

*The main purpose of the role is to Drive Data quality improvement across the business.
*Support the data governance programme (and associated data governance council) through detailed data analysis.
*Builds tools and dashboards to monitor quality and provide alerts to all data users.
*Manage, monitor, assess, resolve data quality issues with EDW, Data Marts and reporting solutions.
*Answer business questions about data, be the bridge between data and Insight.
*Assess and profile new data sources to provide input to integration design.
*Collaborate with business users and analysts to capture business rules and data quality rules.
*Maintain enterprise data model in line with data quality deliverables.

The Candidate:

*Minimum 2 years experience working as a Data Quality Analyst.
*Experience of creating and implementing data quality dashboards and monitoring techniques.
*Experience of data quality profiling, and creating data quality rules agreed with the business stakeholders.
*Good Data Modelling experience.
*Good SQL and/or PL/SQL experience.
*MS Excel/GSheets at an advanced level.
*SQL, with table design and creation as a minimum standard, with the ability to manipulate fields, insert, delete and move data.
*Knowledge of Data integration, reconfiguration and manipulation
*Experience working with cloud platform - especially GCP

Profiles Creative is acting as an Employment Agency in relation to this vacancy.
```

- ***Examining the POS tags assigned for the first record of the job description***

```
*Experience of creating and implementing data quality dashboards and monitoring techniques.
*            PUNCT      NFP      superfluous punctuation
Experience   NOUN       NN       noun, singular or mass
of           ADP        IN       conjunction, subordinating or preposition
creating     VERB       VBG      verb, gerund or present participle
and          CCONJ      CC       conjunction, coordinating
implementing VERB       VBG      verb, gerund or present participle
data         NOUN       NNS      noun, plural
quality      NOUN       NN       noun, singular or mass
dashboards   NOUN       NNS      noun, plural
and          CCONJ      CC       conjunction, coordinating
monitoring   NOUN       NN       noun, singular or mass
techniques   NOUN       NNS      noun, plural
.            PUNCT      .        punctuation mark, sentence closer
```

Figure 23: POS tags for first record of Job Description

```
*Experience of data quality profiling, and creating data quality rules agreed with the business stakeholders.
*            PUNCT      NFP      superfluous punctuation
Experience   NOUN       NN       noun, singular or mass
of           ADP        IN       conjunction, subordinating or preposition
data         NOUN       NNS      noun, plural
quality      NOUN       NN       noun, singular or mass
profiling    NOUN       NN       noun, singular or mass
,            PUNCT      ,        punctuation mark, comma
and          CCONJ      CC       conjunction, coordinating
creating     VERB       VBG      verb, gerund or present participle
data         NOUN       NNS      noun, plural
quality      NOUN       NN       noun, singular or mass
rules        NOUN       NNS      noun, plural
agreed       VERB       VBD      verb, past tense
with         ADP        IN       conjunction, subordinating or preposition
the          DET        DT       determiner
business     NOUN       NN       noun, singular or mass
stakeholders NOUN       NNS      noun, plural
.            PUNCT      .        punctuation mark, sentence closer
```

Figure 24: POS tags for a sentence from JD

```
*MS Excel/GSheets at an advanced level.
*            PUNCT      NFP      superfluous punctuation
MS           PROPN      NNP      noun, proper singular
Excel        PROPN      NNP      noun, proper singular
/            SYM        SYM      symbol
GSheets      PROPN      NNPS     noun, proper plural
at           ADP        IN       conjunction, subordinating or preposition
an           DET        DT       determiner
advanced     ADJ        JJ       adjective
level        NOUN       NN       noun, singular or mass
.            PUNCT      .        punctuation mark, sentence closer
```

Figure 25: POS tags for a sentence from JD

The above figues shows coarse-grained POS tags, fine-grained POS tags, and the explanation for the tags for all the words in the sentence.

- As shown in Figure 23, the sentence extracted from job description- *Experience of creating and implementing data quality dashboards and monitoring techniques.*

The technical skills such as dashboard, monitoring and techniques are of noun type depicted by Noun tags NNS, NN and NNS respectively.

- As shown in Figure 24, the sentence extracted from job description- *Experience of data quality profiling, and creating data quality rules agreed with the business stakeholders.*

The technical skills such as data, quality and, profiling are of noun type depicted by Noun tags NNS, NN and NN respectively.

- Similarly, As shown in Figure 25, the sentence extracted from job description- *MS Excel/GSheets at an advanced level.*

The technical skills such as MS, Excel and, GSheets are of noun type depicted by Noun tags NNP, NNP and NNPS respectively.

- **Visualizing Parts of Speech Tags**

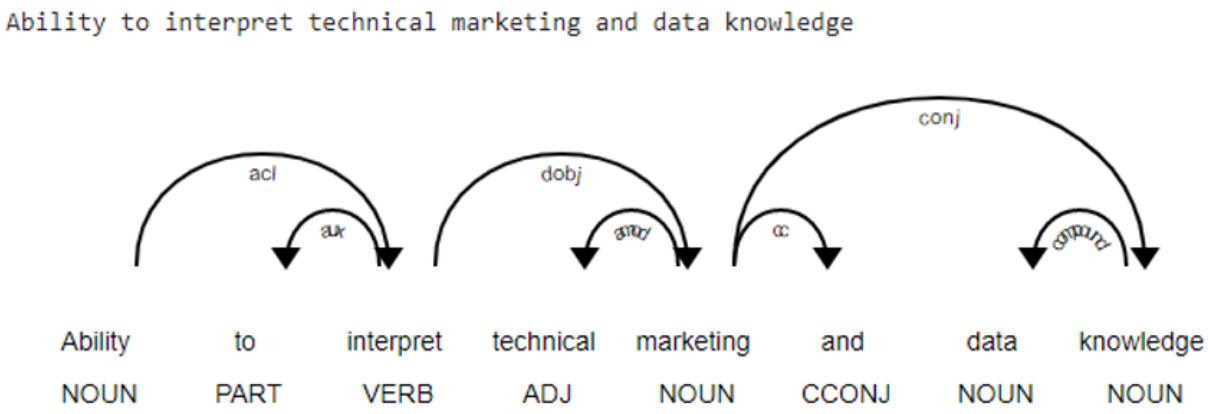


Figure 26:Visulaization through Displacy module

Displacy module from Spacy library has been used to visualize dependency of each token on another along with the POS tag(Figure 26).

To conclude, on the basis of above three examples(fig 23, fig 24 and fig 25), as the technical skills identified from the Topic Modelling and Word2vec model are assigned Noun tag, therefore, this study considered to extract the phrases which are of Noun type.

- **Extracting Noun Phrases**

In order to extract the noun phrases from our dataset, two libraries were considered to be tested on a sentence and the subsequent results were compared.

Sentence- *The ideal candidate will have a background in Marketing and Data, with a keen eye for trends and data analysis.*

- ***Result generated by Textblob Library***

  ideal candidate

data
keen eye
data analysis

- ***Result generated by Spacy library***

  The ideal candidate
  a background
  Marketing
  Data
  a keen eye
  trends
  data analysis

On the basis of analyzing the results through comparison, Spacy library has outperformed the result of Textblob library as it has extracted more number of noun phrases as it has extracted all the words which should be considered as technical skill that is data, trend and data analysis. However, on the flip side, it has considered some noise as well which has been taken care off during labelling of the data in further stage.

As Spacy library proved to be better to extract the noun phrases for the above sentence, therefore it was implemented on the whole dataset. Thereafter, data pre-processing steps were undertaken at the phrase level where all the records containing noun phrases were changed to lowercase, extracted the words containing only the alphabets, removed the stop words and removed the empty spaces(if any) found before or after the phrase.

The cleaned phrases were then converted to dataframe and on inspecting it was found that the total number of records obtained was 51367. These records were further split into training and test data so that our model predicts the new technical skill on the test data by learning the skills which were labelled in our training data.

The training data accounted to be 60%(of total records) consisting of 30890 records while testing data accounted to be 40%(of total records) consisting of 20477 records.

	Clean_Noun_Phrases
0	leading online retail groceries brand
1	considerable investment
2	insight analytics teams
3	business
4	fantastic opportunity
...	...
51362	cv
51363	information
51364	job
51365	similar jobs
51366	lewis phillips

[51367 rows x 1 columns]

Original dataset consisting of 51367 records.

	training_noun_phrases
1	considerable investment
2	insight analytics teams
3	business
4	fantastic opportunity
6	forefront
...	...
51362	cv
51363	information
51364	job
51365	similar jobs
51366	lewis phillips

[30890 rows x 1 columns]

Training dataset consisting of 30890 records.

3.4.1.4. Data Labelling

Our task is to discover and distinguish important phrases that indicate any type of skill from irrelevant noun phrases. Our Skill extractor is built around this.

From the training dataset obtained as a result of Noun phrase extraction, a new column was created which was intended to label the phrases as 1 to identify it as a skill and 0 to identify it as non-skill.

As we are more concerned about labelling the technical skills, so the phrases were labelled as 1 combining the phrases which contained the technical skills obtained from Topic modelling and Word2vec algorithm like Excel, Sql, dashboards, visualization, power bi, visualization and, the technical skills through basic intuition and domain experience.

Thus, the phrases were labelled as a skill or 1 if any of their word matched with the following technical skills:

'python','excel', 'database','queries','dashboards','plsql','modelling','tsql','integration','crm','salesforce','algorithm','impala','security','r','c','sas','bi','sspowerbi','power bi','powerpoint','pipeline','pandas','ssrs','ssis','alteryx','methodologies','visualization','powershell','sqldb','languages','qa','java','cloud','mssql','nosql','linux','dax','jira'

Id	training_noun_phrases	skill_set_bool
27	data quality analyst	0
28	data quality profiling	0
29	business stakeholder	0
30	good data modelling experience	1
31	good sql andor plsql experience	1
32	m excelgsheets	0
33	sql	0
34	table design	0

Figure 27: screenshot of labelled non_phrases

From the Figure 27, the skill set for record number 30 and 31 has been set to 1 as modelling from the given list matched with one of the word present in record number 30- good data modelling experience and, plsql from the given list matched with one of the words present in record number 31-good sql andor plsql experience.

Moreover, other technical words such as statistical, tableau, sql, programming, which could be added based on the my domain experience.

Moreover, some of the technical skills such as insights, reporting, sql, development, azure etc. and other related technical skills drawn through domain experience were intentionally not considered while labelling the training dataset as I wanted my model not to learn from plethora of technical skills or identify the skills in test data which it already learnt at the time of training as this would defeat the purpose or research objective for this project. Instead, some of the models has been developed in such a manner and an optimal model has been chosen which would be presented further that would identify a new technical skill which it had not been trained earlier.

Designing Machine Learning and Deep Learning models for Skill extraction

3.4.2.1 Overview of the obtained dataset

The labelled training dataset which we obtained as in Figure 27, after loading the data, some of the feature engineering steps(explained ahead) were taken before our data was fed into various models.

There are in total 30890 records present in the training dataset consisting of 3 column which is Id, training_non_phrases and skill_set_bool.

As we can see that the third column 'skill_set_bool' holds two values i.e 1 and 0, thus representing 1 as skill and 0 as non-skill, so we can identify the problem to implement the models to correctly identify the label of particular record of the dataset as 1 or 0 , thus this can be termed as a binary text classification problem.

Therefore, binary classification is a classification technique in which a set of data is divided into two groups. It's essentially an estimate as to which of two categories the object belongs to.

3.4.2.2 Feature Engineering

It is the process of converting raw data into characteristics that the predictive model can use to better reflect the underlying problem.

Before, working on the feature engineering, the training data was further split into training and test data so that our model analyzes the results and compares with the test data.

The training data accounted to be 70%(of total records of original training dataset) while testing data accounted to be 30%(of total records of original training dataset)

As we know that machine doesn't understand text so we need to find a way to convert the text in machine readable language and that is converting it into numerical representation. So, Tokenizer class in Keras library was used to convert the text into numbers.

For example:

Tokenizer in keras will assign the text "I love deep learning" to the numbers below.

I : 2,love : 4,deep : 1,learning : 3.

If we feed a large dataset with many documents into the keras tokenizer, all of the text words will be converted into a sequence of numbers.

Thus, these feature engineering steps, would be common to all models implemented in this project.

Detailed step-by-step explanation to convert the text into numbers:

1. *Tokenize the training data*- In this step, we created a Tokenizer Object and supplied num_words=1000 as a parameter which means that we wanted our model to consider maximum number of 1000 most common words from our data and truncate the rest.

2. *Getting training data word index*- A word index is created as a result of the tokenization process, which maps words in our corpus to their numeric representation, a mapping that will be crucial for encoding our sequences.

3. *Encoding training data sentences into sequences*- Here, we used our vocabulary to encode our sequences that we have tokenized our data and created a word to numeric representation mapping. For example: Converting text sentences like "My name is Matthew" to "6 8 2 19," where each of those numbers corresponds to a word in the index.

4. *Padding of the training sequences- One of the challenge while converting the text to numeric representation is that different phrases of text from our dataset possesses different lengths, so to counter this issue, I have used pad_sequence method which pads the sequence of the words with zeros. Also, I specified the maximum length of the sequence that my model should consider which is 100. This removes sequences that are longer than that number but if short than it has padded the sequence with zeros.*

Similarly, the above mentioned feature engineering steps were carried on the test data as well.

On inspection of the target column that is skill_set_bool for training dataset, it was found that there is huge difference between the number of records for skills and number of records for non-skill.

The number of records which was labelled as skill was 755 which is almost one-third of the number of records labelled as non-skill which is 20868.

- ***Machine Learning Models***

3.4.2.3 Constructing Baseline model

- **Introduction to Baseline Model**

  A baseline model is one that is straightforward to set up and has a good probability of producing good outcomes. Because implementations are readily accessible in popular packages, experimenting with them is typically simple and inexpensive.

- **Why its is important to start with a baseline Model**

  A baseline takes just 10% of the time to create, but it will get us 90% of the way to getting pretty decent outcomes. In terms of accuracy, baselines let us put a more complicated model into context.

  Generally, there are three levels of performance you can easily estimate:

  - **Trivially attainable performance** - This is a measure that every model should be able to beat. In a classification job, an example of this value would be the accuracy one would receive by predicting the most common class.
  - **Human performance**- This is the highest level at which a person can do the task. Some jobs (such as Go) are far superior to those performed by humans, while others are very worse (like writing poetry). Knowing how excellent a human is at something might help us to establish expectations for an algorithm ahead of time, but because the human/computer gap varies so much by field, we may need to do some research to calibrate.

- **Required to deploy performance**- This is the bare minimum value that would qualify our model for production in terms of both business and usability.

- **Baseline Model Specifications**

Logistic Regression was used as a baseline model for our project.

- **Introduction to Logistic Regression**

Logistic regression is a model analysis approach in which the target variables (outputs) are discrete values for a collection of characteristics or inputs (X).

In the case of Categorical values, Logistic Regression uses the sigmoid function to generate predictions.

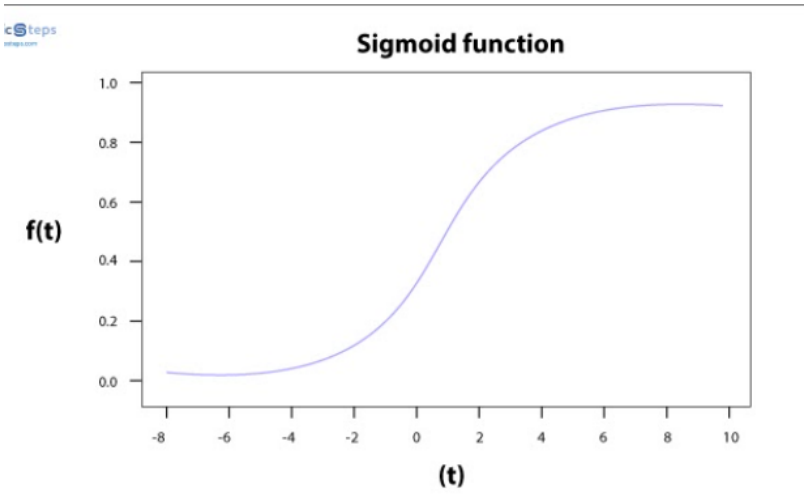
It establishes a cut-off point value, which is often set at 0.5, which, when surpassed by the Logistic curve's anticipated output, yields the relevant predicted output in the form of which category the dataset belongs to.

As an example, if the output exceeds the cutoff point in the Diabetes forecasting model, the forecast output will be Yes for Diabetes, otherwise No if the output is below the cutoff point.

- **What is Sigmoid Function**

It's a mathematical function with the feature of being able to translate any real value to a number between 0 and 1 in the shape of the letter "S" as shown in Figure 28. The logistic function is also known as the sigmoid function.

$$Y = 1 / 1+e^{-Z}$$



Sigmoid function

Figure 28: Sigmoid function graph

As a result, if the value of z approaches positive infinity, the anticipated value of y becomes 1; if it approaches negative infinity, the predicted value of y becomes 0. If the sigmoid function's result is more than 0.5, the label is classified as positive class 1, and if it is less than 0.5, it is classified as negative class, or label class 0.

- **What to do after constructing the baseline model?**

After constructing the baseline model, the next stage is to determine what the baseline does not capture. This will influence our decision to choose a more complicated model. However, we should not overlook the reality that strengthening baselines might cause previously successful instances to fail, as gains are not always additive. This is especially true for deep learning models, where the lack of interpretability makes inferring failure scenarios more difficult.

Furthermore, if the model ignores any aspects that our domain expertise tells us are critical, it provides a warning signal, indicating that our data may not be a good representation of the population or that the model is unsuitable.

3.4.2.4 Multinomial Naïve Bayes (MultinomialNB) model

- **Introduction**

It is a supervised machine learning algorithm and has a probabilistic learning approach which is popular in Natural Language Processing (NLP). The program guesses the tag of a text, such as an email or a news article, using the Bayes theorem. It calculates each tag's likelihood for a given sample and outputs the tag with the highest probability.

The Naive Bayes classifier is made up of a number of algorithms that all have one thing in common: each feature being categorized is unrelated to any other feature. A feature's existence or absence has no bearing on the presence or absence of another feature.

- **How does NB Algorithm work?**

We need to understand the following terms before going through the working:

Conditional Probability- Conditional probability is the chance of one event A occurring after another event B with some link to A has previously occurred.

$$P\left(\frac{B}{A}\right) = \left(\frac{P(A \cap B)}{P(A)}\right)$$

Prior Probability: This probability can be described as previous knowledge or belief, or the likelihood of an occurrence calculated before fresh evidence is collected. As new information becomes available, this likelihood is updated to give more accurate findings.

Thus, Prior probability is what we term the probability that is calculated using prior observations.

Posterior probability: It may be translated as: What is the new likelihood of an event occurring after considering additional information?

Because it incorporates more information, it is a better depiction of the fundamental reality of a data creation process.

Algorithm working steps:

Step 1: Calculate the Prior Probability in training data for provided class labels.

Step 2: For each class, calculate the Likelihood Probability for each feature attribute.

Step 3: Using Bayes' Theorem, calculate posterior probability(which is conditional probability assigned after relevant evidence or background taken into account)

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

Bayes Theorem Equation

- P(A|B) — the probability of A event occurring, given B event has occurred [Posterior Probability]
- P(B|A) — the probability of B event occurring, given A event has occurred [Likelihood Probability]
- P(A) — the probability of A event [Prior Probaility of A]
- P(B) — the probability of B event [Prior Probaility of B]

Step 4: Return a class label for the query sample with a greater Posterior Probability prediction.

- **Hyperparameter Tuning to improve the performance of the MultinomialNB model?**

  Through GridSearchCV which is sklearn's wrapper class, hyperparameter tuning was done. Therefore, this class cycles over all params grid parameters with a number of cross-validation folds as cv parameter, assesses model performance on all combinations, and saves all results in the cv results_ attribute. In the best_estimator_attribute, it additionally saves the model that performs best among all cross-validation folds, as well as the best score in the best score_ attribute.

  Parameters passed in GridSearchCV class:

  Param_grid: This allows searching across any sequence of parameter settings by using a dictionary with parameter names (strings) as keys and lists of parameter settings to attempt as values.

Alpha which is an additive smoothing parameter was passed in param_grid

Verbose: It's the verbosity: the greater the number, the more messages; it's set to 5 in this case.

Cv: It is a cross-validation generator or an iterable, with a 5-fold cross-validation in this example.

N_jobs: It specifies the maximum number of concurrently operating workers; in this example, it is -1, implying that all CPUs are in use.

- ***Deep Learning Models***

Concept Overview- Implementation of Neural Networks

- **Introduction to Keras model**

  For constructing Deep Learning models for research objective, we have used Keras as it is a high-level toolkit for constructing deep-learning models that provides high-level building pieces. Low-level operations like tensor manipulation and differentiation are not supported. Instead, it uses the Keras backend engine, which is based on a specific, well-optimized tensor library.

- **Neural Network Models in Keras**

  Keras defines neural network models as a graph of layers. The sequential and functional APIs of Keras may be used to construct models. Any type of model may be built using both the functional and sequential APIs.

  - The sequential API (used in this project) allows you to create models layer-by-layer for most problems. It is limited in that it does not allow you to create models that share layers or have multiple inputs or outputs.
    Thus, the Sequential model is a linear stack of layers in which we utilize Keras extensive layer library. The Dense layer is the most popular, and it's your standard densely linked neural network layer with all the weights and biases.
  - Alternatively, the functional API allows you to create models that have a lot more flexibility as you can easily define models where layers connect to more than just the previous and next layers. In fact, you can connect layers to (literally) any other layer. As a result, creating complex networks such as siamese networks and residual networks become possible.

- **General Workflow for creating model in Keras**

The simple workflow in Keras is as follows:

1. Create the model- In order to create the model, create an empty Sequential constructor.

2. Create and add layers to the model- Kernel operations are implemented in the Keras core layers, which are utilized in nearly every network design.

   - Dense: This is a simple fully connected neural network layer. This layer produces the output of the following function:

   activation((inputs x weights)+bias) where activation refers to the activation function passed to the layer, which is None by default.

   - Activation: This layer applies the specified activation function to the output. This layer produces the output of the following function:

     activation(inputs) where activation refers to the activation function passed to the layer.

     The following activation functions are available to instantiate this layer:

     softmax, elu, selu, softplus, softsign, relu, tanh, sigmoid, hard_sig moid, and linear.

     However, in this project only Sigmoid/Logistic Activation function is used for binary text classification. The Sigmoid function converts any input into a number between 0 and 1. The function sigmoid returns a value near to zero for small values (-5), and a value close to one for big values (>5). Sigmoid is the same as a two-element Softmax with the second element set to zero. Generally, the sigmoid is commonly employed in binary classification.

   - Embedding layer- Our data is still hardcoded at this time. We can now utilize Keras Embedding Layer, which takes the previously computed integers and translates them to a dense embedding vector.
   - The following parameters has been provided to the Embedding layer in all the deep learning models:
     - **Input_dim**: The size of our vocabulary
     - **Output_dim**: The size of the dense vector. We have set this to 50 in our case.
     - **Input_length**: The length of the input sequence. Here I have set the maximum length of input sequence as 100.

3. Compile the model- The model that was created in the preceding steps must be compiled alongside the model. Before it can be used for training and prediction, it must first be compile().

   The compile method takes three arguments:

   - **Optimizer**: In the optimization iterations, this function is used to update the parameters. *RMSprop optimizer is used for all the deep learning models in this project.* In this optimizer the goal is to keep the average square of gradients moving in the same direction. The gradient is also divided by the average's root in the second step.
   - **Loss:** The optimizer function adjusts the settings such that the loss function's output is as minimal as possible. *Thus, binary_crossentropy function is used as loss function for all the deep learning models in this project* as it calculates the loss in cross-entropy between true and anticipated labels. When there are just two label classes, we employ this cross-entropy loss (assumed to be 0 and 1).
   - **Metrics:** The third argument is a list of metrics that must be obtained while the model is being trained. If verbose output is enabled, the metrics for each iteration are printed. The metrics are similar to loss functions and all loss functions can also be used as metric functions.

4. Train the model- The Keras model is trained by using model.fit() method.
5. Use the model for prediction or evaluation- The trained model can be used either to predict the value with the model.predict() method or to evaluate the model with the model.evaluate() method.

   Thus, the evaluate() function is used to determine the model's correctness. This is done with both training and testing data. Generally, we anticipate that the training data will be more accurate than the testing data. The longer a neural network is trained, the more probable it is to start overfitting.

3.4.2.5 Simple Recurrent Neural network

The Recurrent Neural Network (RNN) is a type of neural network that is designed to handle sequential input. Sequential data can be a series of observations across time, such as time series data, or a series of characters, words, and phrases, such as textual data.

RNN does this by extending traditional neural networks in the following ways:

- By adding loops or cycles to the computation graph, RNN offers the possibility to use the output of one layer as an input to the same or previous layer.
- The memory unit in RNN is included to store past inputs and outputs that may be used in the present calculation.

A fully linked recurrent neural network is implemented in this layer as per figure 29.

SimpleRNN Model Architecture

This model as shown in Figure 28 accepts the input dimension as the size of the vocabulary where the maximum sequence length it accepts at a time is 100 and produces an output of vector of size 50. Thus, these vectors are learned as the model trains. This output vector serves as an input for second hidden layer wherein SimpleRNN model is applied and it produces an output of vector of size equivalent to embedding dimension which is 50. This is finally connected to the Dense layer which is the output later containing a single node to display the output using Sigmoid Activation Function.

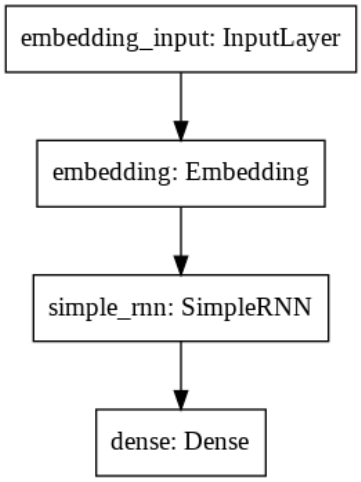


Figure 29: SimpleRNN model

SimpleRNN Model Architecture (Hyperparameter Tuning)

Similar architecture was followed as that of SimpleRNN model. In addition to the previous model, a Dropout layer was added before and after implementing the SimpleRNN recurrent layer to reduce overfitting. This is shown in Figure 30

Hyperparameter Tuning for Simple Recurrent Neural Network (SimpleRNN) model

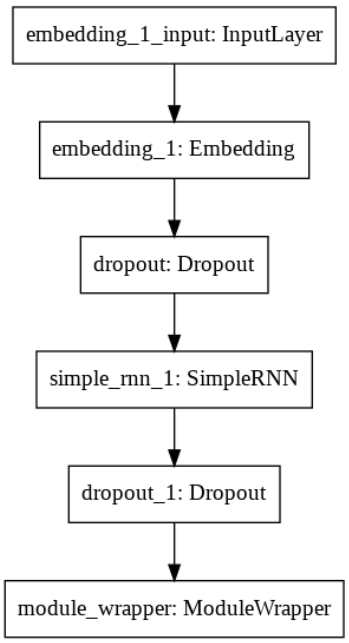


Figure 30: SimpleRNN (Hyperparameter Tuning)

In addition to altering the model architecture by addition drop out layers and changing the batch size and number of epochs, Early Stopping callback was also used.

Detailed Explanation of Early Stopping algorithm

This is one of the widely used technique in Deep Learning models where we can evaluate how well each iteration of a learning algorithm performs when we train it iteratively(figure 31).

New iterations refine the model until a specified number of iterations have been completed. However, if the model begins to overfit the training data, the model's ability to generalize can deteriorate.

Halting the training process before the learner reaches that point is referred to as early stopping.

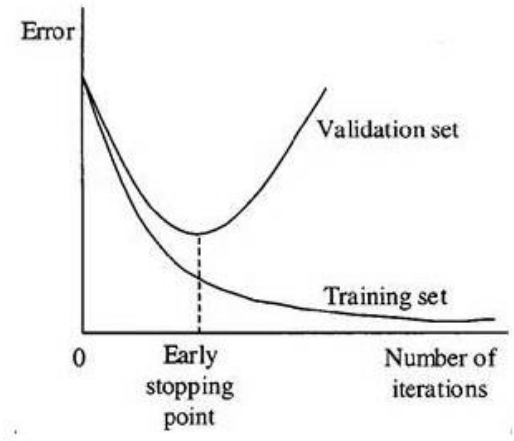


Figure 31: Demonstrating Early Stopping algorithm

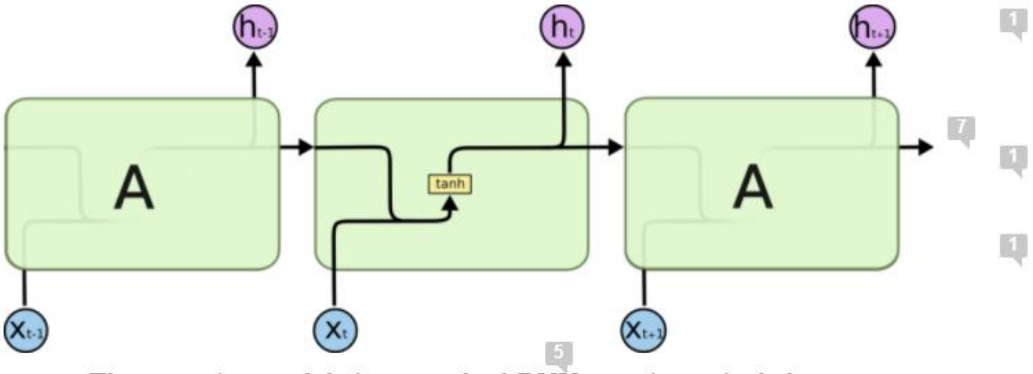
Why Drop Out Layer were used to prevent overfitting?

Because a fully connected layer takes up the majority of the parameters, neurons develop co-dependency with one another during training, limiting each neuron's unique power and resulting in over-fitting of training data.

3.4.2.6 Long and Short Term Memory (LSTM)

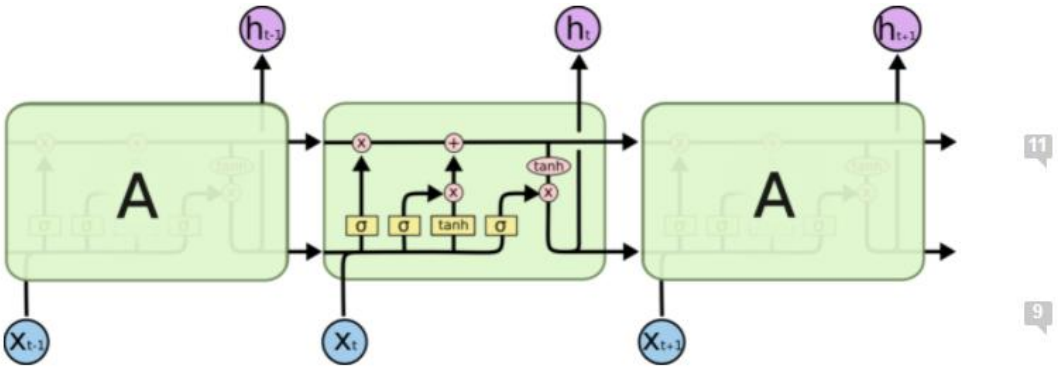
Long Short Term Memory Networks (LSTMs) are specifically intended to avoid the problem of long-term dependence. They don't have to work hard to remember knowledge for lengthy periods of time; it's virtually second nature to them.

All recurrent neural networks are made up of a series of repeated neural network modules. This repeating module in conventional RNNs will have a relatively basic structure, such as a single tanh layer(figure 32). LSTMs have a chain-like structure as well, but the repeating module is different. Instead of a single neural network layer, there are four, each of which interacts in a unique way(Figure 33).



The repeating module in a standard RNN contains a single layer.

Figure 32



The repeating module in an LSTM contains four interacting layers.

Figure 33

LSTM networks solve the vanishing/exploding gradient problem by incorporating gates that regulate access to previous data. When lengthy sequences are passed, the LSTM cell helps train the model more efficiently by selectively learning or deleting information.

The functions that make up the cell are also known as gates because they act as gatekeepers for the data that enters and exits the cell.

There are two types of memory in the LSTM model:

- h stands for working memory (hidden state)
- The letter c stands for long-term memory (cell state).

With only two linear contacts, the cell state or long-term memory moves from cell to cell. Through gates, the LSTM adds and removes data from long-term memory.

LSTM Model Architecture

This model as shown in Figure 34 accepts the input dimension as the size of the vocabulary where the maximum sequence length it accepts at a time is 100 and produces an output of vector of size 50. Thus, these vectors are learned as the model trains. This output vector serves as an input for second hidden layer wherein LSTM model is applied and it produces an output of vector of size equivalent to embedding dimension which is 50. This is finally connected to the Dense layer which is the output later containing a single node to display the output using Sigmoid Activation Function.

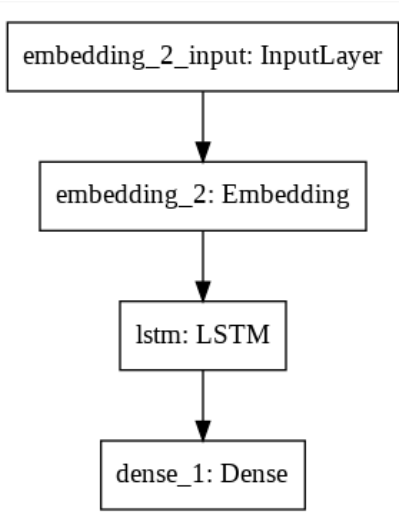


Figure 34: LSTM model

Hyperparameter Tuning for Long and Short Term Memory (LSTM) model

LSTM Model Architecture (Hyperparameter Tuning)

Similar architecture was followed as that of LSTM model. In addition to the previous LSTM model, a Dropout layer was added before and after implementing the LSTM recurrent layer to reduce overfitting. This is shown in Figure 35

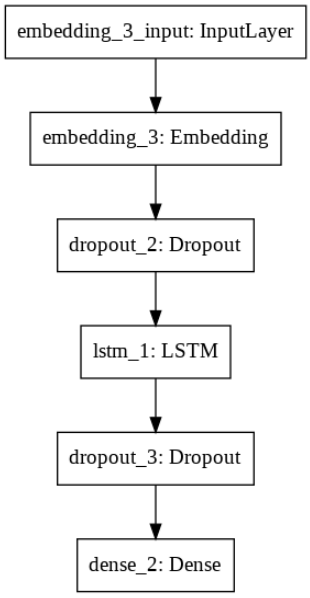


Figure 35: LSTM (Hyperparameter Tuning)

3.4.2.7 Convolutional Neural Networks(CNN)

Generally, In a neural network, neurons are supplied inputs, and the weighted total of those inputs is considered by the neurons, who then pass it through an activation function and send the output to the next neuron(Figure 36)

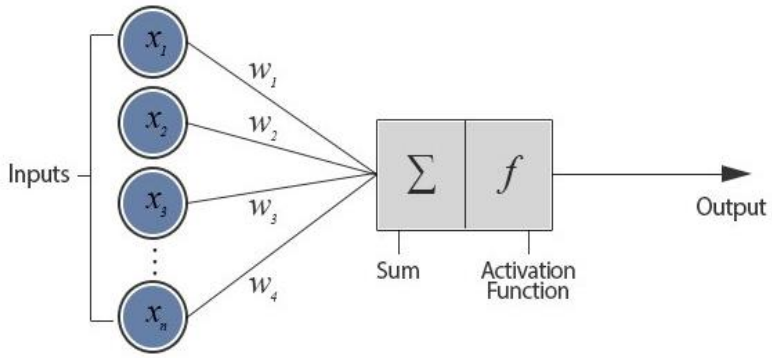


Figure 36

Because it functions over a volume of inputs, a convolutional neural network differs from a neural network.

Each layer searches the data for a pattern or helpful information.

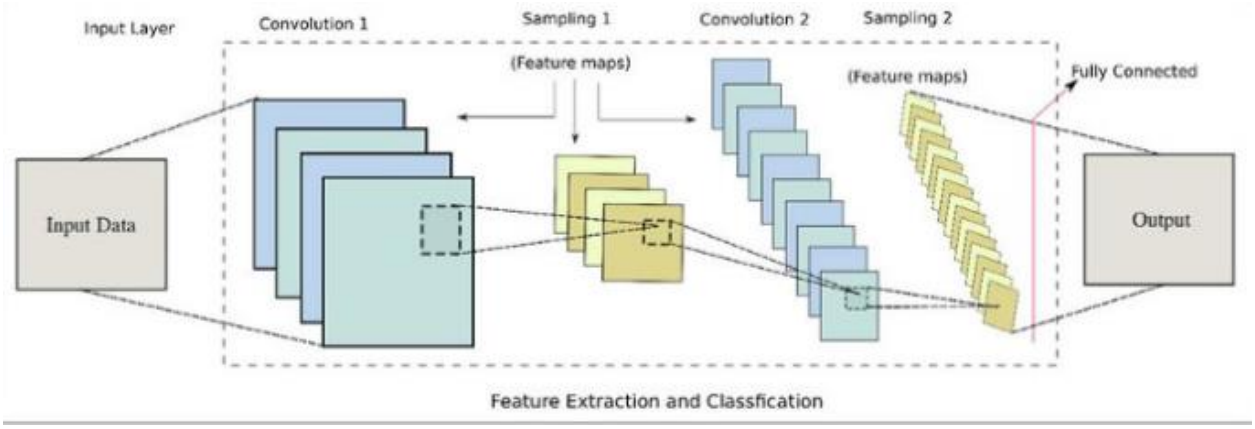


Figure 37: Feature Extraction and Classification

As from Figure 37, this is how CNN's architecture appears in most cases. Depending on the task and data set we're working with, it will be different. Before we begin our goal of text categorization, we must first grasp certain concepts in the architecture of convolutional neural networks.

Convolution: It is a mathematical combination of two relationships to produce a third relationship. Joins two sets of information.

Convolution over input: We utilize convolution to extract features from input data by applying a filter/ kernel (both can be used interchangeably). This is crucial for extracting features. There are some settings for the sliding filter, such as how much input to take at once and how much input should be overlapped.

- Stride: Every instance of time is moved by the size of the step filter.
- Filter count: We want to employ a certain number of filters.

To prevent the feature map from shrinking, we usually add padding around the input(figure 38). If we don't include padding, feature maps with a large number of input components will begin to shrink, and important information outside the borders will be lost.

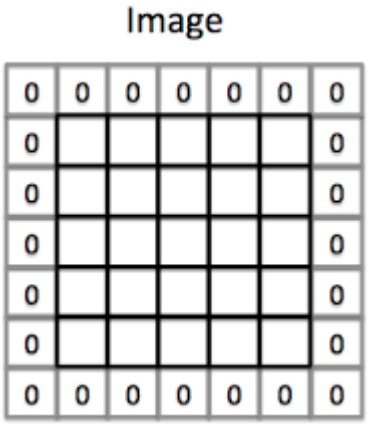


Figure 38: Depicting Padding around the input cell

It also increases performance by ensuring that the filter size and stride are appropriate for the input.

Adding to it, we need something to assist us decrease the amount of processing in the CNN while also avoiding overfitting the data. Instead of learning from the training data, overfitting will cause the model to memorize it.

We utilize a *pooling layer* in between the convolutional layers to reduce dimensional complexity while retaining the convolutions' significant information. The max pooling layer is an example for this. As seen in the Figure 39, it identifies the pool's maximum and passes it to the next layer.

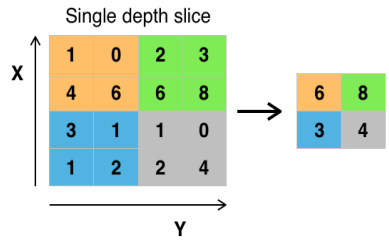


Figure 39: Pooling Layer

A Flatten layer is sometimes used to transform 3-D data into a 1-D vector.

The last layers in a CNN are completely connected layers, which means that each node in one layer is connected to every node in the other layer.

CNN Model Architecture

This model as shown in Figure 40 accepts the input dimension as the size of the vocabulary where the maximum sequence length it accepts at a time is 100 and produces an output of vector of size 50. Thus, these vectors are learned as the model trains. This output vector serves as an input for second hidden layer wherein Conv1D layer is applied which means that the inputs are convolutioned in a single spatial or temporal dimension by this layer. Thus, this Conv1D layer is later flattened (reducing the dimensions of the vector) with the help of GlobalMaxPooiling1D layer and fed to the Dense layer. Therefore, then prediction was made feeding the vector which was obtained from dense layer to another dense layer of 1 unit which is the output later containing a single node to display the output using Sigmoid Activation Function.

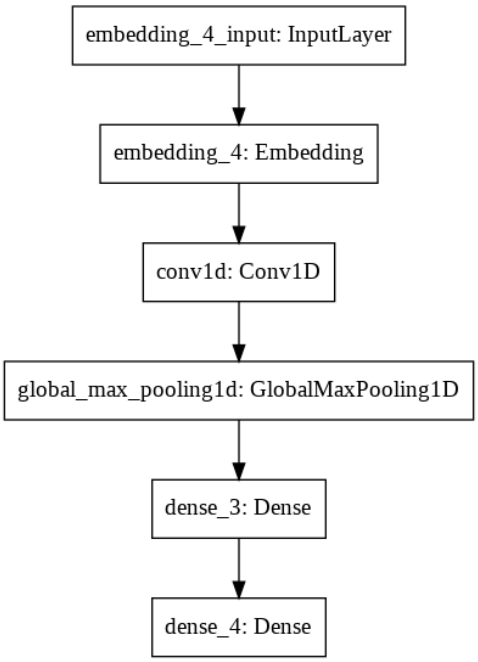


Figure 40

Hyperparameter Tuning for Convolutional Neural Networks (CNN)

CNN Model Architecture(Hyperparameter Tuning)

Similar architecture was followed as that of CNN model. In addition to the previous CNN model, a Dropout layer was added before Conv1D layer and Flatten layer was implemented to reduce the dimensions of the vector before predicting the output which contained a single node. This is shown in Figure 41.

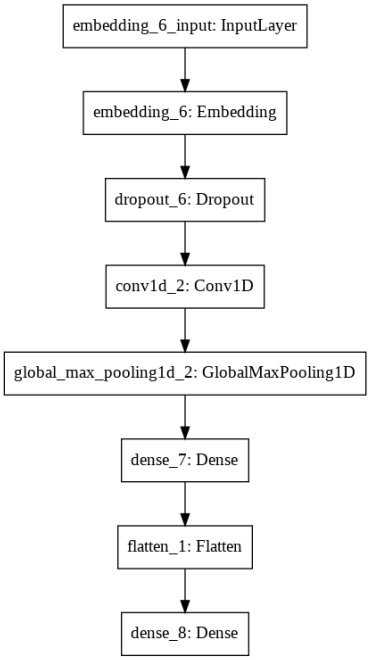


Figure 41: CNN (Hyperparameter Tuned)

The link of the project files can be assessed from the following link:

https://liveastonac-my.sharepoint.com/:f:/r/personal/200192196_aston_ac_uk/Documents/BusinessProject_200192196_skill_extraction?csf=1&web=1&e=MTmt7s

Chapter 4: Results and Discussion

Testing and assessing your machine learning model, like any other software development, is critical before it can be utilized to make actual predictions.

Accuracy, Precision, Recall, Sensitivity, and other quality measures are frequently used to assess the model's performance. However, monitoring just one metric may not provide a whole picture of your model, and we must evaluate numerous measures as per our research objective to fully comprehend the model's dimensionality.

Thus, thorough **Confusion Matrix** and **Classification report**, we have compared the performance of different models

- **Confusion Matrix**

A N x N matrix is used to evaluate the performance of a classification model, where N is the number of target classes. The matrix compares the actual targeted values to the machine learning model's predictions. This provides us with a comprehensive picture of how well our classification model is working and the types of errors it makes.

For a binary classification task, we'd use a 2 x 2 matrix with four values, as seen in figure 42:

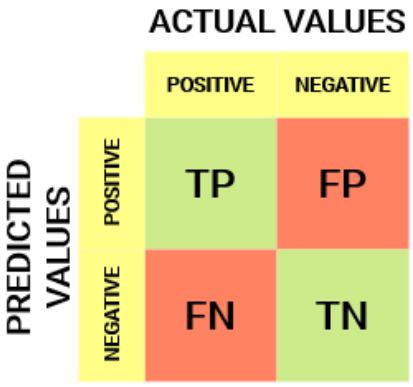


Figure 42: Confusion Matrix

Metrics Evaluation Parameters:

True Positive (TP)

- The anticipated value corresponds to the real value.
- The model forecasted a positive value, and the actual result was positive.

True Negative(TN)

- The anticipated value corresponds to the real value.
- The model forecasted a negative value, and the actual result was negative.

False Positive (FP) – Type 1 error

- The expected value was incorrectly predicted.
- Although the actual value was negative, the model anticipated that it would be positive.
- It is also known as Type 1 error

False Negative (FN) – Type 2 error

- The expected value was incorrectly predicted.
- Although the actual value was positive, the model anticipated that it would be negative.
- It is also known as Type 2 error

By using above terms, the confusion matrix can be used to obtain the following metrics.

Classification Accuracy

The ratio of correct guesses to the total number of predictions is known as classification accuracy. Or, to put it another way, how often does the classifier get it right?

$$Accuracy = \frac{No.\ of\ Correct\ Predictions}{Total\ No.\ of\ Predictions}$$

The confusion matrix can be used to calculate the accuracy. The equation for calculating the accuracy using the confusion matrix is as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Sensitivity/Recall

The ratio of correct positive predictions to the total number of positive predictions is called sensitivity or recall. Or, to put it another way, how sensitive the classifier is to recognizing positive cases. The True Positive Rate is another term for this.

$$Recall = \frac{No.\ of\ Correct\ Positive\ Predictions}{Total\ No.\ of\ Positive\ Predictions}$$

Recall may be computed using the confusion matrix as follows:

$$Recall = \frac{TP}{TP + FN}$$

Specificity

The ratio of true negative predictions to the total number of negative predictions is known as specificity. This determines the classifier's specificity in predicting good outcomes.

$$Specificity = \frac{No.\ of\ Correct\ Negative\ Predictions}{Total\ No.\ of\ Negative\ Predictions}$$

Specificity may be computed using the confusion matrix as follows:

$$Specificity = \frac{TN}{TN + FP}$$

Precision

The ratio of correct predictions to the total number of expected right predictions is known as precision. This metric assesses how well the classifier is in predicting positive outcomes.

$$Precision = \frac{No.\ of\ Correct\ Positive\ Predictions}{Total\ No.\ of\ Positive\ Predictions}$$

Precision may be computed using the confusion matrix as follows:

$$Precision = \frac{TP}{TP + FP}$$

Classification report

A classification report is used to assess the accuracy of a classification algorithm's predictions. It depicts the recall, precision, F1 score and support for the individual classes as well as on the basis of combining their results in the model.

In addition to Precision and Recall(as explained above), the following terms need to be understood for model evaluation:

F1 score- It is basically is the harmonic mean between precision and recall

Support- It can be depicted as the number of occurrence of the given class in our dataset

Macro F1- It calculates the F1 by class, but does not use weights for aggregation.

$$F1_{class1} + F1_{class2} + \cdots + F1_{classN}$$

Thus, considering this metric results in greater penalization of the model when it does not perform well for the minority class(which is skill in our case)

Weighted F1 score - It calculates the F1 score for each class separately, then adds them up using a weight based on the number of true labels in each class:

$$F1_{class1} * W_1 + F1_{class2} * W_2 + \cdots + F1_{classN} * W_N$$

Thus, this metric favors the majority class(i.e non_skill) which didn't seem to be appropriate in our case.

In our research project, as we are interested in getting the correct result for skill as compared to non-skill. Therefore, getting correct skill and reducing the incorrect prediction where the actual skills are wrongly predicted as non-skill would be more beneficial for evaluating the models.

However, for the model to incorrectly forecast as skill which are non-skill, also wouldn't be appropriate for the evaluation of the model though it is less important that the former for considering for the evaluation.

So, F1 score was considered as an optimal metric for evaluation as it takes the harmonic mean for both the above cases.

Therefore, while building the models for the research project, a target was set to improving on the F1 score.

4.1 Baseline Model

The Logistic model was trained on the new training data and in order to analyze the performance, the results were predicted.

The accuracy obtained was 97%

Confusion Matrix

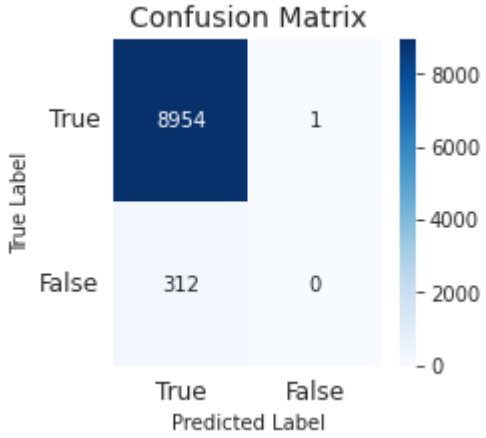


Figure 43: Confusion Matrix for Baseline

Observations:

- Correctly classified skill for 8954 out of 8955 samples.
- Correctly classified non-skill for 0 out of 312 samples.
- 1 skill was wrongly classified as non-skill.
- 312 non-skill were wrongly classified as skill.

Classification Report

	precision	recall	f1-score	support
0	0.97	1.00	0.98	8955
1	0.00	0.00	0.00	312
accuracy			0.97	9267
macro avg	0.48	0.50	0.49	9267
weighted avg	0.93	0.97	0.95	9267

Figure 44: Classification report for Baseline

From **Figure 44** as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 48%
- Macro Average Recall- 50%
- Macro Average F1-score- 49%

Thus higher accuracy in our case may attribute to the class imbalance scenario that arises when there is an uneven distribution of class in a dataset, i.e. the negative class (majority class) has a greater number of observations comparison to the positive class (minority class).

In our case, as our dataset is imbalanced in nature having the minority class i.e skill as one-third of the majority class. So, the higher accuracy in our model would result in our model learning from the majority class that is non-skill and generalizing the results according to it.

Another Example: Assume if we are working on a fraud detection challenge involving health insurance. In such cases, we typically find that 99 insurance claims are non-fraudulent and 1 is fraudulent for every 100. As a result, a binary classifier model does not need to be complicated in order to forecast all outcomes as 0 (non-fraudulent) with a high accuracy of 99 percent. Clearly, the accuracy metric is biased and not preferred in such instances where class distribution is skewed.

Therefore, I have considered oversampling the data using SMOTE so that my training dataset consists of equal number of records of skill and non-skill phrases.

Introduction to SMOTE algorithm used for balancing the data

Synthetic Minority Oversampling Technique (SMOTE) is an acronym for Synthetic Minority Oversampling Technique. This is a statistical method for evenly increasing the number of instances in your dataset. The module generates new instances based on current minority cases you provide as input. The number of majority cases does not change as a result of SMOTE adoption.

The method creates new examples by sampling the feature space for each target class and its nearest neighbors, then combining features from the target case with features from its neighbors. This method enhances the number of characteristics accessible to each class while also making the examples more generic.

- **What if the unbalanced data is not treated?**

The performance of the classifier model will be harmed if the unbalanced data is not handled beforehand. The bulk of the predictions will be for the majority class, and the minority class characteristics will be treated as noise in the data and ignored. As a result, the model will have a significant bias.

- **Applying SMOTE algorithm to the baseline model**

After applying SMOTE algorithm to the new training data, equal observation of classes(skills and non-skill) in target class was observed.

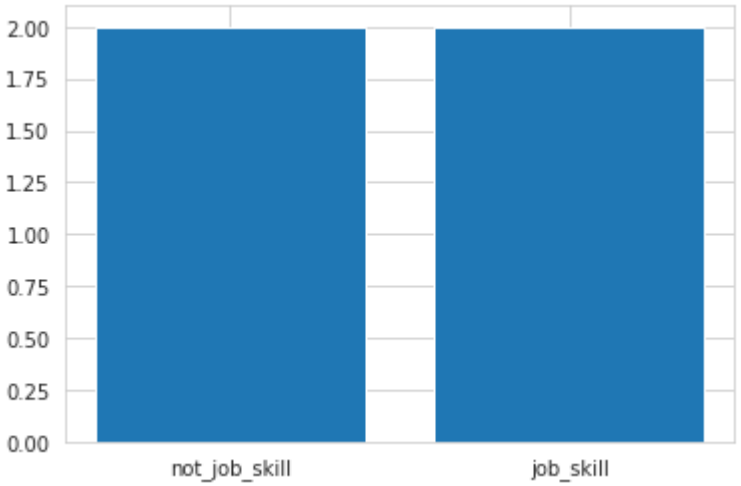


Figure 45: class balanced after applying SMOTE

The Logistic model was trained on the balanced training data (obtained through SMOTE algorithm) and in order to analyze the performance, the results were predicted(figure 45)

The accuracy obtained is 42%

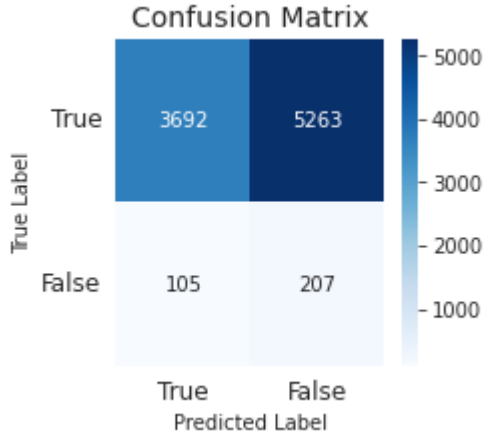


Fig 46: **Confusion Matrix for Baseline** for Baseline after applying SMOTE

Observations:

- Correctly classified skill for 3692 out of 8955 samples.
- Correctly classified non-skill for 207 out of 312 samples.
- 5263 skill was wrongly classified as non-skill.
- 105 non-skill were wrongly classified as skill.

Classification report:

	precision	recall	f1-score	support
0	0.97	0.41	0.58	8955
1	0.04	0.66	0.07	312
accuracy			0.42	9267
macro avg	0.51	0.54	0.33	9267
weighted avg	0.94	0.42	0.56	9267

Fig 47: CR for Baseline after applying SMOTE

From Fig 47 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 51%
- Macro Average Recall- 54%
- Macro Average F1-score- 33%

So, the baseline model was considered as the one which had balanced distribution of both the classes. Adding to it, same training data which is balanced using SMOTE algorithm has been implemented in rest of the Machine and Deep Learning models

4.2.1 Multinomial Naïve Bayes (MultinomialNB) model

The MultinomialNB model was trained on the training data and in order to analyze the performance, the results were predicted(Fig 48)

The accuracy obtained is 88%

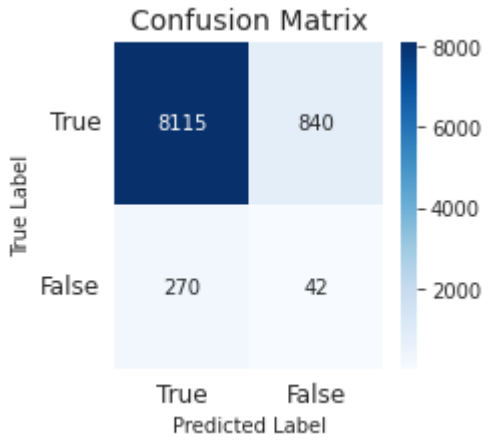


Fig 48: **Confusion Matrix** for Multinomial NB

Observations:

- Correctly classified skill for 8115 out of 8955 samples.
- Correctly classified non-skill for 42 out of 312 samples.
- 840 skill was wrongly classified as non-skill.
- 270 non-skill were wrongly classified as skill.

	precision	recall	f1-score	support
0	0.97	0.91	0.94	8955
1	0.05	0.13	0.07	312
accuracy			0.88	9267
macro avg	0.51	0.52	0.50	9267
weighted avg	0.94	0.88	0.91	9267

Fig 49: CR for Multinomial NB

From Fig 49 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 51%
- Macro Average Recall- 52%
- Macro Average F1-score- 50%

4.2.2 Multinomial Naïve Bayes (MultinomialNB) model (Hyperparameter tuning)

A range of values for alpha was passed in the model: 0.01, 0.1, 0.5, 1.0, 10.0, but the optimal parameter through which the Multinomial Naïve Bayes Model's performance has turned out to be alpha- 0.01.

The accuracy obtained is 88%

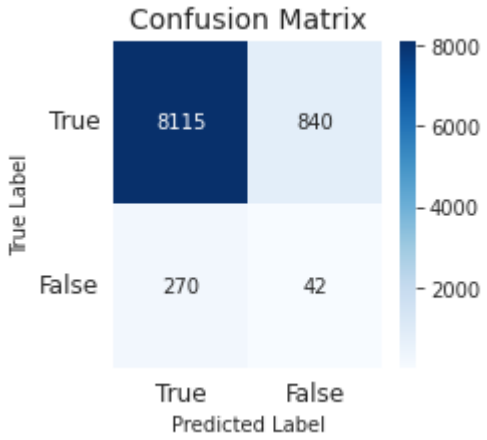


Fig 50 : Confusion Matrix for Multinomial NB (hyperparamter Tuning)

Observations:

- Correctly classified skill for 8115 out of 8955 samples.
- Correctly classified non-skill for 42 out of 312 samples.
- 840 skill was wrongly classified as non-skill.
- 270 non-skill were wrongly classified as skill.

	precision	recall	f1-score	support
0	0.97	0.91	0.94	8955
1	0.05	0.13	0.07	312
accuracy			0.88	9267
macro avg	0.51	0.52	0.50	9267
weighted avg	0.94	0.88	0.91	9267

Fig 51 : CR for Multinomial NB (hyperparamter Tuning)

From Fig 51 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 51%
- Macro Average Recall- 52%
- Macro Average F1-score- 50%

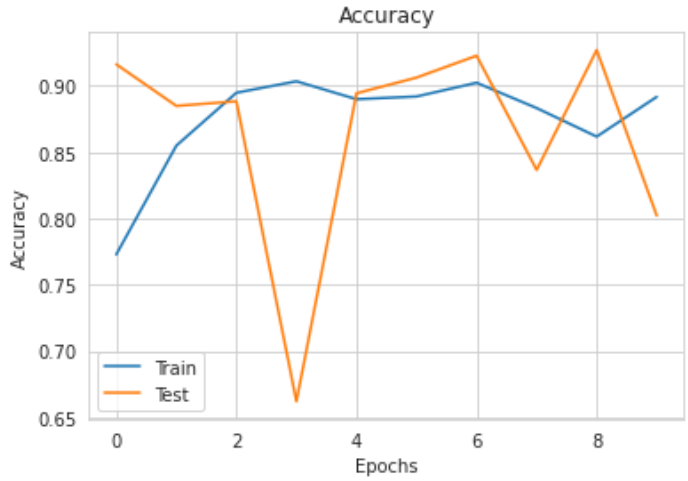
4.3.1 Simple Recurrent Neural Network (SimpleRNN)

The SimpleRNN model was trained for *batch of size* 60 and 10 *number of epochs*

```
Epoch 1/10
696/696 [==============================] - 115s 159ms/step - loss: 0.4632 - acc: 0.7728 - val_loss: 0.2575 - val_acc: 0.9164
Epoch 2/10
696/696 [==============================] - 110s 158ms/step - loss: 0.3404 - acc: 0.8549 - val_loss: 0.2370 - val_acc: 0.8850
Epoch 3/10
696/696 [==============================] - 110s 159ms/step - loss: 0.2752 - acc: 0.8949 - val_loss: 0.2806 - val_acc: 0.8884
Epoch 4/10
696/696 [==============================] - 111s 160ms/step - loss: 0.2496 - acc: 0.9035 - val_loss: 0.6943 - val_acc: 0.6619
Epoch 5/10
696/696 [==============================] - 111s 160ms/step - loss: 0.2859 - acc: 0.8900 - val_loss: 0.2712 - val_acc: 0.8945
Epoch 6/10
696/696 [==============================] - 111s 159ms/step - loss: 0.2813 - acc: 0.8921 - val_loss: 0.2692 - val_acc: 0.9063
Epoch 7/10
696/696 [==============================] - 111s 160ms/step - loss: 0.2478 - acc: 0.9024 - val_loss: 0.2479 - val_acc: 0.9228
Epoch 8/10
696/696 [==============================] - 111s 159ms/step - loss: 0.3072 - acc: 0.8833 - val_loss: 0.3429 - val_acc: 0.8366
Epoch 9/10
696/696 [==============================] - 110s 158ms/step - loss: 0.3489 - acc: 0.8618 - val_loss: 0.2225 - val_acc: 0.9272
Epoch 10/10
696/696 [==============================] - 111s 159ms/step - loss: 0.2763 - acc: 0.8918 - val_loss: 0.4603 - val_acc: 0.8021
```

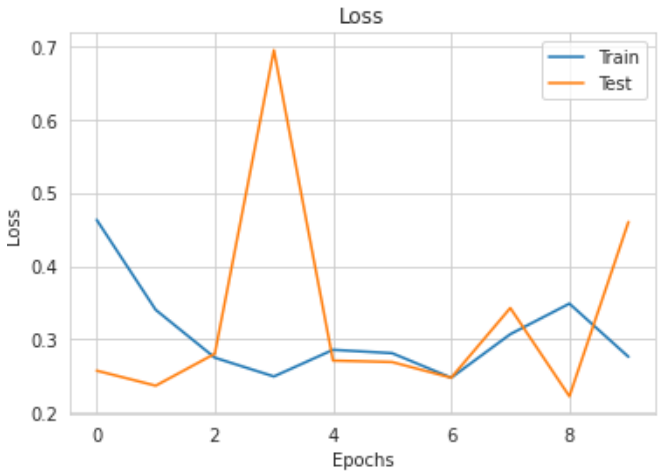
Fig 52: Training model SimpleRNN

After 10 epochs, the accuracy of the training data is 89% while that of test data is 80%



The accuracy for both the training and test data is fluctuating but the greater range for fluctuation has been observed more in the case of test data for 4th epoch where its accuracy decreased abruptly from 88% to 66% and then moderate change of state from 5th epoch.

Fig 53: Accuracy graph SimpleRNN



There is no major shift observed for the training data. However, for the test data there is a sudden increase in the loss value from 28% to 69% % at 4th epoch and then moderate change of state from 5th epoch.

Fig 54: Loss value graph SimpleRNN

As per the Gradient Descent optimization, it should minimize the desired quantity on every iteration. Thus, in such case with each epoch, the training loss lowers while the training accuracy improves.

However, the same isn't the case with validation loss and accuracy. They seem to peak after few number of epochs which is observed around 4 epochs in my case.

Thus, this is an example of overfitting which means that the model has performed better on the training data then it does on the test data or the data that it has never seen before. At this point, the model over-optimizes and learns the representations or patterns specific to the training data that do not generalize well to the test data.

Therefore, we can conclude from the loss plot above(fig 54) that we can see clear signs of overfitting because as the train Loss decreases, the validation (or test) loss depicts a fluctuation behavior as it frequently changes its state until it decreases finally.

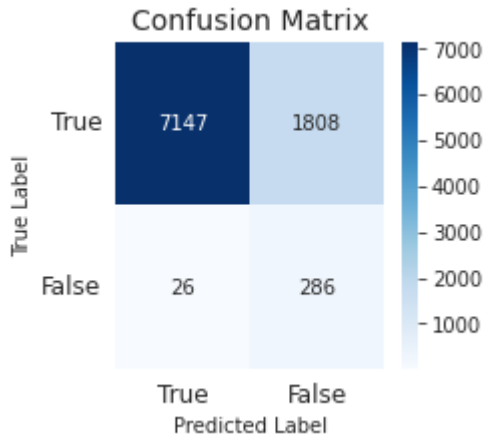


Fig 55: Confusion Matrix for SimpleRNN

Observations:

- Correctly classified skill for 7147 out of 8955 samples.
- Correctly classified non-skill for 286 out of 312 samples.
- 1808 skill was wrongly classified as non-skill.
- 26 non-skill were wrongly classified as skill.

	precision	recall	f1-score	support
0	1.00	0.80	0.89	8955
1	0.14	0.92	0.24	312
accuracy			0.80	9267
macro avg	0.57	0.86	0.56	9267
weighted avg	0.97	0.80	0.86	9267

Fig 56: CR for SimpleRNN

From fig 56 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 57%
- Macro Average Recall- 86%
- Macro Average F1-score- 56%

Through Hyperparameter tuning, an attempt was made to improve the performance of the SimpleRNN model.

4.3.2 Simple Recurrent Neural Network (SimpleRNN)- Hyperparameter Tuning

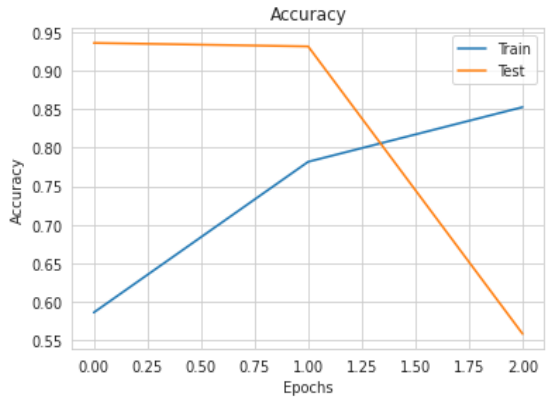
The hyperparameter tuned SimpleRNN model was trained by altering the *batch of size to be* 120 (60 earlier) and 20 *number of epochs(10 earlier)*

```
Epoch 1/20
348/348 [==============================] - 61s 170ms/step - loss: 0.6583 - acc: 0.5857 - val_loss: 0.3768 - val_acc: 0.9362
Epoch 2/20
348/348 [==============================] - 59s 169ms/step - loss: 0.4917 - acc: 0.7817 - val_loss: 0.2232 - val_acc: 0.9317
Epoch 3/20
348/348 [==============================] - 58s 167ms/step - loss: 0.3696 - acc: 0.8528 - val_loss: 0.7434 - val_acc: 0.5579
```

Fig 57: Training model SimpleRNN(Hyperparameter Tuning)

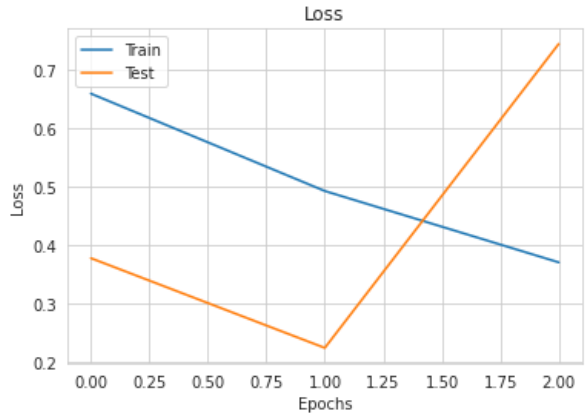
In addition to altering the model architecture by addition drop out layers and changing the batch size and number of epochs, Early Stopping callback was also used.

After 3 epochs the model stopped learning, the accuracy of the training data was observed as 85% while that of test data is 56%



For the first two epochs, the test accuracy for steadily high but after second epoch, a sharp fall was observed in the test accuracy while there is a steady increase in the training data accuracy.

Fig 58: Accuracy graph SimpleRNN(Hyperparameter Tuning)



For the test data, the loss value decreased for the first two epochs and then it had a sudden rise while a gradual decrease was observed over all the three epochs for

Figure 59: Loss value graph SimpleRNN(Hyperparameter Tuning)

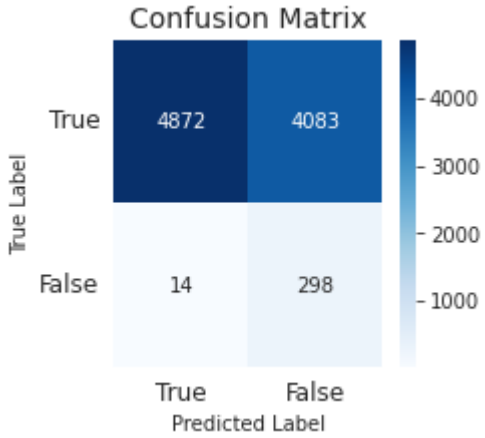


Fig 60: Confusion Matrix for SimpleRNN (hyperparamter Tuning)

Observations:

- Correctly classified skill for 4872 out of 8955 samples.
- Correctly classified non-skill for 298 out of 312 samples.
- 4083 skill was wrongly classified as non-skill.
- 14 non-skill were wrongly classified as skill.

	precision	recall	f1-score	support
0	1.00	0.54	0.70	8955
1	0.07	0.96	0.13	312
accuracy			0.56	9267
macro avg	0.53	0.75	0.42	9267
weighted avg	0.97	0.56	0.68	9267

Fig 61: CR for SimpleRNN(Hyperparameter Tuning)

From fig 61 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 53%
- Macro Average Recall- 75%
- Macro Average F1-score- 42%

4.4.1. Long and Short Term Memory (LSTM)

The LSTM model was trained for *batch of size* 60 and 10 *number of epochs*

```
Epoch 1/10
696/696 [==============================] - 501s 715ms/step - loss: 0.6933 - acc: 0.4972 - val_loss: 0.6990 - val_acc: 0.0337
Epoch 2/10
696/696 [==============================] - 495s 712ms/step - loss: 0.6932 - acc: 0.4999 - val_loss: 0.6991 - val_acc: 0.0337
Epoch 3/10
696/696 [==============================] - 485s 696ms/step - loss: 0.6932 - acc: 0.4969 - val_loss: 0.6915 - val_acc: 0.9663
Epoch 4/10
696/696 [==============================] - 489s 703ms/step - loss: 0.6932 - acc: 0.4960 - val_loss: 0.6951 - val_acc: 0.0337
Epoch 5/10
696/696 [==============================] - 494s 710ms/step - loss: 0.6932 - acc: 0.4951 - val_loss: 0.6884 - val_acc: 0.9663
Epoch 6/10
696/696 [==============================] - 492s 706ms/step - loss: 0.6932 - acc: 0.4960 - val_loss: 0.6907 - val_acc: 0.9663
Epoch 7/10
696/696 [==============================] - 482s 693ms/step - loss: 0.6932 - acc: 0.4944 - val_loss: 0.6905 - val_acc: 0.9663
Epoch 8/10
696/696 [==============================] - 474s 681ms/step - loss: 0.6932 - acc: 0.4950 - val_loss: 0.6941 - val_acc: 0.0337
Epoch 9/10
696/696 [==============================] - 473s 679ms/step - loss: 0.6932 - acc: 0.5015 - val_loss: 0.6843 - val_acc: 0.9663
Epoch 10/10
696/696 [==============================] - 475s 682ms/step - loss: 0.6932 - acc: 0.4966 - val_loss: 0.6894 - val_acc: 0.9663
```

Figure 62: Training model LSTM

After 10 epochs, the accuracy of the training data is 50% while that of test data is 97%

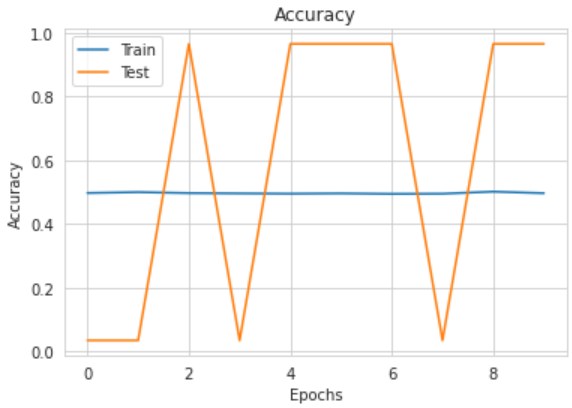
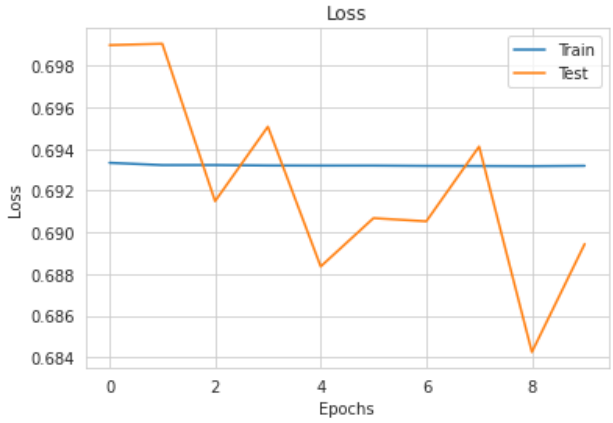


Fig 63: Accuracy graph LSTM

The accuracy for the training data has been nearly the same over all the epochs but a greater fluctuation was observed for the testing data where there were higher frequent changes for accuracy in test data.



The loss value for the training data has been exactly the same over all the epochs expect the first one. Moreover, a decreasing trend in the loss values for test data was observed despite the moderate fluctuation in the loss values.

Figure 64: Loss value graph LSTM

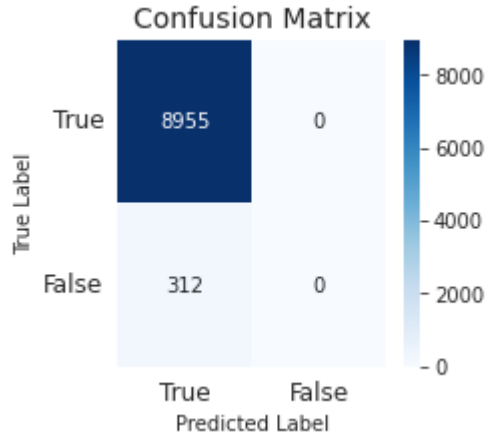


Figure 65: Confusion Matrix for LSTM

Observations:

- Correctly classified skill for 8955 out of 8955 samples.
- Correctly classified non-skill for 0 out of 312 samples.
- 0 skill was wrongly classified as non-skill.
- 312 non-skill were wrongly classified as skill.

	precision	recall	f1-score	support
0	0.97	1.00	0.98	8955
1	0.00	0.00	0.00	312
accuracy			0.97	9267
macro avg	0.48	0.50	0.49	9267
weighted avg	0.93	0.97	0.95	9267

Figure 66: CR for LSTM

From fig 66 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 48%
- Macro Average Recall- 50%
- Macro Average F1-score- 49%

4.4.2. Long and Short Term Memory (LSTM)- Hyperparamter Tuning

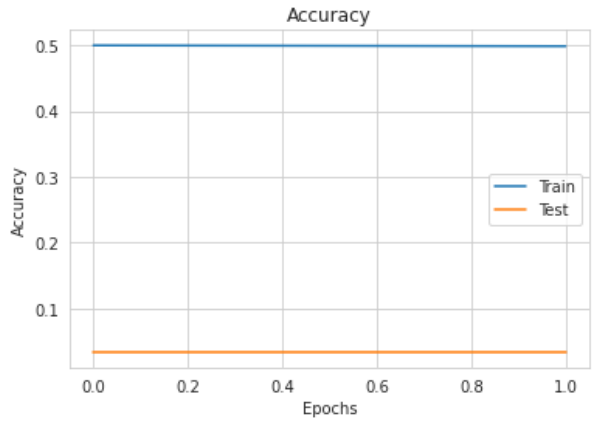
The hyperparameter tuned LSTM model was further trained by altering the *batch of size to be* 128 (60 earlier) and 10 *number of epochs(same earlier)*

```
Epoch 1/10
327/327 [==============================] - 235s 707ms/step - loss: 0.6934 - acc: 0.5000 - val_loss: 0.6978 - val_acc: 0.0337
Epoch 2/10
327/327 [==============================] - 231s 705ms/step - loss: 0.6933 - acc: 0.4988 - val_loss: 0.7133 - val_acc: 0.0337
```

Fig 67: Training model LSTM(Hyper parameter tuning)

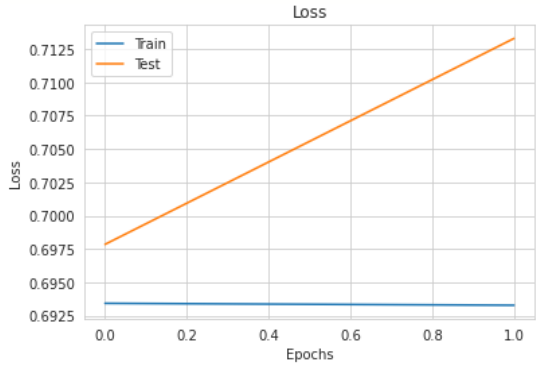
In addition to altering the model architecture by addition drop out layers and changing the batch size and number of epochs, Early Stopping callback was also used.

After 2 epochs the model stopped learning, the accuracy of the training data was observed as 50% while that of test data is 3%



The accuracy for the training data has been nearly the same over both the 2 epochs which is 50% while that of testing data has been constant around 3%.

Fig 68: Accuracy graph LSTM(Hyper parameter tuning)



The loss value for the training data has been nearly constant over both the 2 epochs. However, a small increase was observed for the loss value in testing data.

Fig 69: Loss value graph LSTM(Hyper parameter tuning)

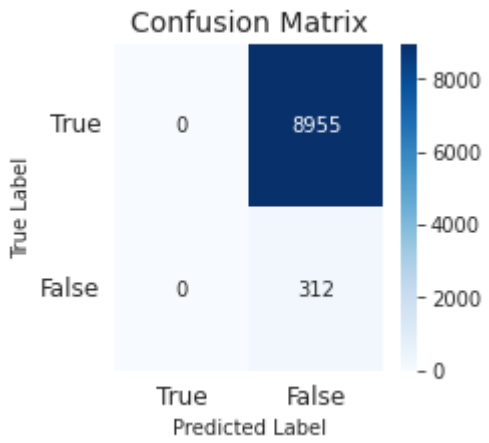


Figure 70: Confusion Matrix for LSTM (Hyperparameter Tuning)

Observations:

- Correctly classified skill for 0 out of 8955 samples.
- Correctly classified non-skill for 312 out of 312 samples.
- 8955 skill was wrongly classified as non-skill.
- 0 non-skill were wrongly classified as skill.

	precision	recall	f1-score	support
1	0.03	1.00	0.07	312
micro avg	0.03	1.00	0.07	312
macro avg	0.03	1.00	0.07	312
weighted avg	0.03	1.00	0.07	312

Fig 71: CR for LSTM(Hyper parameter tuning)

From Fig 71 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 3%
- Macro Average Recall- 100%
- Macro Average F1-score- 7%

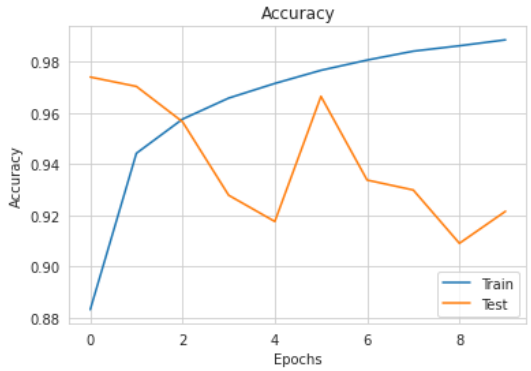
4.5.1. Convolutional Neural Networks (CNN)

The CNN model was trained for *batch of size* 60 and 10 *number of epochs*

```
Epoch 1/10
696/696 [==============================] - 35s 8ms/step - loss: 0.2691 - accuracy: 0.8831 - val_loss: 0.1019 - val_accuracy: 0.9740
Epoch 2/10
696/696 [==============================] - 6s 8ms/step - loss: 0.1436 - accuracy: 0.9442 - val_loss: 0.1056 - val_accuracy: 0.9703
Epoch 3/10
696/696 [==============================] - 6s 8ms/step - loss: 0.1110 - accuracy: 0.9575 - val_loss: 0.1465 - val_accuracy: 0.9565
Epoch 4/10
696/696 [==============================] - 6s 8ms/step - loss: 0.0888 - accuracy: 0.9658 - val_loss: 0.1418 - val_accuracy: 0.9278
Epoch 5/10
696/696 [==============================] - 5s 8ms/step - loss: 0.0739 - accuracy: 0.9715 - val_loss: 0.1806 - val_accuracy: 0.9176
Epoch 6/10
696/696 [==============================] - 5s 8ms/step - loss: 0.0615 - accuracy: 0.9766 - val_loss: 0.1376 - val_accuracy: 0.9664
Epoch 7/10
696/696 [==============================] - 6s 8ms/step - loss: 0.0518 - accuracy: 0.9806 - val_loss: 0.1532 - val_accuracy: 0.9337
Epoch 8/10
696/696 [==============================] - 6s 8ms/step - loss: 0.0429 - accuracy: 0.9841 - val_loss: 0.1689 - val_accuracy: 0.9299
Epoch 9/10
696/696 [==============================] - 6s 8ms/step - loss: 0.0370 - accuracy: 0.9862 - val_loss: 0.2477 - val_accuracy: 0.9090
Epoch 10/10
696/696 [==============================] - 6s 8ms/step - loss: 0.0313 - accuracy: 0.9885 - val_loss: 0.2082 - val_accuracy: 0.9215
```

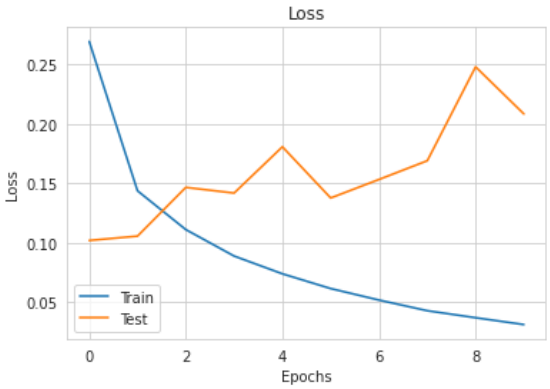
Figure 72: Training Model CNN

After 10 epochs, the accuracy of the training data is 99% while that of test data is 92%



The accuracy for the training data has been gradually increasing over all the epochs but a higher fluctuation was observed for the testing data from 5th to 7th epoch until it decreased at a moderate rate in the end.

Figure 73: Accuracy Graph CNN model



The loss value for the training data has been decreasing but for the test data it has shown an increasing trend.

Figure 74: Loss Graph CNN model

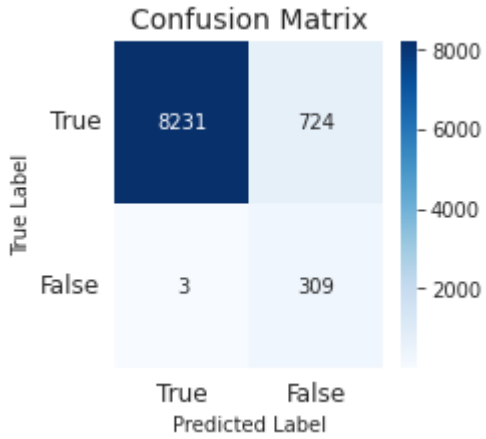


Figure 75: Confusion Matrix for CNN model

Observations:

- Correctly classified skill for 8231 out of 8955 samples.
- Correctly classified non-skill for 309 out of 312 samples.
- 724 skill was wrongly classified as non-skill.
- 3 non-skill were wrongly classified as skill.

	precision	recall	f1-score	support
0	1.00	0.92	0.96	8955
1	0.30	0.99	0.46	312
accuracy			0.92	9267
macro avg	0.65	0.95	0.71	9267
weighted avg	0.98	0.92	0.94	9267

Figure 76: CR for CNN model

From Fig 76 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 65%
- Macro Average Recall- 95%
- Macro Average F1-score- 71%

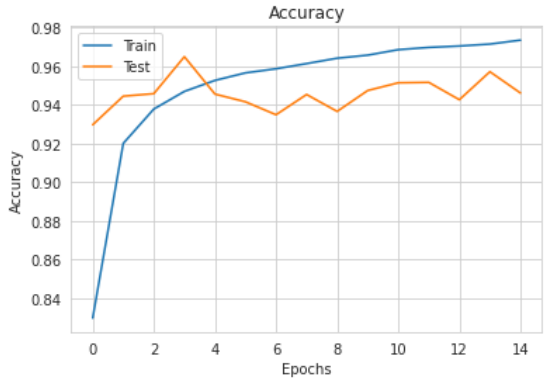
4.5.2. Convolutional Neural Networks (CNN)- Hyperparameter Tuning

The CNN tuned model was further trained for *batch of size* 128(earlier 60) and 15 *number of epochs(earlier 10)*

```
Epoch 1/15
327/327 [==============================] - 4s 10ms/step - loss: 0.3727 - accuracy: 0.8298 - val_loss: 0.2043 - val_accuracy: 0.9296
Epoch 2/15
327/327 [==============================] - 3s 9ms/step - loss: 0.1936 - accuracy: 0.9200 - val_loss: 0.1641 - val_accuracy: 0.9444
Epoch 3/15
327/327 [==============================] - 3s 9ms/step - loss: 0.1603 - accuracy: 0.9378 - val_loss: 0.1593 - val_accuracy: 0.9457
Epoch 4/15
327/327 [==============================] - 3s 9ms/step - loss: 0.1423 - accuracy: 0.9469 - val_loss: 0.1296 - val_accuracy: 0.9648
Epoch 5/15
327/327 [==============================] - 3s 9ms/step - loss: 0.1307 - accuracy: 0.9526 - val_loss: 0.1737 - val_accuracy: 0.9455
Epoch 6/15
327/327 [==============================] - 3s 9ms/step - loss: 0.1220 - accuracy: 0.9565 - val_loss: 0.1867 - val_accuracy: 0.9415
Epoch 7/15
327/327 [==============================] - 3s 9ms/step - loss: 0.1147 - accuracy: 0.9586 - val_loss: 0.2180 - val_accuracy: 0.9348
Epoch 8/15
327/327 [==============================] - 3s 9ms/step - loss: 0.1088 - accuracy: 0.9613 - val_loss: 0.1927 - val_accuracy: 0.9453
Epoch 9/15
327/327 [==============================] - 3s 9ms/step - loss: 0.1022 - accuracy: 0.9640 - val_loss: 0.2132 - val_accuracy: 0.9365
Epoch 10/15
327/327 [==============================] - 3s 9ms/step - loss: 0.0967 - accuracy: 0.9656 - val_loss: 0.1835 - val_accuracy: 0.9473
Epoch 11/15
327/327 [==============================] - 3s 9ms/step - loss: 0.0910 - accuracy: 0.9684 - val_loss: 0.1889 - val_accuracy: 0.9513
Epoch 12/15
327/327 [==============================] - 3s 9ms/step - loss: 0.0882 - accuracy: 0.9696 - val_loss: 0.1838 - val_accuracy: 0.9515
Epoch 13/15
327/327 [==============================] - 3s 9ms/step - loss: 0.0861 - accuracy: 0.9703 - val_loss: 0.2191 - val_accuracy: 0.9426
Epoch 14/15
327/327 [==============================] - 3s 9ms/step - loss: 0.0831 - accuracy: 0.9713 - val_loss: 0.1755 - val_accuracy: 0.9571
Epoch 15/15
327/327 [==============================] - 3s 9ms/step - loss: 0.0788 - accuracy: 0.9734 - val_loss: 0.2124 - val_accuracy: 0.9460
```

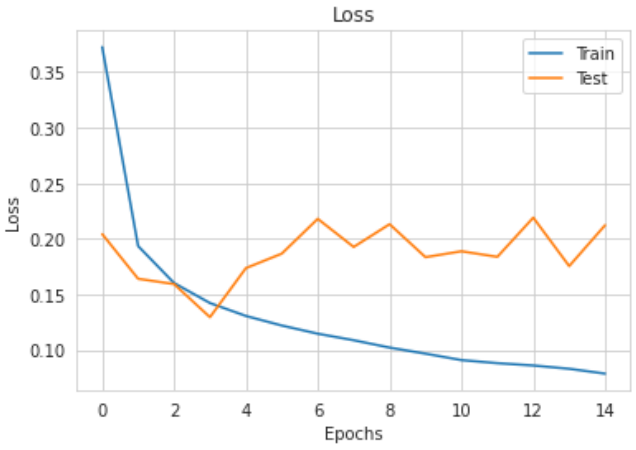
Fig 77: Training model for CNN(hyperparameter tuning)

After 15 epochs, the accuracy of the training data is 97% while that of test data is 95%



The accuracy for the training data has been gradually increasing over all the epochs from 82% to 97%. But, the accuracy for the test data has been moderately fluctuating over all the epochs and its accuracy was approaching near to the accuracy for training data.

Fig 78: Accuracy Graph for CNN(hyperparameter tuning)



The loss value for the training data has been gradually decreasing while moderate fluctuation was observed for the loss value in test data.

Fig 79: Loss graph for CNN(hyperparameter tuning)

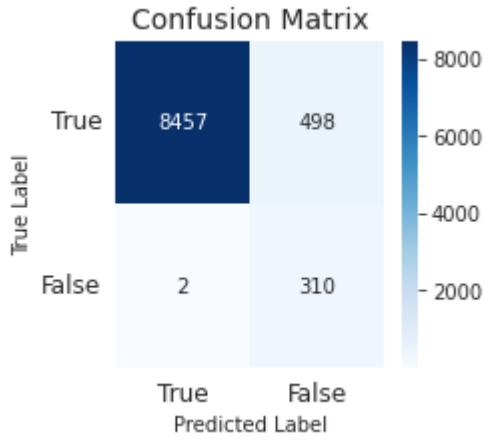


Figure 80: Confusion Matrix for CNN (Hyperparameter Tuning)

Observations:

- Correctly classified skill for 8457 out of 8955 samples.
- Correctly classified non-skill for 310 out of 312 samples.
- 498 skill was wrongly classified as non-skill.
- 2 non-skill were wrongly classified as skill.

	precision	recall	f1-score	support
0	1.00	0.94	0.97	8955
1	0.38	0.99	0.55	312
accuracy			0.95	9267
macro avg	0.69	0.97	0.76	9267
weighted avg	0.98	0.95	0.96	9267

Figure 81:CR for CNN(hyperparameter tuning)

From Fig 81 as macro average parameter is of our interest for our research objective,

- Macro Average Precision- 69%
- Macro Average Recall- 97%
- Macro Average F1-score- 76%

4.6 Model Performance Comparison

No.	Approach	Macro Avg F1-score
1	Topic Modelling	N/A
2	Word2Vec	N/A
3	Baseline	49%
4	Baseline(SMOTE)	33%
5	MultinomialNB	50%
6	MultinomialNB (Hyperparamater Tuning)	50%
7	SimpleRNN	56%
8	SimpleRNN(Hyperparamter Tuning)	42%
9	LSTM	49%
10	LSTM(Hyperparameter Tuning)	7%
11	CNN	71%
12	CNN(Hyperparameter Tuning)	76%

Figure 82: Comparison of Models

4.6.1. On the basis of performance

- To compare the performance among different models, Macro Avg F1 score was considered as a Key indicator since it considers the average precision and recall. Thus, LSTM model(Hyperparameter Tuning) holds the least score while CNN(hyperparameter Tuning) holds the greatest score. Moreover, it can also be claimed that Hyperparameter Tuning didn't improved the performance of LSTM.
- The Macro Avg F1 score for Multinomial Naïve Bayes is same despite of Hyperparameter Tuning that us 50% which depicts that there is no significant effect of optimizing the parameters for Multinomial NB model.
- The Macro avg F1 score for Baseline model was not considered for comparison with other models since it was overfitting and SMOTE algorithms was implemented due to class imbalance.
- The Macro avg F1 score for Baseline(SMOTE) was considered as a baseline for other models. Though, it had lesser score as compared to the baseline model but it had greater value for the macro avg F1 score for the minority class(1 or job_skill) which is 7% as compared to baseline which was 0%.
- The Macro avg F1 score for SimpleRNN(Hyperparameter Tuning) model is less as compared to SimpleRNN and thus we can claim that hyperparameter tuning didn't had positive impact of the performance of the SimpleRNN model. Instead, it resulted in decline in its performnace. Moreover, the SimpleRNN model has higher macro avg F1 score for minority class which is 24% as opposed to SimpleRNN(model) which is 13%.
- Very less skills were extracted from topic Modelling while comparatively more skills were extracted from Word2vec model but also, the Word2Vec model was accompanied with noise.
- Only CNN(hyperparamter tuning) model has shown improvement with the optimization of its hyperparameter as the macro avg f1 score is 76% as compared to the CNN model which is 71%. Adding to it, it has also shown improvement in the scores for minority class which is 55% as compared to CNN model that is 46%.

The following skills were labelled in the model:

'python','excel', 'database','queries','dashboards','plsql','modelling','tsql','integration','crm','salesforce','algorithm','impala','security','r','c','sas','bi','sspowerbi','power bi','powerpoint','pipeline','pandas','ssrs','ssis','alteryx','methodologies','visualization','powershell','sqldb','languages','qa','java','cloud','mssql','nosql','linux','dax','jira'

4.6.2. On the basis of extracting new skills from unseen(test) data

- Baseline model seem to extract noise as compared to extracting relevant technical skills as it labelled irrelevant phrases in my model as skill like- global insurance company, candidates, working, company culture, fantastic team, birthday etc.
- Multinomial NB extracted noise as well, but it seems to label the technical words in my model as skill even if they are not domain specific. For Instance: digital analytics software, data science degree, data migration testing, data privacy environment, data analyst role, datadriven culture passion, etc.
- Starting with SimpleRNN as the first model for Deep Learning, it was observed that the model labelled the domain-specific skills related to Data Analytics more as compare to the previous models. For Instance: the extracted skills were: Microsoft office suite, hr analytics, data sourcing, data management, application testing, risk analytics, etc.
- LSTM didn't perform well at all as it labelled no records as job_skill.
- CNN model outperformed all the other models that were considered for this research project as it identified most domain-specific skills. For Example: Vlookup, g suite documentation, Cassandra, kafka, statistics, correlation, relational database management systems, statistical analytical tools.

CHAPTER 5: Conclusion

5.1. Limitations

- The data collection to extract the job Advertisement from Reed website during the initial phase of the project was done using BeautifulSoup library in Python. However, this couldn't be implemented well in our project as frequent change in HTML was observed in the website.
- The word2vec model was implemented so that it could identify new technical skills, but in our case this model has been more inclined towards predicting the relevant words instead of correctly identifying new technical skills. Thus, Word2Vec extracted more noise than useful abilities, which makes the output unappealing. While the extracted keywords holds meaningful information, but the result included a lot of noise, making it impossible to separate essential abilities from the noise. Therefore, in order to detect new technical skill sets, we must split down our dataset into phrases and train the model to learn the skill sets by labelling the training data set.
- The initial Baseline model had imbalanced data, so SMOTE algorithm was applied to reduce overfitting caused by imbalance of the data. Still, it was observed that not many models showed improvement in performance even after balancing the data.
- Omitting duplicates in the training data might have overall improved the performance of other models. But, they were not omitted as that will skew the base rate of each distinct object. If the training data are a representative sample of the real world, then its unlikeable, because it will be then preferable to train the data for a slightly different world (one with different base rates).

Considering the following scenario in which there are only two unique items. The original data had 99 instances of object A and one instance of object B. After removing duplicates, only one item A is left and one object B. Therefore, A classifier trained using de-duplicated data will differ significantly from one learned on the original data.

5.2. Conclusion of the study

The main aim of this project was to expand on the work done by Sharma(2019) to extract domain-specific job skills using Machine and Deep Learning approach. Therefore, the study conducted by Sharma(2019) was expanded through implementing a baseline model so that the performance of other models can be compared against the baseline model. Thus, it was observed that all the models performed better that Baseline model (which was evaluated by macro F1 score and extracting new skills from the unseen data) except LSTM model In addition to this, SimpleRNN and CNN model was also implemented apart from LSTM as deep learning approach and it was found that CNN model proved to be best as compared to other models as it identified the skills from the unseen data on which it was never been trained. This can also be interpreted as CNN model being an intelligent model, preserved the semantic relationship and was successful in extracting the domain-specific skills as compared to other Machine Learning and Deep Learning models.

References

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems, 3111–3119.

ESCO: European Skills, Competences, Qualifications and Occupations https://ec.europa.eu/esco/ . Accessed 17 Dec 2015. Madely du Preez

Bastian, M.; Hayes, M.; Vaughan, W.; Shah, S.; Skomoroch, P.; Kim, H.;Uryasev, S.; and Lloyd, C. 2014. LinkedIn Skills: Large-Scale Topic Extraction and Inference. In Proceedings of the 8th ACM Conference on Recommender Systems, 1–8. New York: Association for Computing Machinery.

Phuong, H., Mahoney, T., Javed, F., McNair, M., 2018. Large-Scale Occupational Skills Normalization for Online Recruitment. AI MAGAZINE. Association for the Advancement of Artificial Intelligence. ISSN 0738-4602

Sharma, N. Job Skills extraction with LSTM and Word Embeddings. University of Technology Sydney - UTS, Sydney, Australia. Sept 2019.

Gugnani, A. Implicit Skills Extraction Using Document Embedding and Its Use in Job Recommendation. Conference: AAAI - Innovative Applications of Artificial Intelligence (IAAI). February 2020.

Gugnani, A.; Kasireddy, V. K. R.; and Ponnalagu, K. 2018. Generating unified candidate skill graph for career path recommen- dation. In 2018 IEEE International Conference on Data Mining Workshops (ICDMW), 328–333. IEEE.

Watson Natural Language Understanding. https://www.ibm.com/cloud/watsonnatural-language-understanding. Accessed 15 Oct 2020.

MediaWiki API help. https://en.wikipedia.org/w/api.php Accessed 15 Oct 2020.

Coursera. 2021. [online] Available at: <https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python> [Accessed 4 August 2021].

Google Developers. 2021. *Python Installation - Colab Notebook | Google Earth Engine*. [online] Available at: <https://developers.google.com/earth-engine/guides/python_install-colab> [Accessed 1 July 2021].

Bonner, A. 2021. *Getting Started With Google Colab*. [online] Available at: <https://towardsdatascience.com/getting-started-with-google-colab-f2fff97f594c> [Accessed 10 July 2021].

Medium. 2021. *Topic modeling using Latent Dirichlet Allocation(LDA) and Gibbs Sampling explained!*. [online] Available at: <https://medium.com/analytics-vidhya/topic-modeling-using-lda-and-gibbs-sampling-explained-49d49b3d1045> [Accessed 4 August 2021].

Machine Learning Plus. 2021. *Topic Modeling in Python with Gensim*. [online] Available at: <https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/#12buildingthetopicmodel> [Accessed 8 August 2021].

Hussain, F. and profile, V., 2021. *NLP DEEP NLP*. [online] Fahadhussaincs.blogspot.com. Available at: <https://fahadhussaincs.blogspot.com/p/nlp-deep-nlp.html> [Accessed 12 August 2021].

Stack Abuse. 2021. *Text Classification with Python and Scikit-Learn*. [online] Available at: <https://stackabuse.com/text-classification-with-python-and-scikit-learn/> [Accessed 23 August 2021].

Machine Learning Plus. 2021. *Topic Modeling in Python with Gensim*. [online] Available at: <https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/#12buildingthetopicmodel> [Accessed 2 September 2021].

Medium. 2021. *A Beginner's Guide to Word Embedding with Gensim Word2Vec Model*. [online] Available at: <https://towardsdatascience.com/a-beginners-guide-to-word-embedding-with-gensim-word2vec-model-5970fa56cc92> [Accessed 6 September 2021].

Stack Abuse. 2021. *Python for NLP: Parts of Speech Tagging and Named Entity Recognition*. [online] Available at: <https://stackabuse.com/python-for-nlp-parts-of-speech-tagging-and-named-entity-recognition/> [Accessed 12 September 2021].

Mishra, U., 2021. *Binary and Multiclass Classification in Machine Learning | Analytics Steps*. [online] Analyticssteps.com. Available at: <https://www.analyticssteps.com/blogs/binary-and-multiclass-classification-machine-learning> [Accessed 16 September 2021].

Medium. 2021. *Building a Basic Binary Text Classifier using Keras*. [online] Available at: <https://medium.com/nerd-for-tech/building-a-basic-binary-text-classifier-using-keras-4972a7c36616> [Accessed 19 September 2021].

Li, D., Hasanaj, E. and Li, S., 2021. *3 - Baselines*. [online] Machine Learning Blog | ML@CMU | Carnegie Mellon University. Available at: <https://blog.ml.cmu.edu/2020/08/31/3-baselines/> [Accessed 24 September 2021].

Medium. 2021. *Always start with a stupid model, no exceptions.*. [online] Available at: <https://blog.insightdatascience.com/always-start-with-a-stupid-model-no-exceptions-3a22314b9aaa> [Accessed 11 September 2021].

Subscription.packtpub.com. 2021. Simple Recurrent Neural Network. [online] Available at: <https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788292061/6/ch06lvl1sec41/simple-recurrent-neural-network> [Accessed 4 October 2021].

Subscription.packtpub.com. 2021. Simple Recurrent Neural Network. [online] Available at: <https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788292061/6/ch06lvl1sec41/simple-recurrent-neural-network> [Accessed 4 October 2021].

Long Short Term Memory. https://en.wikipedia.org/wiki/Long_short-term_memory 02 Oct 2021.

Medium. 2021. Intuitive guide to word embedding, RNN (SimpleRNN, LSTM) with step by step implementation in keras…. [online] Available at: <https://hemantranvir.medium.com/spam-detection-using-rnn-simplernn-lstm-with-step-by-step-explanation-530367608071> [Accessed 4 October 2021].

kernel_size, K. and Möller, D., 2021. Keras conv1d layer parameters: filters and kernel_size. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/46503816/keras-conv1d-layer-parameters-filters-and-kernel-size> [Accessed 4 September 2021].

Medium. 2021. Text classification using CNN. [online] Available at: <https://medium.com/voice-tech-podcast/text-classification-using-cnn-9ade8155dfb9> [Accessed 4 October 2021].

Keras?, H. and Pokkalla, H., 2021. How does the Flatten layer work in Keras?. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/44176982/how-does-the-flatten-layer-work-in-keras> [Accessed 4 October 2021].

objects?, S., Abraham, S. and Dodier, R., 2021. Should I keep/remove identical training examples that represent different objects?. [online] Stack Overflow. Available at: <https://stackoverflow.com/questions/26197700/should-i-keep-remove-identical-training-examples-that-represent-different-object> [Accessed 4 October 2021].

Kaggle.com. 2021. RNN for spam detection. [online] Available at: <https://www.kaggle.com/kentata/rnn-for-spam-detection> [Accessed 4 October 2021].

TensorFlow. 2021. tf.keras.callbacks.EarlyStopping | TensorFlow Core v2.6.0. [online] Available at: <https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping> [Accessed 4 October 2021].

Medium. 2021. Don’t Overfit!—How to prevent Overfitting in your Deep Learning Models. [online] Available at: <https://towardsdatascience.com/dont-overfit-how-to-prevent-overfitting-in-your-deep-learning-models-63274e552323> [Accessed 4 September 2021].

Quora. 2021. My testing accuracy is 97% and training accuracy is 99%. Is my model over fitting? Why is it performing so well? I have computed other me.... [online] Available at: <https://www.quora.com/My-testing-accuracy-is-97-and-training-accuracy-is-99-Is-my-model-over-fitting-Why-is-it-performing-so-well-I-have-computed-other-metrics-and-they-are-giving-good-results> [Accessed 4 October 2021].