# Task 1

In an ever-competitive job market, finding employment can be quite difficult. Finding a suitable employment opportunity can become quite time consuming for job seekers with a medium sized posting having an average of 550 words (Jennifer Gladstone, 2017). On average, it takes approximately 15 applications to land a job interview, and around 10 interviews to secure a single job offer. This means that an applicant would need to apply for 150 job positions before receiving a single job offer (Chakrabarti et al., 2019). If each job listing takes three minutes to read, a job seeker would need to spend 5 hours reading listings, plus additional time to complete the employer’s application process.

Currently, job seekers must scour through multiple sites worth of job postings and read through entire the entire listings in order to determine if they possess the required skill set whilst also ensuring the day-to-day tasks they will be undertaking are suitable for their career aspirations.

Job seekers would benefit significantly from a central repository of job listings, with each listing containing a summarization, that is restricted to a handful of sentences, and metadata tags of the required skills. This repository would streamline the job application process and enable applicants to apply for jobs at a notably faster rate by reducing the amount of time spent reading applications and filtering out listings that require skills the candidate does not possess.

A web-crawler could be employed to scrape through multiple sites that contain job listings, extracting the job title, job position and job description from each article. By utilising a web crawler that crawls multiple sites, the process of job listing collection can be automated and enable the central repository to contain a large quantity of employment opportunities.

By making use of Natural Language Processing tasks, the central repository can provide skill metadata tags and listing summarizations. For this use case, two tasks will be employed:

* Summarisation through the use of extraction-based summarisation NLP techniques
* Skill keyword extraction through the use of a Long Short-Term Memory (LSTM) deep learning network and word embeddings.

# Task 2

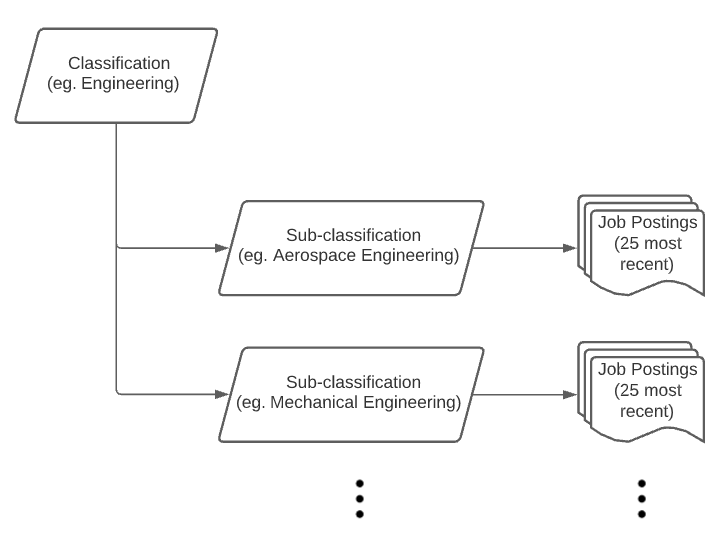
1. In total, two websites will be crawled. The first will Seek.com.au, a website which focuses on facilitating the match between jobseekers and employment opportunities and helping hirers find candidates for advertised roles (*About SEEK*, 2011). Seek has the ability to filter job listings according to the job classification, the job sub-classification, the job location, as well as the job work type (full time, part time, etc) and the pay.

The second site to be crawled will be thebalanacecareers.com. This website provides numerous articleson a wide range of topics ranging from finding a job and human resources to succeeding in the workplace. These articles are carefully curated by a select group of industry experts with experience in job searching, resume writing, salary negotiations, and other career planning topics (*About Us*, 2018).

1. Seek.com.au was chosen as the primary site containing job postings to be extracted as it is the largest provider of this service in the Asia-Pacific region with exposure to over 2.9 billion people and relationships with over 1 million hirers.

Thebalancecareers.com was chosen to be crawled as it contains a plethora of job skills on its site, curated by a series of experts in the field that remain up to date on the current job trends and their associated requirements.

1. As this task is only a prototype, the job listing coverage is limited to only advertisements posted to SEEK, further restricted to the 25 most recent posts per sub-classification.



As of writing, Seek lists nearly 3,000 job posts for the Developers/Programmers sub classification of the parent Information & Communication Technology classification. By restricting the number of job listings per sub-classification to the 25 most recent, the resulting data set is of a much more manageable size. There also exists a plethora of other sites that contain job postings, including but not limited to: jobsearch.com.au, careerone.com.au, adzuna.com.au, indeed.com.au. These previously listed sites also restrict the job location to Australia. If one were to create a world-wide central repository, the crawler would need to crawl through an unknown but presumably large number of sites that exist for this purpose worldwide.

Balance Careers provides a large number of employment requirements from their handful of experts in the field, but these requirements they provide are purely a small sample of the total number of skills required. Due to the ever-evolving nature of careers and their required skill sets, choosing just a single site to represent all of the possible skills is severely restricting. Using multiple sites, or by employing industry experts that can stay up to date on current and emerging competencies.

1. The layout of seek.com.au is relatively simple. The search filters are all stored in a html section tag which can easily be extracted through the use of the BeautifulSoup4 (BS4) python package. The selected filters are then used to generate a URL to show results that meet the filters’ criteria. The query results are returned in a paginated format with each page containing around 20 job listings. This pagination hurdle can be solved by appending the *?page=x* parameter to the query URL and incrementing the *x* until we extract the maximum allowed listings of 25. A loop can then be employed to iterate over each article on each page and extract the link to the full listing. Once on the single listing’s page, we aim to extract the job description which can be located in three different elements depending on the layout of the listing. The description can be in a div element with the ‘data-automation’ attribute set to either ‘jobDescription’ or ‘jobAdDetails’. If the description cannot be found in either of those elements, it will be located in the next div element after the job title.

Only a single page from thebalancecareers will be crawled, <https://www.thebalancecareers.com/employment-skills-listed-by-job-2062389>. This web page contains a plethora of possible job titles, divided by their industry sector, with each job title actually being a link to a dedicated page for this profession. Each individual professions’ page is laid out differently to each other, but each share the fact that they all contain multiple dot point lists of skills. These skill lists are stylised as unordered lists with each list element having no class. There also exists paragraphs of content elaborating on these skill lists, but these paragraphs prove to be of no value for the task at hand. By employing BeautifulSoup once again, the content of each skill list can be extracted and subsequently written to a file.

1. The robots.txt file present in the majority of websites contains the URLs that are not allowed to be crawled. In this case, both sites’ robot.txt files shows that all the URLs present on the site and are allowed to be crawled. To prevent copy right issues, the logos from the individual companies will not be stored.
2. The job descriptions metadata will be supplemented with the title, classification, and sub-classification. The complete skill list, created by appending each individual professions’ list of require skills obtained from thebalancecareers, will be used in conjunction with a manually labelled data set to train a Recurrent Neural Network (RNN) on what a skill is. This RNN will then be used to extract all the skills contained in the job descriptions.
3. The SEEK web crawler will take the parent job description element and combine all the children p tags into a single string. This string will be stored in a csv file under the heading DESCRIPTION. The job title can be obtained by casting the HTML pages’ title attribute to a string and then storing it in the csv file under the TITLE heading. The classification and sub-classification will be available as variables in the workspace but can also be obtained from the webpages’ URL.

The crawler for thebalancecareers will append the unique skills found from each individual profession page to a single skill. This skill list will be written to a csv file with the skills being under the TEXT heading. An additional column titled TARGET will also be written the value of ‘1’ for each skill. This TARGET column will be utilised when training the RNN.

1. CODE
2. SCREENSHOTS
3. The Seek web crawler will save its initial data in a comma separated file with four columns:

* FIELD
* SUB-FIELD
* TITLE
* DESCRIPTION

Processing and cleaning of this retrieved data will be achieved through the use of python’s re package and python’s standard replace function. The re package is used as it provides regular expression matching operations that can be combined with the substitute function to replace the matched pattern. In this case, single and double quotes will be removed from the job descriptions to assist in the NLP tasks. The newline character (\n) is present in the extracted HTML as the job descriptions have been formatted before publishing and will be replaced with a single space. The Unicode replacement character (\ufffd) was found to be causing issues when writing to a csv file. This character was therefore removed from the dataset.

As the field and sub-field values are taken from the constructed URL during the web crawl, they must undergo some processing to be the correct value for the NLP tasks. Seek formats the field value as ‘job-in-FIELD’ eg ‘jobs-in-accounting’. Extracting the field is done with a simple regular expression to remove the “jobs-in-“ section. Due to the job title being taken from the HTML web page, a “ – seek” is appended on the end. This can also be removed with a simple regular expression.

The crawler for thebalancecareers returns clean data due to the skills being extracted being plain text inside p tags. For consistency, all of the skills are converted to lowercase before being stored in a csv file.

1. Exploring the scraped Seek data first, we can first look at the shape of the data and see that the crawler retrieved 8866 jobs. Next we can look at the mean job description word count.

After that, we can have a more in depth look into the job description word count per classification. Utilising the pandas boxplot function, we can visualise the minimum, first quartile, median, third quartile, and maximum job description lengths per classification.

We can then plot a histogram of the job description word length to better understand the overall distribution. From the plot, it appears the data is normally distributed around the mean of 373 words. As the data set has over 1000 observations, this can be attributed to the central limit theorem.

Exploring the retrieved skills data set next, we can see we successfully scraped 1954 skills from thebalancecareers.

# Task 3

## NLP 1 – Text summarizer

1. Lit review

Extraction based text summarisation can be completed through a number of approaches. One such system, TextRank, proposed in Sehgal et al (2017) modifies Google’s PageRank algorithm to extract useful and important phrases from the available text. This paper evaluates the algorithm through the ROGUE 2.0 Evaluation toolkit and achieves the following results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Rogue Type | Task Name | Average Recall | Average Precision | Average Fscore | Number reference summaries |
| ROGUE 1 | Sample 1 | 1.0 | 0.29664 | 0.45732 | 1 |
| ROGUE 1 | Sample 2 | 1.0 | 0.09125 | 0.16841 | 1 |
| ROGUE 1 | Sample 3 | 1.0 | 0.33504 | 0.50192 | 1 |
| ROGUE 1 | Sample 4 | 1.0 | 0.44071 | 0.61180 | 1 |

While efforts had been made to extract a meaningful and coherent summary

from the article, there is still a lot of scope of improvement as to how the sentences are extracted and whether they take the summary to its logical meaning. Another possible approach is to use k-means clustering as proposed in Shetty and Kallimani (2017). This paper proposes a three-step unsupervised extraction approach consisting of:

* + Document tokenization
  + Compute sentence score
  + Apply Centroid Based Clustering on the sentences and extract important sentences as part of summary

The paper does not measure the accuracy of the summarizer and instead compares their output to a human written abstractive summary. They state that their approach provides more favourable results than current state-of-the-art approaches such as TextRank, ranking their proposed approach in second place to a human approach, very close to the first place.

1. Rationale for selection of NLP task

With the recent surge in the amount of online content available to the general population, a fast and effective automatic summarization has become more important. By summarizing content with a large amount of data down to a few sentences, users are able to access the most important aspects of the content without having to sift through redundant and insignificant information. This proves key in reducing the amount of time taken reading job applications, allowing the prospective applicant to read a quick summarization of the post without needing to read the whole thing.

An extraction-based approach was chosen as opposed to an abstractive-based summariser due to the simplicity of the former method. The extraction-based approach also retains the writing style and nuances of the original job description which can provide context that would be lost with an abstractive text summariser.

1. The input to the summarizer will be the job description, with the output being an arbitrary amount of the most impactful sentences as determined by the extractor. As the job description had been cleaned previously, there is no additional pre-processing of the input required. As the output is just text sentences extracted from the input, is requires no additional cleaning or processing before the summary can be stored alongside the rest of the data in a csv file.
2. As this NLP task is not a machine learning task, no hyperparameter is used. Judging the effectiveness of the summarizer will be down to pure human judgement as this task is only a prototype. Further research can be done into what metrics can be applied and how to apply them to this task.
3. To test the summarizer, we can first test it on a single job ad to gauge its effectiveness. Further to that we can then run the summarizer for each job in order to find the average length of the summary compared to the job posting.

From the output, the summary lacks coherence as it is simply extracting impactful sentences and does not try to develop a new paragraph based on the data, a task more suitable to an abstractive -based summarizer. We can also see the summary has an average length of 70 words, a 453% reduction from the average 317 words in a job posting.

While the summary lacks coherence, it’s quality is more than suitable for this prototype task and can be applied to the whole dataset, with the summary saved in a column titled “SUMMARY”.

## NLP 2 – LSTM RNN Keyword Extraction

1. Lit review

Job candidates obtain skills through formal education, vocational education, internships, on-the-job training, or experience from previous occupations. The key function of a job search engine is to assist in the matching of the candidates skills to jobs that also require a similar skill set. A common approach while doing a skill match is to use standard keyword matching

or information retrieval framework as explained in (Salton & Buckley, 1988). A few challenges with this approach include:

* + The skill may be referenced in many different forms or synonyms (e.g., OOP, Object Oriented Programming)
  + Some skills may not be explicitly stated in the job description, but industry knowledge would dictate experience with the stated skill requires experience with the unstated skill (e.g. Experience with python denotes the candidate would require experience with Object Oriented Programming)
  + A skill dictionary would quickly become outdated as new skills from unseen and emerging domains appear.

A framework for skill extraction and normalization was proposed in (Sharma, 2019). This paper proposes the use of a Recurrent Neural Network subtype of an artificial neural network in combination with word embeddings to solve the problems encountered with a static skill dictionary. The system first extracts noun phrases from job descriptions before applying a Long Short-Term Memory (LSTM) deep learning network combined with word embeddings to extract the relevant skills from the text. The authors were able to achieve a test accuracy of 76.58% by restricting the job domain to jobs Data Science category. The method proposed in this paper is limited by the use of noun phrases for the core training dataset. As many job posts are represented by verb phrases, a new training set and model must be developed to extract the phrases.

1. Rationale for selection of NLP task

Job skills are the common link between job applications, applicant resumes and job listings by companies. Identifying skills in job postings is a significant problem and can provide a pathway for job seekers and hiring organisation. By ‘tagging’ each job listing with the required skills and enabling users to filter jobs by these skills would drastically improve the job search process.

1. The input to the skill extractor will be noun phrases as defined in by the following grammar:

NBAR:

{<NN.\*|JJ>\*<NN.\*>}

NP:

{<NBAR>}

{<NBAR><IN><NBAR}

These noun phrases will be extracted from each job description and fed through the RNN with the output being a float value from 0 to 1 representing the probability that the noun phrase is a skill. If the output of the RNN is greater than 0.5, the phrase is determined to be a skill and is appended to a list of skills for that job description. This list is then saved into a column in the dataset’s csv file titled “SKILLS”.

1. For the specified problem, there are multiple classes (skill and not\_skill) but only one of the classes can be present at a single time. As such, the softmax activation function was chosen as it enables the model to interpret the output as probabilities.

The implemented RNN model also uses the adaptive moment estimation, adam, optimizer from keras which uses default values of:

* 1. Learning rate = 0.001
  2. Beta\_1 = 0.9
  3. Beta\_2 = 0.999
  4. Epsilon = 1e-7

Keras also offers multiple metrics to judge the model. In our case, the accuracy of the models prediction will be used in order to provide a proper comparison to past literature.

1. Utilising an industry standard 80/20 split on the labelled data along with a 5-fold cross validation, we can judge the accuracy of the model. The average accuracy of the model after the cross validation is 71.29%. Whilst less accurate than models from previous literature, this model is not restricted to covering a single job industry and proves to work well enough for a prototype task.

After proving sufficiently accurate, the model was used to extract the skills from each description in the seek dataset.