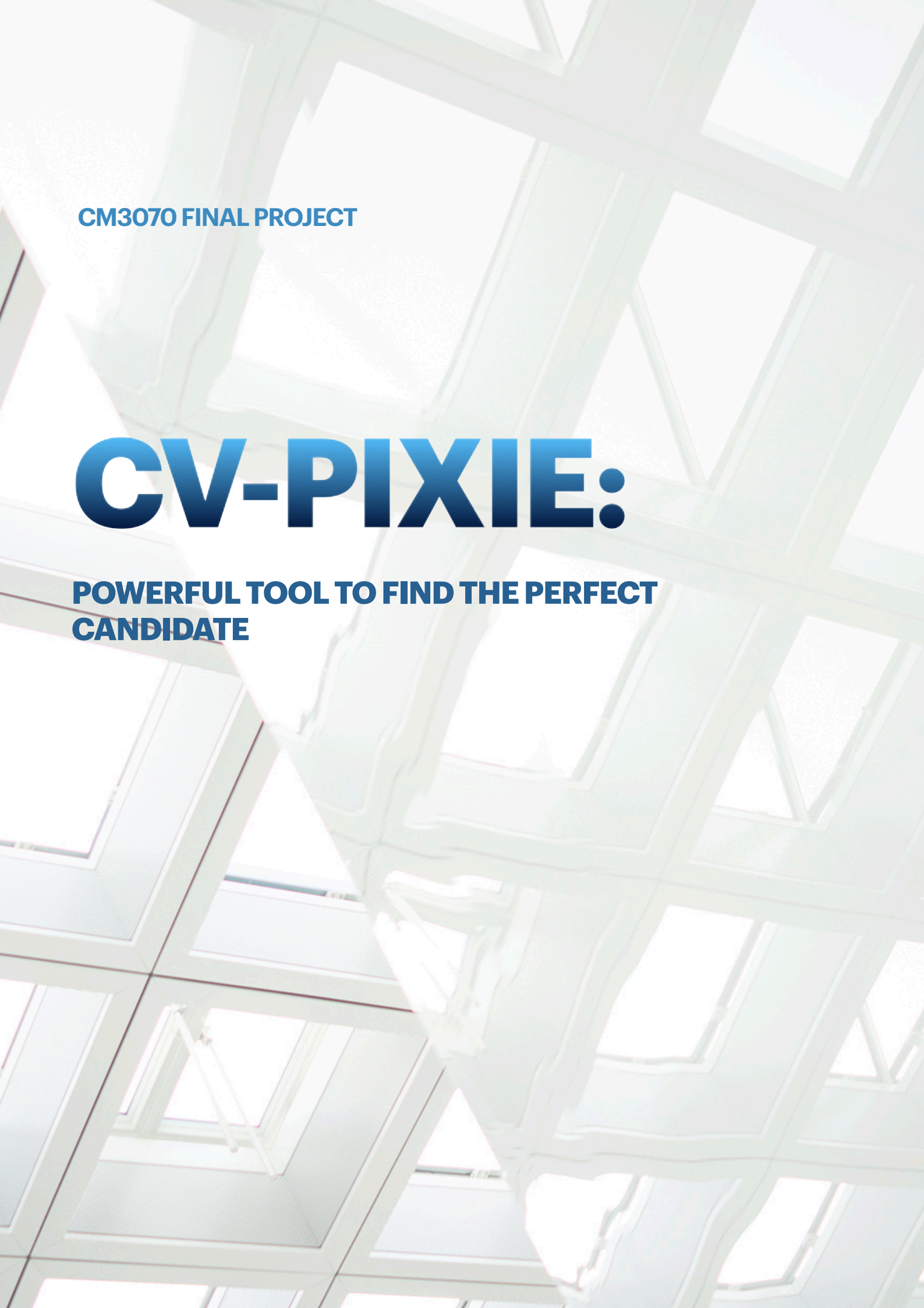
CM3070 FINAL PROJECT

CV-PIXIE:

POWERFUL TOOL TO FIND THE PERFECT CANDIDATE



1. Introduction

Title : CV-Pixie : Powerful tool to find the perfect candidate

Selected Project Template: Automated search strategy generation

Github Repository: https://github.com/holyjc827/final_project

Background and context:

The project aims to generate a search query or strategy which helps to find the most ideal candidate for a job. Lately, it is often said that the market for some jobs is incredibly competitive. This means that finding the best, most appropriate candidate adds an extra burden to companies. The ideal candidate means different things to different companies. Some companies are looking for candidates that are in a specific location, or with certain qualification, and so on. So, the project is trying to build an application which helps the company to do the hard, but incredibly important task for them.

The goal of the project and motivation

As stated before, the goal for this project is to help the company to find the best candidate in the overheated job market. The recruitment process is generally risky for every company, as a resume screening and a series of interviews may not lead to the perfect candidate. In order to reduce the risk, the company should try to be as thorough as possible with the candidate screening.

Since what constitutes “the ideal candidate” might be different for each company, for each position, or even for each hiring cycle, it is imperative for the project to be capable of adjusting parameters, so it generates the most appropriate search strategy depending on what the company is looking for at that particular moment. Also, reversely, this project could be used to implement the complete blind screening, which means the process through which the company screens candidate without adding the individual recruiter’s personal biases about certain area, such as gender and race.

Technical details

The project intends to use several technical stacks, which utilizes a combination of different techniques and ideas. Some of them are also state-of-the-art and experimental.

Firstly, the application will be written in Python. Python is quite useful for this project for several different reasons. One of the most important characteristics of Python is that it is easy to learn, while being performative. It is considered as an interpreted language, which means that it supports convenient features, such as dynamic typing. This makes easier for people to learn, while showing a great level of performance. Also, Python has a lot of libraries that are useful for data processing and natural language as well. This is particularly beneficial for this project.

Secondly, boolean search technique is required. When it comes to SEO, Search Engine Optimization, boolean search technique is actively used. This means that choosing the right keywords must be chosen. Although choosing the right keywords is not done by boolean search technique, it is where those keywords can generate the right search query string.

Thirdly, the project will actively use natural language process and machine learning. The application will receive a job description, which is a text, and process it. The processing part will

actively use NLP techniques, such as lemmatization and stemming. Machine learning will be applied after those texts are processed. By doing that, the more examples the application processes, the better accuracy we can anticipate.

2. Literature Review

I. Introduction

This literature review is for the project based on the provided template "Automated search strategy generation". The idea is to build an application which receives a job description and provides the search query string that is the most appropriate to find the ideal candidate for the job. The application is a python CLI (command-line interface) application, so everything will be written in Python. This literature review will explore a number of different examples which have a similar objective as this project, and those examples will run the whole gamut from proprietary applications to academic journals.

II. Assessment of outcomes

As discussed before, the output of the application will be a search query string which can be used to find the most ideal candidate according to a given job description. From this, the outcomes can be assessed several different ways: the quality of keywords and the accuracy of search result.

- Quality of keywords

  Good keywords are important to build a search query string. Based on the quality of keywords, the quality of search result will be determined. For this project, keywords will be extracted from a job description. Er. Tanya Gupta says that automated keyword extraction from a text is very imperative for many industries because it is almost impossible to manually go through a large amount of data [Gupta 2017]. This signifies that the application should be able to understand that certain keywords are more important than the others in the text. Depending on how qualitatively these keywords are extracted, the better the application serves its purpose.

  According to Rachel Leist, the quality of keywords can be determined by three criteria: relevance, authority, and volume. Relevance is self-explanatory: the keywords must be relevant to the result that one is looking for [Leist 2022]. Authority means that the keywords must be authoritative and legitimate. They should be able to deliver a clear meaning and be understood by a broad range of viewers, which effectively excludes a type of words, such as slang [Leist 2022]. Finally, volume signifies how popular and frequently searched keywords are. This means that, although some keywords seem relevant and authoritative to the topic, they might not be the best ones to choose if there are not enough web pages against which those keywords can be matched [Leist 2022].

- Accuracy of search result

  When the relevancy of keywords centers around the idea of extracting important, meaningful keywords, the accuracy of search result is about how accurately the most appropriate candidate is found from the search query composed of those keywords. Even if the search strategy with the extracted keywords seem to be relevant, the accuracy of the actual search result might be really low. The ways in which the accuracy of the search result is measured and boosted are suggested by a number of companies and experts.

  1) Criteria for finding the ideal candidate

The accuracy of search result, to a large extent, relies on how the ideal candidate can be found from the search strategy. Hppy, the website which provides insights into HR and workspace programs, suggests that choosing the most appropriate keywords should lead to the most ideal candidate. The keywords should be "the exact wordings people use when searching for similar jobs", while being qualified with local keywords, such as city or region [Hppy 2021].

Also, searching the candidate in the right place for the right position can lead to a better search result as well. According to Mary Pratt, the recruiter should be mindful of where to post the job posting in order to find or narrow down candidates that they are looking for [Pratt 2020]. For instance, in order to find recent graduates, the recruiter should actively use social media, such as Linkedin. This means that the application should be able to add a filter to the search query depending on the job description it processes.

2) Boosting the accuracy

Since the goal of this project is to display the best accuracy for finding candidates, it is imperative in the literature review to explore the ways in which the accuracy can be boosted. Google explains that, for their search algorithm, the simple enumeration of keywords would not necessary lead to the best result [Google 2022]. The search query must have a strategy to pick the most clear, unambiguous words.

In addition, the recruiters have capitalized on the power of machine learning and artificial intelligence to find the best candidates. Javed and Brishti states in the academic paper that "AI has been used and implemented significantly in recruiting professionals in various companies from 2018 and becomes one of the latest trends in the recruitment industry since then" [7, Javed and Brishti 2020]. This means that for the project, AI and machine learning can be used to improve the accuracy of the search result.

III. Relevant examples

This section will explore some examples that are similar to this project. These examples are investigated because they may be able to offer some insights into the project. Also, by comparing examples, one should be able to find new aspects to add to this project.

1) 2DSearch

2DSearch is a tool for researchers and recruiters to solve some complex search issues. It can help recruiters or sourcing professionals to find the best candidate by combining x-ray searches with Boolean searches. According to their official website, it states that it can help for recruiters to "eliminate[s] many errors associated with traditional Boolean strings" while having an option to generate a query suggestion powered by artificial intelligence [2DSearch 2022]. 2DSearch also boasts that it provides query analysis and error detection features, while allowing the user to extend the search on various social media websites, such as GitHub and Linkedin.

One big advantage of 2DSearch is that it offers GUI, which means that the user does not have to type or insert anything. With a simple drag and drop of keywords and Boolean operators, the user is able to generate a query string and URL for a search engine.

For the final project, 2DSearch offers one meaningful inspiration: AI-powered search query builder. As discussed before, many recruiters have been embracing AI in their recruitment process. This shows that the final project should be able to capitalize on it as well.

However, a noticeable downside is that the filters it provides are not really granular, which means that the user cannot build a search query that is precisely tailored to his or her needs. For instance, if the user wants to generate a search query specifically for software developer, there are a number of specific things that need to be considered, such as remote job and tech stacks. These things are not really necessary for a lot of other jobs, so it would be great if there are some specific filters for different types of vocations.

2) Recrutee

Recrutee is one of the most popular HR software to find a candidate and track the application process. It strives for optimizing and streamlining the recruitment process by offering user-friendly tools and encouraging collaboration with other team members. It claims that its software "makes candidate sourcing 64% more efficient" [2022, Recruitee]. One of the most noticeable features is that it offers a vast number of extensions and support for popular platforms and social media, such as Linkedin. Also, it offers some useful features, such as tagging candidates and template.

One clear weakness of Recruitee is that it does not seem to use the state-of-the-art technology to fine-tune the quality of candidate search. Although it provides effective tools to increase the footprint of the recruiter on different social network platforms, it does not clearly explain how they can improve the candidate finding and how they came to the conclusion that it can make candidate sourcing more efficient.

3) Eightfold.ai

Eightfold.ai is a powerful tool when it comes to talent acquisition. It is widely used by global companies, such as Booking.com. The main strength of its software comes from its state-of-the-art technology. It clearly states that it is "AI-powered" and helps "hir[ing] for potential with Deep Learning AI" [2022, Eightfold.ai]. This proprietary software offers some great insight for this project, as it claims to boost "the number of highly qualified and diverse applicants" [2022, Eightfold.ai]. Unlike other proprietary softwares that have been investigated for this literature review, this understands that diversity could be an important factor. In the final project, there will be a feature where the recruiter could optimize some parameters to increase the impact on diversity.

Despite the powerful features that it offers, Eightfold.ai seems to lack a feature which allows the recruiter's presence in social media. Since it is often used by big companies with a lot of applicants, it might lack this aspect. In the age of social media and automated hiring, it would be wise to have a strong feature to extend candidate search on a wide range of social media.

4) A systematic approach to searching: an efficient and complete method to develop literature searches by Bramer et al.

This academic paper, written by Bramer et al, introduces an interesting model which can help researchers to search literatures. Although this model is not directly related to recruitment process, it provides some important insights into the ways in which a search on complex documents can be prepared and executed. Since the Internet is basically a

collection of scattered documents and the search engine is what helps the user to find a document on the Internet, the core idea of this research paper mirrors the objectives of the final project.

Bramer et al. acknowledges that literature is a very long and complex document, which merits an efficient search system [Bramer et al. 2018]. This means that the traditional Boolean search-based search strategies cannot be applied. Hence, Bramer et al. develop the following search strategy to search a complex document.

1. Determine a clear and focused question
2. Describe the articles that can answer the question
3. Decide which key concepts address the different elements of the question
4. Decide which elements should be used for the best results
5. Choose an appropriate database and interface to start with
6. Document the search process in a text document
7. Identify appropriate index terms in the thesaurus of the first database
8. Identify synonyms in the thesaurus
9. Add variations in search terms
10. Use database-appropriate syntax, with parentheses, Boolean operators, and field codes
11. Optimize the search
12. Evaluate the initial results
13. Check for errors
14. Translate to other databases
15. Test and reiterate

This example indicates that searching complex documents can be done if there is a process thoroughly thought out. This provides some inspiration on how to write an algorithm to build a search query string. However, some pruning would be necessary, as job descriptions are not as complex as literatures.

IV. Technical Details

For this project, several technologies will be utilized to improve or optimize the result of the application.

1) Python NLTK(Natural Language Toolkit)

NLTK is a library available in Python that allows the user to process human language. According to its official website, it “provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum” [NLPK 2022]. Normally, when a text is to be processed, it is tokenized and either stemmed or lemmatized. Stemming and lemmatization are converting the words to the most basic, meaningful form. This library provides all the pre-built functions and classes that can execute the aforementioned process easily.

2) Scikit-learn library for machine learning

Chamberlain and Russell-Rose argue that “[r]ecruitment professionals spend approximately 27% of their time actively searching for candidates and need to rapidly evaluate candidate

suitability", which encourages them to use "machine learning is used to select the best-suited individual to perform a particular task" [Chamberlain and Russell-Rose 2016]. As discussed above, machine learning technique is actively being used in the recruitment process. Hence, the project should explore this technique, as it will help to understand and classify the job description to a particular vocation and boost the accuracy of the result. There are several machine learning models that can be used. The most common model is Naive Bayes or Support Vector Machines (SVM). Scikit-learn library has many pre-built functions and classes, which will be useful to implement this.

3. Design

Template

As discussed above, this project will use the template : *automated search strategy generation*. The below project design is to achieve the best way to generate a search strategy or query to find a candidate that fits the most for a given job.

Overall project structure

In this subsection, the way in which the project is designed will be explored. This can be explained in two different ways. Firstly, it is imperative to discuss how the project itself is designed and structured. Secondly, it is pivotal to understand how the project management was designed.

1. The project

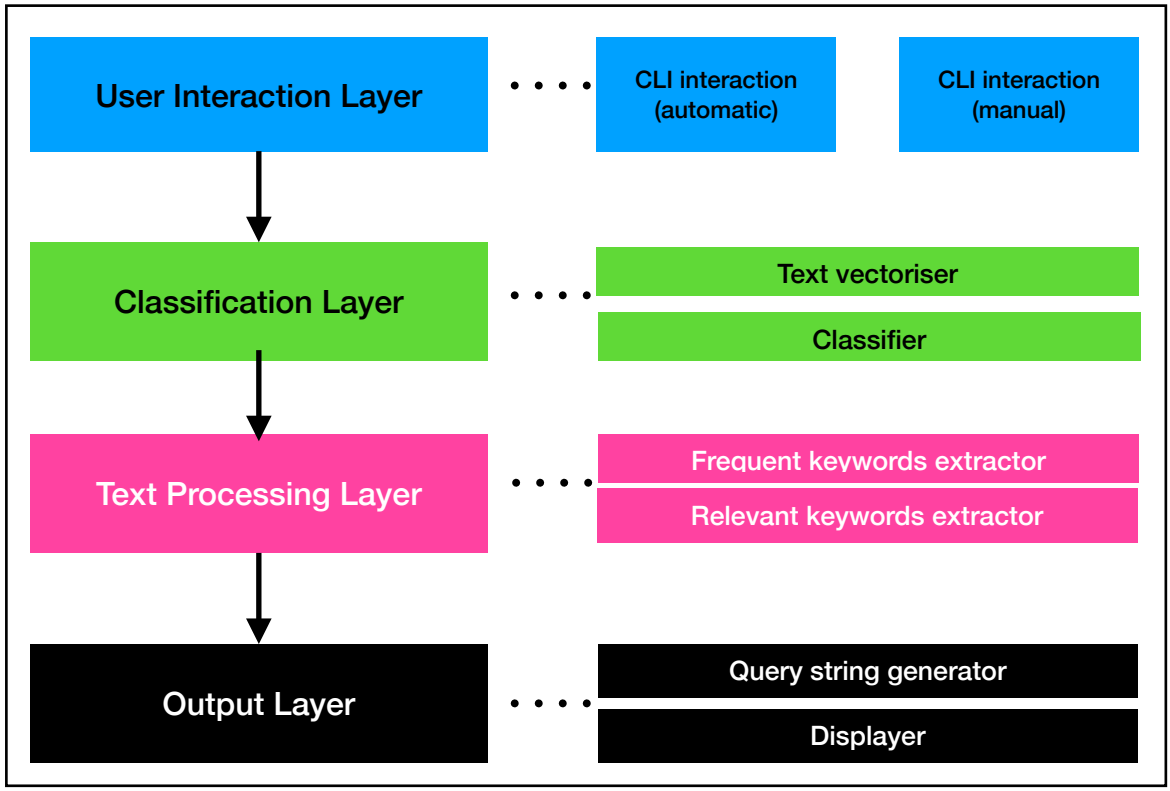


Diagram 3.1 : CV-Pixie diagram

The diagram 3.1 shows the design of the application, CV-Pixie. What the application is supposed to do can be summarized as follows: the command-line application, written in Python, will receive a job description for any type of job. Then, the application parses the job description and processes the text. The processed text will be used to build a search query string or strategy that can be used on a search engine. With this, a user should be able to see a search result which can lead him or her to a place where the ideal candidate can be found. There are several layers which allow the application to actualize the goal.

- User Interaction Layer

This is the layer where the user can interact with the application via command or terminal. The user will have two options to interact with the application: automatic and manual.

If the user selects manual, then the application will ask several questions and receives inputs from the user. If the user decides to use the manual configuration, then it will bypass classification layer, as the user has to provide the job description classification as well. If the automatic option is selected, then the application will choose the most appropriate configuration for the user.

- Classification Layer

  At this level, the provided job description gets processed and classified. In order to achieve this, there are several things that need to be done. The text needs to be parsed. Parsing here means that the whole text needs to be broken down into words that have a meaning. The text will be broken down into sentences, and those sentences will be separated into words. This whole process is called “tokenization”, and the final words are called “tokens”. The text processing does not stop here, as these tokens will be processed further. From these tokens, stopwords will be removed, and the left tokens are lemmatized.

  After that, these tokens are used to classify the job description. Depending on a job, the search strategy needs to be built differently. For instance, what a recruiter is looking for in a developer could be very much different from a customer support role. There is a pre-trained machine learning model in the application which classifies the job description automatically. This means that we can feed a job description to the model and expects the classification, which makes the whole process a lot easier and more automated for the user.

- Text Processing Layer

  At this layer, several keywords extractors are available to help generating the query string. Once the job description is classified, the necessary configuration will be automatically done at this layer. Here, the configuration means that, depending on the job description type, a different function will be executed.

  One of the keyword extractors finds the most repeated keywords. This extractor is created on the ground that the job description will repeat a number of words that the company wants to emphasize. The other extractor is to pull some relevant keywords out of the job description. For instance, if we are looking for a candidate in Science and Engineering sector, it will extract some keywords that are related to hard and soft skills, such as Python and Computer Science.

- Output Layer

  With the extracted keywords, a search query will be built. The structure of search query will be meaningful tokens enumerated with a boolean operator. Once the query is concatenated, it will be displayed on the command or terminal.

2. Plan

The below chart and table explain how the project was designed to be organized and implemented. The number in Diagram 3.2 corresponds to the id on Diagram 3.3.

Number	Task	Week
1	Choosing the project template Research relevant projects and academic paper	1-4
2	Research tech stack and its limit	5-6
3	Start building a prototype	7
4	Write a preliminary report	8-10
5	Explore machine learning model for text classification and build a working version	11
6	Explore NLP techniques and check how to build a meaningful model	12
7	Update the application with the research result	13
8	Research on SEO and how to build an effective search query	14
9	Create a search query builder	15
10	Research on machine learning model for building a search query builder	16
11	Finish the final version of the application	17
12	Write the final report	18-20

Diagram 3.2 : Task and week

Gantt chart

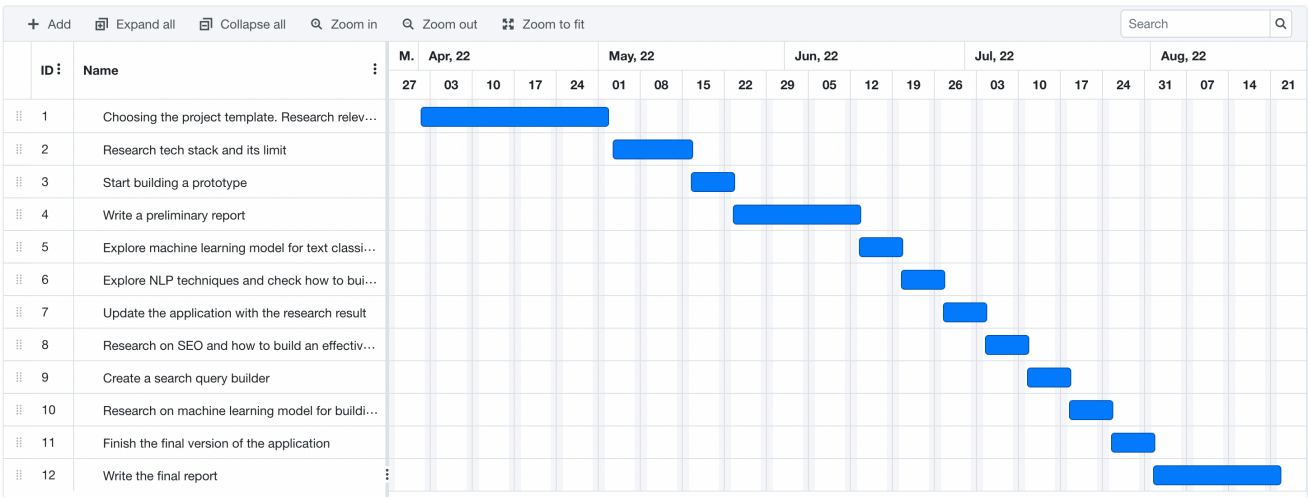


Diagram 3.3 : Gantt chart to visual the project organization

4. Implementation

The project is divided and implemented to uphold the separation of concern. This is well-reflected on the structure of repository as well. By going through this structure, some of the major algorithms and techniques, additional explanation, and some visual representations will be given. The structure is as follows:

- Data

  This folder contains job descriptions that the user can use to test. All of the job descriptions have to be in txt format.

- Model

  This folder has all the codes that create and train the machine learning model for job description classification. One of the most important techniques that are used in this project is to build a machine learning model to classify a job description. Here, the classification means that, depending on the content of the job description, the model tells what kind of job the recruiter is looking. The reason why this was implemented is to make the generated search query more sophisticated based on the job type. For instance, if a job description is classified as "Education", then it is most likely that the job description is to find someone who is supposed to teach or help someone.

  The final project comes with the pre-trained model, as the model was trained and exported by pickle. This way, the user does not have to train the model every time the application is run. To train the model, approximately 18000 job descriptions were used. These dataset was found in Kaggle [Bansal 2019]. Hence, training the model will take some time.

  The machine learning model is written in Python. It actively uses the available library, 'sklearn' and 'nltk' to process the data and build the model. However, since the application deals with natural language, the model has to be go through preprocessing. Firstly, the data will be cleaned. For instance, unnecessary characters, such as new line ('\n') and special characters, will be removed. After that, all the English stopwords will be removed. NLTK library offers a function which return all the stop words. Thanks to this, the following code can be run to remove all the stop words. Diagram 4.1 shows the code.

```
stop_words = set(stopwords.words("english"))
df['text'] = df['text'].apply(lambda x:' '.join([word for word in x.split() if word not in (stop_words)]))
```

Diagram 4.1 : Stopwords removal

  Once it is done, the pre-processed texts will be vectorized by using TfidfVectorizer. This is also available in sklearn. This vectorize uses TF-IDF, which stands for Term Frequency Inverse Document Frequency. In simple terms, TF-IDF converts text into a numerical representation, which can be used to fit a machine learning algorithm. As the application deals with natural language, this process is mandatory.

  After the vectorization, the model will be trained. In order to find the best performing model, several machine learning algorithms were used: K-nearest, Naive Bayes, and Random Forest. Before fitting the data into an algorithm, it was split into train and test dataset. The ratio was set to 0.2, so 80% of the pre-processed data will be used for training.

The training dataset will be fit to a chosen machine learning algorithm, and a prediction will be made with the test data. After that, a number of statistics will be given, such as accuracy. Based on this statistic, Random Forest Classifier was chosen to train the model. The Diagram 4.2 shows the accuracy of three different algorithms:

Name	Accuracy
K-nearest	0.58
Random Forest	0.78
Naive Bayes	0.61

Diagram 4.2 : Accuracy of algorithms

This justifies why Random Forest was chosen for the model.

- Util

This folder contains the codes that are responsible for the main features. It has codes that parse the job description and generate a search query. Building a search query works in the following way. Firstly, it makes a job type prediction with the generated machine learning model. This means that the application parses and vectorizes a job description. With this, the model makes a prediction and return a number. There is a mapping dictionary which converts it back to the job type. It looks like this:

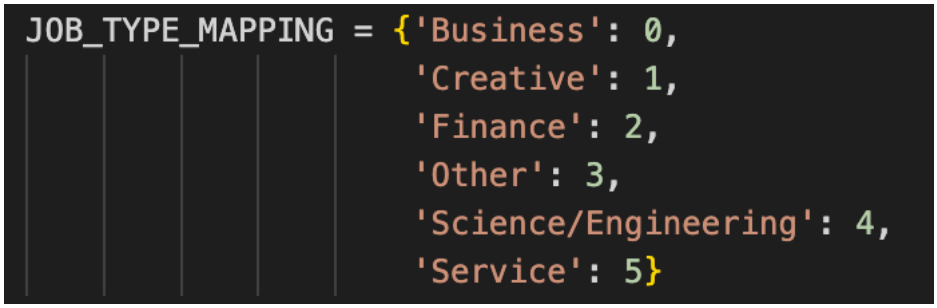


Diagram 4.3 : Job Type Mapping

Based on this, an appropriate function is called to generate the most appropriate search query. For instance, if the job description is classified as “Science/Engineering”, inside “generate_search_query” method, “__generate_search_query_for_science” is called. This method is optimized to generate a better search query for any job that is in science and technology sector. The search query is composed of keywords extracted from the given description, who are concatenated with boolean operators. The application has a simple, but powerful keyword selection mechanism. The main idea is to extract the most repeated keywords inside the job description. Certain keywords are often repeated inside a job description to emphasize some attributes that the company is looking for. If a search query is generated with the most repeated keywords, then the search result is not so good because these keywords are not necessarily indicative of the job itself. In order to improve the performance, the number of measures were taken. Firstly, depending on the configuration that the user set or the job group, certain keywords get more weight and will be extracted.

These extracted keywords will be used to generate a search query. This way, the search result is more polished and accurate.

- Root

This is not a folder, but at the root of the repository, there is main.py. In order to run the application, main.py file has to be executed. Inside the file, there is a mechanism for command-line operation. The command line operation on this application is interactive. The Diagram 4.4 shows how the configuration is done depending on the user input.

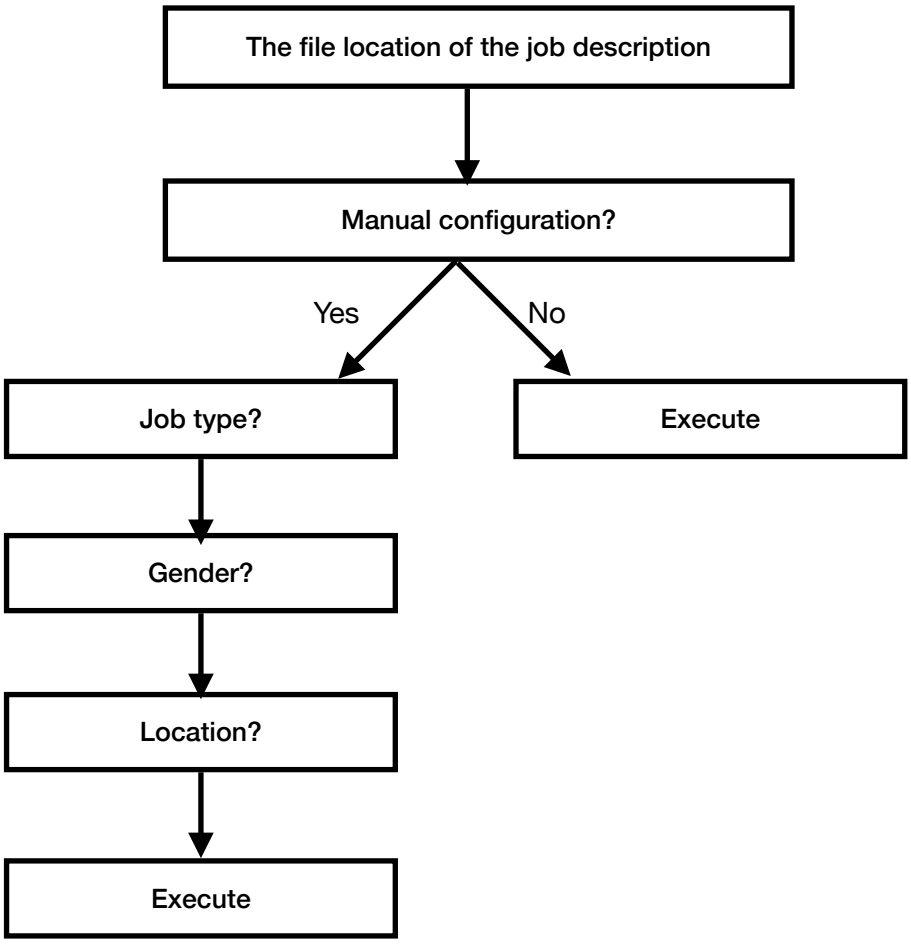


Diagram 4.4 : Configuration Process

By simply following the instruction on the terminal, the user should be able to get the desirable result.

5. Evaluation

Although it is clear what the project aims to achieve, it could be rather difficult to evaluate its performance with numerical values. This is because the performance of the search query string is abstract. For instance, determining the query accuracy involves a lot of potential metrics, such as how many social media platforms is queried, if the correct job was mentioned, and etc. However, throughout this section, it will be attempted to break down metrics as much as possible to come up with a feasible, effective way to evaluate the project.

1) Plan

The final project will be evaluated based on the domain and target user studies, the classification result of the machine learning model, and the quality of the search query. The classification result will be measured by feeding 20 different job descriptions and see if the results are highly accurate. For the quality of the search query, it will be measured to see how many relevant job-seeking or social platforms are shown as a search result when generating 20 search queries from different job descriptions.

2) Domain and User

The project domain is within the area of HR and recruitment. This project, CV-Pixie, is meant to help to alleviate stress and reduce monotonous tasks that the user has to do when finding a candidate for a job. This can be achieved by generating a search query which will provide the search result where the user can find those candidates. From here, the project domain can be defined as follows:

- The project should be able to receive a well-written job description and extract the meaningful data out of it.
- The project should concern HR and recruitment process.
- The project will lead the user to the best search result to find the most ideal candidate, but it does not evaluate individual candidates' suitability.
- The project is a command-line application (CLI), which means there is no GUI and the user must have some understanding on how to use CLI.
- The project should use the state-of-the-art technology to improve the quality of search result.

Based on the domain and the main goal of the project, it is quite clear that the target user is recruiters or any human resources workers. The recruitment process can be expensive and risky; companies would like to hire the best candidate with little mistakes. However, this is neither easy nor fast process. As Saved and Brishti argue that the more and more recruiters see that the recruitment process can be streamlined and improved by applying the state-of-the-art technologies, such as machine learning, this project is deeply imbued with the idea that the ideal candidate finding could be positively affected by computer science and information technology [Javed and Brishti 2020]. Based on this domain and user studies, this project makes a positive impact on tackling the current issues and streamlining the whole process.

3) Evaluation

- The classification result of the machine learning model

The performance of the machine learning model depends heavily on a number of factors. Firstly, the right machine learning algorithm has to be chosen in order to maximize the accuracy of the job description classification. The accuracy and the reasoning behind choosing Random Forest Algorithm have been clearly explained in Section 4 Implementation.

Secondly, the machine learning model has to be trained with the unbiased data that guarantees the correct classification of the job description. There were several obstacles while training the model. Initially, there were 37 label classes, which means that the training set has classified the job description into 37 categories. This is not a problem by itself, but if one or two classes are over-represented in the dataset, then it could lead to a bias. For instance, the initial dataset contained over 3000 classified job descriptions for "consulting". When the model is trained with this set without any pre-processing, the classification result is very biased.

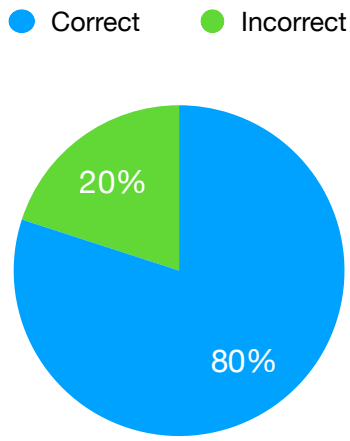


Diagram 5.1 : Pie Chart to show the accuracy

Hence, before training the model, a limit was set to 500, so any label class that has either more than or less than 500 records will be preprocessed. If a label class that has more than 500 records was trimmed to contain only 500 rows. On the other hand, if a label class has less than 500 records, then it is either dropped or left. The idea is that the model does not develop a bias towards a certain class, as job description can repeat same words over and over again. After that, the classification result was significantly improved:

After this preprocessing, 20 different job descriptions were fed into the model to measure the accuracy. The Diagram 5.1 shows the result. 80% accuracy rate was observed, which is a reasonable accuracy rate for a machine learning model.

- The quality of the search query

Once the machine learning model classifies the job description, the application will choose the function that is going to generate a search query that is the most optimal for that job. Instead of coming up with functions for every job type, the application has grouped similar jobs together. These are named "job group". So, the search query will be generated based on these "job group". For instance, the job group "Science" will contain jobs like engineer and developer.

The performance of the search query depends on the selected keywords and how these keywords are woven by operators. The Diagram 5.4 shows the final result which tracks how many relevant job sites are shown on the first page of the search result. The reasoning is that the search query should take the user to the relevant places where the candidate can be found. On Google, one page usually contains 8 - 10 websites.

The result shows that the overall performance of the search queries is good. There are some jobs, such as Barista, that do not display a good result.

Job Name	Number of job sites
Backend Developer	6
Customer Support Assistant	6
Legel Assistant	4
Researcher	7
Dermatologist	6
Pharmacist	5
Account Manager	5
Frontend Developer	7
Java Developer	7
Paralegal	5
Lecturer	6
Barista	3
Copyrights Lawyer	4
Nurse	6
Sales Manager	5
Marketing Director	8
Social Media Expert	6
Production Manager	7
Product Manager	8
Supply Chain Expert	8

Diagram 5.4 : Number of job sites

4) Weakness and potential extension

The weakness of the project comes from the fact that the trained model does not have an incredibly high number of training data. The data is from Kaggle, and it would be highly beneficial if we are able to have more qualitative data to train the model, which will boost the accuracy and support more granular classification.

Therefore, the potential extension of this project is to have a better trained model with more diverse dataset to cover some unusual jobs. This will be broaden the usability of the application, which leads to the wider range of potential users. Also, this will automated the whole recruitment process much better, as the user does not have to do a lot of manual configurations.

6. Conclusion

This project is based off the provided template “automated search strategy generation”. The main goal of the project is to generate a search query string for the recruiter which will lead to platforms where the most ideal candidate can be found. The only requirement is a job description. The project is designed to provide both automatic and manual configuration to generate a search query string. If the automatic option is selected, then the user simply has to provide the job description in txt format. Then, the pre-trained machine learning model will determine the closest job type and build a search query. If the manual option is selected, then the user has to provide all the answers to the questions that the application asks, such as gender and seniority.

For the aforementioned machine learning model, Random Forest is selected after comparing several different algorithms. In order to use Random Forest, the training set has to be processed and vectorized. The preprocessing actively used Natural Language Processing. Thanks to Python’s nltk library, stopwords can be easily removed, and each tokenized words can be lemmatized. Once all the words are processed, they will be vectorized with TF-IDF. With this vectorized text, the model is trained and exported, so the model does not have to be trained every time the application runs.

Once either the machine learning model determines or the user provides the job type, the right search query will be generated. The mechanism of building a search query consists of two main ideas. Firstly, the most repeated keywords will be extracted. Secondly, certain keywords will be weighted based on the job type. For instance, if the job type is for science and engineering, then the application will extract a number of keywords from a given job description that are highly representative of or relevant to the job, such as “SQL” and “Biology”. These extracted keywords will be combined by using the binary operators, such as AND or OR. The result will be prompted on the terminal or command prompt. The user can copy the result and paste on a search engine.

Despite some weaknesses, the project is evaluated to be promising and successful, as it delivers the functionalities that it promises and those functionalities are proven to be useful for the potential users.

Bibliography

Eightfold.ai. 2022. *Talent acquisition*. https://eightfold.ai/products/talent-acquisition/

Gupta, Er. Tanya. 2017. *Keyword Extraction: A Review.* International Journal of Engineering Applied Sciences and Technology. 2, 4, 215-220.

Leist, Rachel. 2022. *How to Do Keyword Research for SEO: A Beginner's Guide*. https://blog.hubspot.com/marketing/how-to-do-keyword-research-ht

Hppy. 2021. *How to Find Candidates Faster Using Basic SEO Skills*. https://gethppy.com/talent-management/how-to-find-candidates-faster-using-basic-seo-skills

Pratt, Mary K. 2020. *Social media recruiting strategies that work*. https://www.techtarget.com/searchhrsoftware/feature/20-social-media-recruiting-strategies-that-work

Google. 2022. *How Search algorithms work*. https://www.google.com/search/howsearchworks/algorithms/

Javed, Ayesha and Brishti, Juthika Kabir. 2020. *The viability of AI-based recruitment process*. Umeå University

2DSearch. 2022. *Recruiters and sourcing professionals*. https://www.2dsearch.com/recruiters

Bramer, Wichor M and et al. 2018. *A systematic approach to searching: an efficient and complete method to develop literature searches*. J Med Library Association. 106, 4, 531-541.

NLTK. 2022. *Documentation*. https://www.nltk.org/

Recruitee. 2022. *Recruitee*. https://go.recruitee.com/capterra

Chamberlain, Jon and Russell-Rose, Tony. 2016. *Searching for talent: The information retrieval challenges of recruitment professionals*. Business Information Reviews. 33, 1, 1-26.

Bansal, Shivam. 2019. *Real / Fake Job Posting Prediction*. https://www.kaggle.com/datasets/shivamb/real-or-fake-fake-jobposting-prediction