Accenture Content Analytics Group

Search and match for MyScheduling - model overview

Contents

[Search & Match for MyScheduling 2](#_Toc21601064)

[Overview 2](#_Toc21601065)

[What is Matching? 2](#_Toc21601066)

[Matching Algorithm 3](#_Toc21601067)

[Vectors, Multipliers & Filters 3](#_Toc21601068)

[Cosine Similarity 3](#_Toc21601069)

[Algorithm Formula 4](#_Toc21601070)

[Vector Descriptions 4](#_Toc21601071)

[Skills Vector 4](#_Toc21601072)

[Key Terms Vector 5](#_Toc21601073)

[Topics Vector 6](#_Toc21601074)

[Multiplier Descriptions 6](#_Toc21601075)

[Region Multiplier 6](#_Toc21601076)

[Location Multiplier 6](#_Toc21601077)

[Filter Descriptions 7](#_Toc21601078)

[Career Level Filter 7](#_Toc21601079)

[Geographical Unit Filter 7](#_Toc21601080)

[Features Extraction 7](#_Toc21601081)

[Statistical Analysis 7](#_Toc21601082)

[Structured Data 9](#_Toc21601083)

[Training 10](#_Toc21601084)

[Training steps 10](#_Toc21601085)

[Re-training Frequency 11](#_Toc21601086)

[Explainability 11](#_Toc21601087)

[Bias Detection 12](#_Toc21601088)

[Tuning the Matching Algorithm 12](#_Toc21601089)

[Manual tuning 12](#_Toc21601090)

[Automated Learning 13](#_Toc21601091)

[Results/Output Documentation 14](#_Toc21601092)

[User Interface 14](#_Toc21601093)

[Application Programming Interface - API 16](#_Toc21601094)

[Operations 17](#_Toc21601095)

[Bulk Processing 17](#_Toc21601096)

[Incremental Processing 17](#_Toc21601097)

[Automated Learning 17](#_Toc21601098)

# Search & Match for MyScheduling

## Overview

This document describes the Search & Match system which will be tested for finding Accenture employees to fill open Accenture roles. It describes:

* What is meant by Matching and how this differs from traditional search.
* How the Matching algorithm works.
* The information which is used as inputs to the Matching algorithm.
* How the Matching algorithm can be modified to improve its accuracy.

## What is Matching?

Before explaining what Matching is, let’s start by defining the traditional process for searching for a person to fill a role or for a role for a person to perform. We will then explain how Matching differs from this process.

To search for a person to fill a role, the user must:

1. Read and understand the role description.
2. Consider how best to construct a search query to represent the requirements for this role.
3. Construct a query. Complex queries will tend to return better results and may include:
   1. Key words to describe the role requirements.
   2. Phrases to describe the role requirements – these are sequences of words which must appear as a phrase because they represent a conceptual meaning which is more than just the component words. For example, when looking for a “project manager” it is important to find instances of that phrase rather than just people who mention the words “project” and “manager” in their CVs with no relationship between them.
   3. Boolean operators such as AND and OR to define the required relationships between query terms.
   4. Synonyms to reflect that different people may use different terms to describe their skills, capabilities and experience.
   5. Use of boosts to reflect that some terms are more important than others.
   6. Filters to describe hard requirements. For example, the person must live in a certain geographical region and have a career level in a given range.

Matching automates this process so that the user does not have to explicitly construct a query or be aware of complex search syntax. When performing a Match, the user only needs to provide the system with the source data against which the Match should be made. This is done by either selecting a role or person from a results list or by typing or pasting a free text description of the requirement. The Matching process then automatically constructs an underlying query by identifying the key features from the source data. The processes for identifying these features and performing the Match to find the best results is explained within this document.

Matching provides the following benefits:

* Time is saved because there is no need to construct a query.
* Users do not need to be experts in complex search syntax.
* Users can concentrate on their expertise of evaluating people for the roles which need to be filled.
* Matching tends to generate better results than a user with average search skills because it can consider more information and construct better queries.

## Matching Algorithm

### Vectors, Multipliers & Filters

The Matching algorithm compares the similarities of items (people or roles) using different characteristics, or dimensions. The dimensions used for MyScheduling are:

* Skills Vector
* Key Terms Vector
* Topics Vector
* Region Multiplier
* Location Multiplier
* Career Level Filter
* Geographical Unit Filter

These dimensions can be split into three groups: vectors, multipliers and filters.

* A **vector** is a list of values which could be words, phrases and/or codes. Each value has a corresponding weight which defines its importance in the vector. Vectors can be compared for similarity using a mathematical formula called **Cosine Similarity** to give a score between 0 and 100%.
* A **multiplier** is a value between 0 and 1 which can be multiplied with the match score to reduce the match score if certain conditions are not met. A multiplier, therefore, reduces relevance but does not exclude items from the results. For example, a multiplier can be used to reduce the match score if the person is not located in the required region.
* A **filter** restricts the match results using a metadata restriction. Unlike multipliers, filters will exclude items from the results. For example, a filter can be used to only return people whose career level is in the required range.

### Cosine Similarity

[Cosine Similarity](https://en.wikipedia.org/wiki/Cosine_similarity) measures the similarity between two vectors. The similarity score will be higher when:

* The vectors have more values in common with each other.
* The weights of the shared values are high.
* The vectors have fewer values which are not in common with each other.
* The weights of the non-shared values are low.

In other words, the similarity between two items will be high if they share a lot of characteristics and those characteristics are important to those items.

Matching in MyScheduling uses a variation on standard Cosine Similarity. In the standard formula, the similarity decreases if the two items have characteristics which are not shared. For role matching, although it is important that a person has the skills required by the role, a person should not be penalised for also having additional skills not required by the role. For that reason, values in the target vector have less impact if they are not present in the source vector. The Cosine Similarity formula is therefore changed from being a measure of absolute similarity between the role and person to a measure of how well the person meets the requirements of the role.

### Algorithm Formula

#### Dimension weight

The different vectors are not equal in their contribution to the final match score. Some are more important, or have higher weight, than others. The weights of the dimensions are initially set as global configuration parameters and can be tweaked based on the results as we go. From the outset we will use the following values:

* Skills Vector = 50%
* Key Terms Vector = 40%
* Topics Vector = 10%

#### The actual match score calculation

1. The **match filters** are applied to restrict the results set to only those which meet the criteria for the career level and geographical unit.
2. A **similarity score** between 0 and 100% is calculated for each of the three vector types using the modified Cosine Similarity formula.
3. A **vectors match score** is calculated as a weighted average of the vector similarity scores:

***Vectors match score = (Skills similarity score \* Skills weight) + (Key Terms similarity score \* Key Terms weight) +***

***(Topics similarity score \* Topics weight)***

For example, using the default dimension weights and similarity scores as follows: Skills=0.8, Key Terms=0.7, Topics=0.85. The vectors match score would be calculated as:

***Vectors match score = (0.8 \* 0.5) + (0.7 \* 0.4) + (0.85 \* 0.1)***

***= 0.765***

1. The multipliers are then applied to the vectors match score through a series of multiplications to calculate the **match score**. If we continue the example above by applying a Region Multiplier of 0.95 then the raw match score will be calculated as:

***Match score = 0.765 \* 0.95***

***= 0.72675***

This result will be displayed to the user as 73%.

1. Any result which has a match score lower than 40% will be removed from the results list.

## Vector Descriptions

### Skills Vector

The Skills Vector represents the skills required by a role or the skills possessed by a person. The source data for both roles and people contain skills values and proficiency levels (1-5) which are used to construct the Skills Vector.

Accenture has a skills ontology which defines and quantifies the relationships between skills. For example, some of the correlated skills for “python scripting” and their level of correlation include:

* scikit-learn (0.801532647)
* pandas python library (0.797203884)
* tensorflow (0.639030992)
* r programming (0.627191856)
* linear programming (0.3450905)
* natural language processing (nlp) (0.314821274)
* data normalization (0.1592012)

The Skills Vector takes all these relationships and their strengths into account. This enables matches to be made where the person’s skills do not exactly match the roles skill requirements. For example, the above skill correlations show that a person with “scikit-learn” as a skill would be a good match for a job requiring “python scripting”. Furthermore, the skills ontology also tells us that this person would be a good match for many other skills to which “scikit-learn” is correlated, including “data science” and “machine learning”.

The Skills Vector currently contributes the most to the overall match score because skills are a key factor for determining the fit of a person to a role and because the information which feeds into the Skills Vector (the skills, the proficiency levels and the correlations) all come from reliable sources.

### Key Terms Vector

The Key Terms Vector is composed of the most statistically important words and phrases in the item. The most statistically important terms will be those which appear many times in this item but relatively rarely in the entire population of roles and people profiles. The terms in the Key Terms Vector will provide a high-level overview of the item, like a “signature”.

The Key Terms Vector is generated using the following process:

1. All words are extracted from the content of the item.
2. Known job titles and skills are identified within the content.
3. Noun chunks (phrases) and standard entities (locations, company names, etc) are identified within the content. Phrases can add a lot of value to the Key Terms Vector because they are more specific than the individual words that make them up. For example, “senior”, “project” and “manager” are all quite common words in a CV and so, on their own, would have little importance, but when they appear as the phrase “senior project manager”, they describe a very specific job title.
4. Terms (words and phrases) are removed from the item’s set of terms if they are considered to be unimportant:
   1. Terms are removed if they are one of a set of certain parts of speech (e.g. prepositions, articles, adverbs).
   2. Terms are removed if they are so common (e.g. “and”, “the”) that they do not add any value to the vector. These are called **stop-words.**
5. The remaining terms are considered for inclusion in the Key Terms Vector. A **weight** is calculated for each unique term taking into account:
   1. The number of occurrences of the term in the item’s content. A higher frequency gives a higher vector weight; this is because words that appear many times in a single document are, in most cases, highly related to what that document is about.
   2. The number of occurrences of the term in the entire database of items. A higher frequency gives a lower vector weight; this is because a term that only appears in a small number of documents is an indication that those documents are the most relevant for that particular term. Therefore, terms that are very frequent among the entire set of documents do not help to distinguish a document as much as those terms that are infrequent.
   3. The combination of the above two values is referred to as [TF/IDF](https://en.wikipedia.org/wiki/Tf%E2%80%93idf) (Term Frequency/Inverse Document Frequency) and is used in Information Retrieval for determining term importance.
6. The vector is restricted to a maximum of 50 terms and to terms with a vector weight within a specified percentage threshold of the top weight in the vector. These restrictions prevent the use of very large vectors which include values which have minimal impact on the match score but which lead to long match response times.

### Topics Vector

The **Topics Vector** contains the key ‘**Topics**’ in the content. In its simplest form a topic can be thought of as a set of words that tend to be used together. For example, a topic could contain terms such as “health”, “safety”, “health and safety”, “occupational safety and health” and “risk assessment”.

The topics are identified using [Latent Semantic Indexing](https://en.wikipedia.org/wiki/Latent_semantic_analysis#Latent_semantic_indexing) which is a statistical technique which identifies patterns in the relationships between terms and concepts in the entire population of content. Each topic which gets generated has a very long list of terms each with a probability that they belong in that topic plus a list of terms which should not appear in content relating to that topic. The terms in a document are checked against these topics and probabilities to identify the topics and their weights for that item. This information is what is added to the Topic Vector.

Looking at the key terms in a topic, we can get a good idea about the subject of the topic; the topic in the example above is about “health and safety”. However, the process for generating the topics only knows the terms and their probabilities but does not understand the subject of the topic. For that reason, topics do not come with a user-friendly label and are not easy to express to the users.

Matching against the Topics Vector will compare the relevant topics in each item and will find items where at least some of those topics are identical. The key point is that this does not require that the items contain identical terms, the terms just need to be related to the same topic.

## Multiplier Descriptions

### Region Multiplier

The **Region Multiplier** will reduce the overall match score if the person is not in the same region as the role. The Region Multiplier is only used for North America.

If the person and role are located in the same geographical region, the Region Multiplier will be 1 and the match score will not get reduced. Otherwise, the Region Multiplier will be 0.9 and the match score will be reduced by 10% to reflect the imperfect match on the region. Out-of-Region people will still appear in the results list because it may still be worthwhile using them on the role if they are a good match in all other respects.

### Location Multiplier

The **Location Multiplier** will reduce the overall match score if the person is not in the ideal location for the role, or vice-versa. The Location Multiplier is disabled by default for North America.

If the person and role locations are within 100km of each other, the Location Multiplier will be 1 and the match score will not get reduced.

If the person and role locations are between 100km and 200km of each other, the Location Multiplier will be 0.9 and the match score will be reduced by 10% to reflect a less than ideal location.

If the person and role locations are more than 200km from each other, the Location Multiplier will be 0.8 and the match score will be reduced by 20% to reflect the poor location. It is important to note that the people who are located more than 200km away will still appear in the results list if their final match score is greater than 40%; this allows the best matching people to be considered even if they are not in the required location.

## Filter Descriptions

### Career Level Filter

All roles have a required career level associated with them and all people have a career level. When using a role or person as the source for the match, the career level for the source is extracted and converted into a filter to restrict the results to only those with acceptable career levels.

Example 1: If matching on a role which has a required career level range of 7-9, a filter will be applied to only return people who have a career level between 7 and 9 inclusive.

Example 2: If matching on a person whose career level is 7, a filter will be applied to only return roles which have a career level range requirement which includes career level 7.

### Geographical Unit Filter

All matches are restricted to the same geographical unit by applying a filter using the value of the geographical unit of the source item.

## Features Extraction

Relevant features where obtained using two sources: **statistical analysis** and **structured data**.

NOTE: A feature in this context is defined as measurable property of the objects that are being analysed. In this case, supply and demand data.

### Statistical Analysis

The provided data (both structured and unstructured) was analyzed to extract a set of features, which is composed of both individual tokens and phrases. **NOTE:** A token, in this context, refers to a series of one or more words.

The first step for feature extraction is to create a **token dictionary**, which contains meaningful features and their frequencies. The tokens which are considered for inclusion in the token dictionary are all individual words plus entities (organization, locations ,etc), noun phrases, skills and job titles. For example:

|  |  |  |
| --- | --- | --- |
| ID | Term/Phrase | Frequency |
| 81 | core trading & settlement ops | 74497 |
| 6 | functional test planning | 72041 |
| 538 | manager | 70847 |
| 54 | integration | 69749 |
| 195 | planning | 68772 |
| 18 | product | 68166 |
| 65 | project management body of knowledge (pmbok) | 68166 |

This dictionary was built by analyzing all tokens of both structured and unstructured data fields. More specifically:

Demand:

* Role Description
* Role Title
* Assigned Role

Supply:

* Job Experience
* Profile Summary
* Training
* Certifications
* Education
* Specialization/Aspirational Skills

#### How is the token dictionary built?

The following steps describe the process of building the token dictionary:

1. Load Documents:English documents (which were previously filtered as described above) are loaded for processing.
2. Pre-process text: This is the most extensive and important stage of the training phase, as it will refine tokens to keep valuable and meaningful ones. It consists of:
   1. Remove boilerplate text:Some documents contain some auto-filled sections which are discarded since they don’t add any value and are common across a good portion of the documents. These boilerplate texts were identified by analyzing common phrases found in the text. For example: “mySched auto-fill: Please Update”. If any occurrences of these texts are found in the analyzed documents, then they are removed.
   2. NLP Analysis:For this analysis, SpaCy has been used. This a powerful Python library used for natural language processing (NLP).
      1. Free text is split into smaller tokens and tagged with their part-of-speech (POS). The POS represents the semantic meaning of each word based on its context. The complete text is provided to SpaCy, which will split, analyze and tag content, word by word.
      2. Custom pipelines have been written and plugged-in to Spacy with the objective of grouping and identifying custom entities such as job titles and skills. This tagging is dictionary based. This means that the content is scanned looking for occurrences of values that are present in a unique list (dictionary).
   3. Token Extraction:Tokens and extracted entities are analyzed. This involves several steps:
      1. Unwanted parts-of-speech are discarded (such as conjunctions, numbers, punctuation marks, etc.) These don’t add any semantic value, therefore can be ignored.
      2. Tokens are lemmatized. Lemmatization entails removing inflectional endings to words and returning its base form, or lemma. It normalizes words to their common lemma, which facilities search and matching. For example:
         1. “am“, “are“, “is“ are all lemmatized to: “be“
         2. “car“, “cars“, “car’s“, “cars’“ are all lemmatized to: “car”.
         3. “teach”, “teaching” and “taught” are all lemmatized to “teach”.
      3. Phrases and tokens are cleaned up:
         1. Stop-words are removed from the beginning or end of phrases.
         2. Punctuation marks are removed.
         3. Leading or trailing whitespaces are removed.
      4. Individual tokens are filtered. The following are ignored:
         1. Small tokens (less than 2 characters). Although there is an exception list that contains terms like: “C”, “UI”, “UX”
         2. Numbers: e.g. 1234
         3. Dates: e.g. 05/08/2010
         4. Tokens containing a timestamp: e.g. 12:00am-shift’
         5. Tokens with more than 3 consecutive numbers: e.g. v3453
   4. Normalization: Terms and phrases are replaced with their standard synonym (if exists) based on Accenture’s technology aliases list.
3. Create Token Dictionaries: Once all tokens per document are filtered, the token dictionary is built, taking into account the following thresholds in order to keep only the most meaningful tokens:
   1. Minimum token frequency: Tokens which have a frequency lower than this value are excluded. The reasoning behind this is that tokens which appear very infrequently will not be useful for matching purposes. The default value for this threshold has been set to 20.
   2. Maximum proportion threshold: Tokens which appear in more than this % of the documents are excluded. This effectively rejects tokens which are too common (and not very meaningful).
   3. Maximum dictionary size: The dictionary will be restricted to a maximum of 100,000 tokens. If there are more than this number of tokens after the two thresholds above are applied, then the tokens with the lowest frequencies are removed until the maximum size is reached.

The token dictionary is then used to create two vectors: key terms vector and topics vectors. More on these later.

### Structured Data

A list of unique skills has been obtained from two sources:

* Supply/demand structured data that contains skill names and proficiency levels.
* Accenture’s skills ontology, which contains correlations between related skills.

Unique skill names have been collated from these two sources in order to create a feature list that is going to be used to create a skills vector.

|  |
| --- |
| Skills Dictionary |
| … |
| jacoco |
| japanese |
| jasmine |
| jaspersoft reports |
| java 2 micro edition (j2me) |
| java 2d and 3d |
| java api for json processing |
| … |

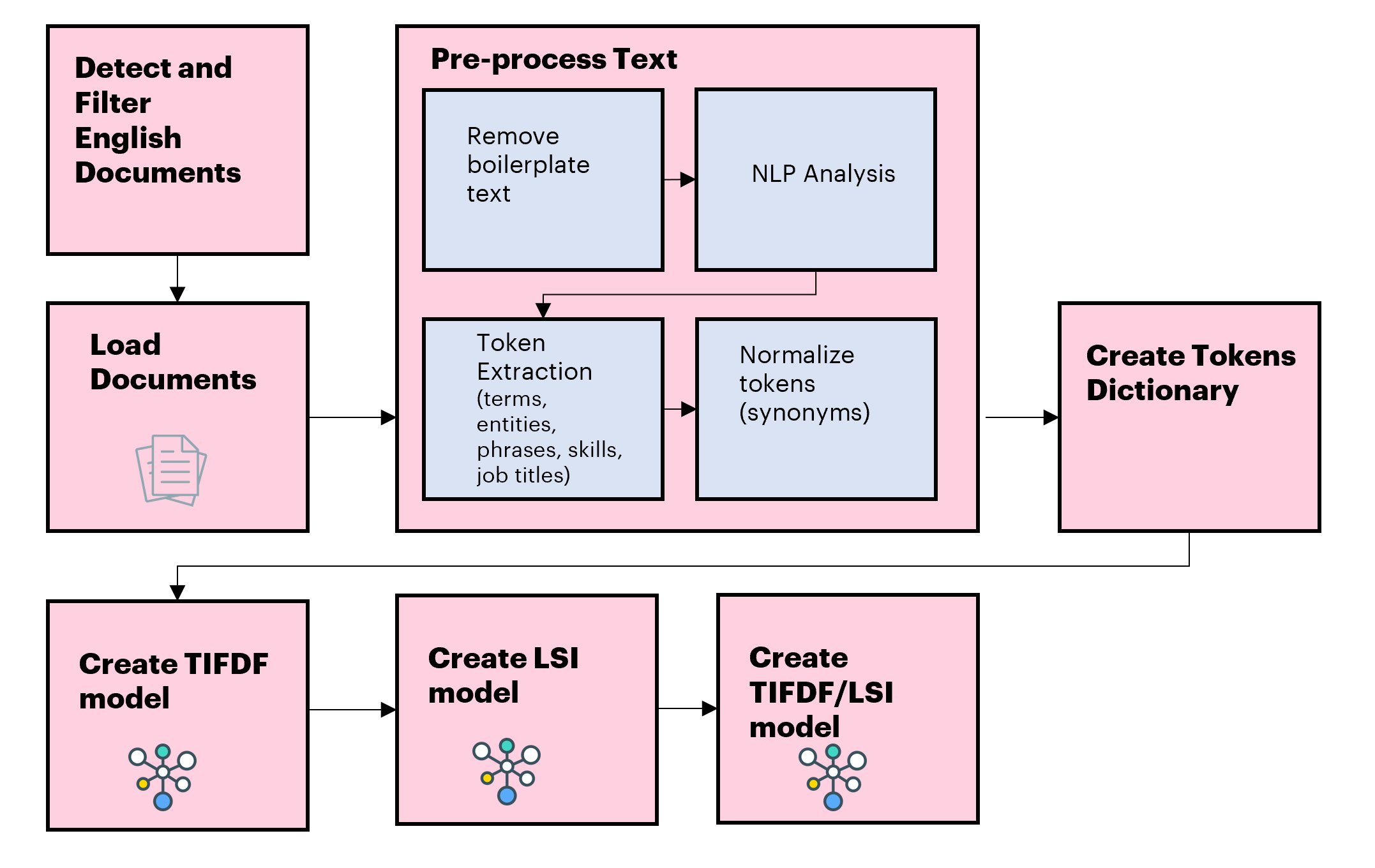
For example:

## Training

Statistical models (TFIDF and LSI) have been trained in order to generate key terms and topics vectors.

### Training steps

1. Build the token dictionary: This process is explained in the feature extraction section above.
2. Create TFIDF and LSI models: Using the token dictionary, the TFIDF and LSI models are built. For this, we use Gensim, a Python library for unsupervised topic modelling.
   1. The TFIDF model is used to generate the Key Terms vectors. TFIDF calculates the most relevant terms that describe a particular document. For this, it takes into account the number of occurrences of the term in the item’s content as well as the number of occurrences of the term in the entire database of items. Terms which have a high TFIDF score for a document are those which appear in that document more than they do for what could be expected for an average document. It is therefore these terms which will be useful for including in a Match comparison.
   2. The LSI (Latent Semantic Indexing) model is used to generate the Topics Vector. Latent Semantic Indexing is an unsupervised machine learning algorithm which identifies clusters of related terms. This enables matches to be made between a supply and demand even if they do not share any of the same words (as long as they contain the same topics).



Initially models will be trained for US. However, for other scenarios this could be tuned per geographic unit, DTE, etc. The most appropriate level to tune at must be determined, based on logical differences between regions, DTEs, etc. and based on volume and quality of training data. As we gather more data, it will become more appropriate to use lower level models with individual tuning.

Since the search and match services will be accessed via API, key hyperparameters could be tuned manually by user to experiment. The application could potentially learn from these interactions in order to provide better results.

### Re-training Frequency

On weekly basis, dictionaries and models will be rebuilt, including incremental data during that period. Also, on the background, all data will be re-indexed into the search engine to reflect the changes in the models. This re-index includes regenerating all vectors to reflect the new model changes.

Once finished, seamlessly new indices, models and dictionaries will replace the old versions without any down time.

As the models are regenerated weekly, new terms will automatically be added if they become sufficiently significant to be included.

## Explainability

Our experience of using Search & Match in recruitment has highlighted the importance of explainable results. Explainability is a key feature of our design, both for the structure of the algorithm and for understanding why individual results have been returned.

The application has ‘explain match’ features that provide a breakdown of the partial scores that contributed to the final one. Also, the most relevant terms that influenced each partial score are reported, including a relative percentage of their contribution.

The ‘explain match’ service will expose all necessary information that the UI requires to visually present it.

## Bias Detection

Based on the team’s previous experience with matching solutions, we were confident that the suggested models and algorithms wouldn’t introduce significant bias towards any particular demographic group. However, analysis and evaluation were performed in order to back this.

All vectors were semantically analyzed to extract part of speech for each word in a term or phrase. Regarding gender, masculine/feminine strong-coded words (taken from a [gender decoder study](http://gender-decoder.katmatfield.com/)) that introduce bias are more likely to be adjectives than any other part of speech. Therefore, our analysis was focused on them. After collecting and studying results, we concluded that:

* 0.372% of TFIDF vector terms are masculine-coded
* 0.224% of TFIDF vector terms are feminine-coded

Additionally, Accenture’s Data Science practice provided a list of terms and phrases that could unveil not only gender bias, but also religion, ethnicity, sexual orientation, etc. Vectors were analyzed against this data set using the same approach as explained above. We concluded that:

* Only 0.60% of the vectors’ single terms exist in the biased terms list. For example: functional, source, root, color, perspective, history, Germany, Africa.
* Only 0.03% of the vectors’ phrases exist in the biased terms list. For example: Quantitative research, United Kingdom, South Asia, cognitive science.

These results show that there is no strong bias in our application. However, for the future, we could implement bias reduction techniques:

1. Hard debias: Remove any possible biased terms from the vector. This option tackles the issue in a more aggressive manner as it would also remove words that could be meaningful under other contexts.
2. Soft debias: Reduce the weight of biased terms. Relevant terms would still be part of the vector but with diminished weights. Relevant terms under other contexts would be still available for matching. The reduction factor can be configurable and tuned using machine learning techniques.

## Tuning the Matching Algorithm

### Manual tuning

The following values can be manually configured:

* Dimension weights
  + Skills Vector weight (currently 50%)
  + Key Terms Vector weight (currently 40%)
  + Topics Vector weight (currently 10%)
* Multiplier parameters
  + Out-of-Region Multiplier (currently 0.9)
  + Inner location distance (currently 100km)
  + Multiplier for between inner and outer distance (currently 0.9)
  + Outer location distance (currently 200km)
  + Multiplier for between inner and outer distance (currently 0.9)
  + Multiplier for outside outer distance (currently 0.8)
* Minimum match score threshold (currently 40%)

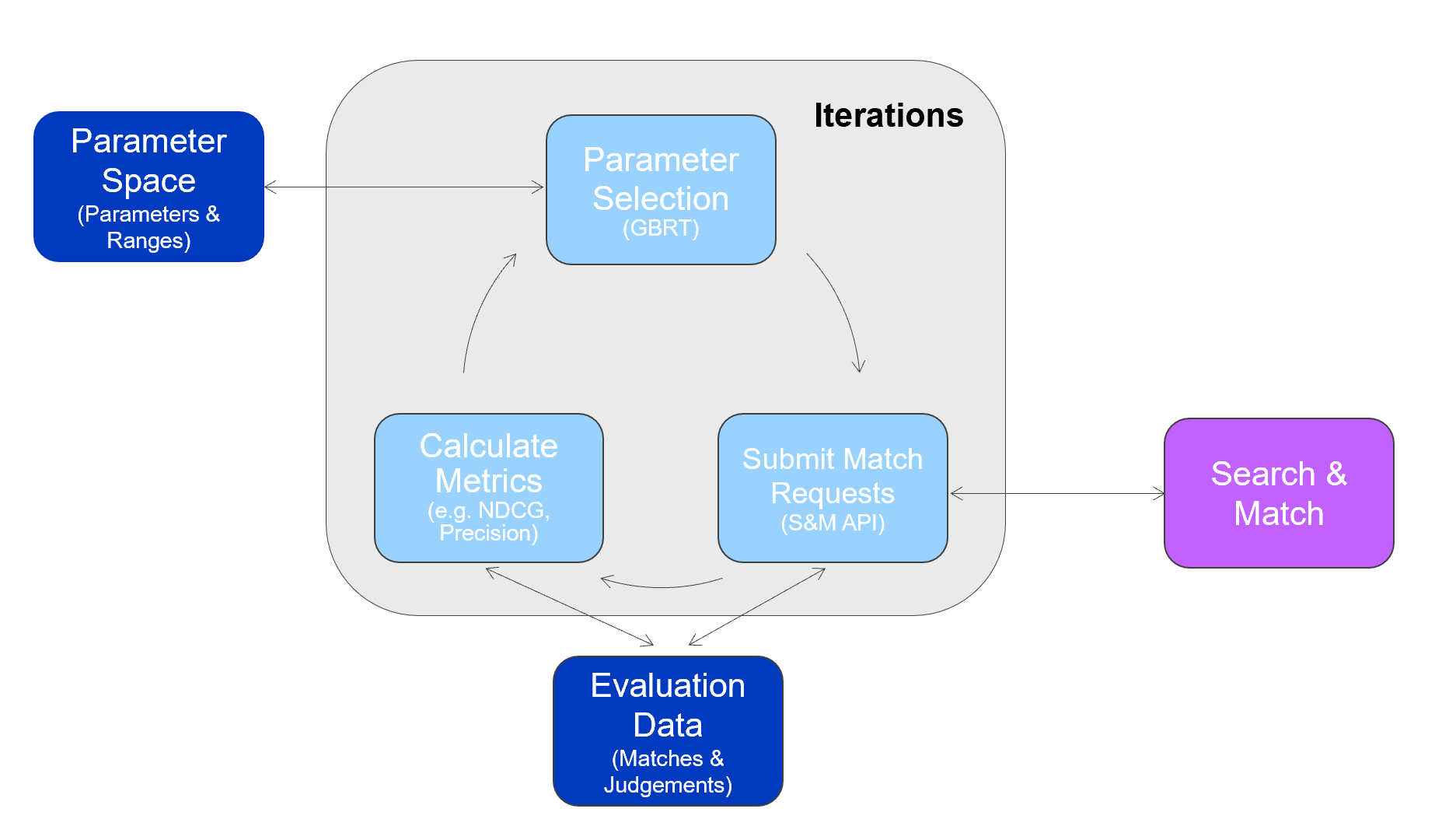
These values can be manually changed globally to modify the matching behaviour.

### Automated Learning

Once feedback/actual matches data is capture in sufficient quantity, we will be able to start using machine learning to determine the parameter values which will optimise the quality of the match results.

Gradient Boosted Regression Trees (GBRT), a machine learning algorithm, has been implemented to provide the necessary self-learning robustness to adjust parameters based on feedback gathered from actual usage by recruiters.

The tool will learn through the use of an optimization component which will optimize the algorithm's hyper-parameters. The optimization component will use data on "positive" and "negative" results for evaluation matches. It will iteratively and intelligently explore the defined parameter space for the hyperparameters to find the set of parameter values which optimizes the accuracy metric based on the positive and negative matches. In each iteration, the evaluation matches are executed against the Search & Match system and the results are checked against the positive and negative matches to calculate the accuracy metric. The optimization component uses machine learning to set the parameter values for each iteration. This enables the optimization component to find the optimal parameter values much faster than if it were using a conventional grid or random search. For example, for another project, the optimization algorithm was able to identify the optimal values within several hundred iterations for a parameter space which consisted of 10^27 combinations.



The optimization component directly requires positive and negative results for each match in an evaluation set. This data can be generated externally to the optimization component based on a range of different sources including thumbs-up/down, star rating, mouse clicks, page browsing data and ranks.

These are the parameters that will be adjusted/optimized:

* Vector weights: Impact of each vector in the final score. These values delimited from 0 to 100%. Current values are:
  + Skills Vector weight: 50%
  + Key Terms Vector weight: 40%
  + Topics Vector weight: 10%
* Proximity skills weight: Proximity skills weights are calculated by multiplying Accenture’s ontology proximity score, proficiency level of the original skill and a proximity weight factor. The value for the latter is the one than can be adjusted. This value ranges from 0 to 1. For the baseline report creation, we have experimented with three values: 0.75, 0.25 and 0.1.
* Unmatched terms: Factor to reduce the impact of unmatched terms. This is an integer value from 0 to 100. It is currently set to 10.
* TFIDF cutoff threshold: This value determines which terms to include for matching, based on their relationship with the top weight of the same vector. The purpose of this threshold is to ignore terms that add little or no value to the overall match. Currently set to 12%. It can vary from 0% to 100%.
* Location multiplier: The location multipliers penalizes results that are too far away from the source document. This is defined by radiuses and weights. If a candidate is within 100 km of the job location, then the score is kept unaltered. If a candidate is between 100 and 200 kms, then the score is multiplied by 0.9. Else, is multiplied by 0.8. These mentioned values are being used currently.
* Region multiplier: If a candidate is not within the same US region than a job, then its score is penalized by multiplying it by 0.9. This is the current value however it can be adjusted. NOTE: This multiplier applies only for US.

## Results/Output Documentation

There are 2 ways to look at the output from the Search and Match solution. One, through the interface it offers, and another one as the set of commands that will allow external tools to query the results it offers.

**NOTE:** The interface was developed for evaluation purposes at the pilot stage.

### User Interface

The Interface offers 2 types of outputs: results page and the entity (either a candidate or a position) results.

#### Results page

After a query is performed to find, for example, candidates that match a type of position, there results page displays the series of candidates that match the position.

The following are the possibilities queries that would create a results page:

1. Candidates for a position
2. Positions for a candidate
3. Candidates who are like another candidate (implemented but no in scope for phase 1)
4. Positions that are like another (implemented but no in scope for phase 1)

Therefore, the results list can be comprised of either positions or candidates, depending on the type of query entered.

The results will be filtered and then sorted by the following criteria:

1. Filters
   1. Any user-selected specific filters
   2. Career level (CL)
   3. Geographical Unit (GU)
   4. It is also planned that the LCR (loaded cost rate) is added as a filter (not implemented yet)
2. Sorts: the list will be sorted by candidates/positions in relevance order. The relevancy will be evaluated by the following features:
   1. The skills present. This will consider Accenture’s standard ontology of skills which include proximity of skills (e.g. “Spring Framework” implying the skill “Java programming language). The skills account for a total of 50% of the relevance to appear on the result list.
   2. Key terms. Terms appearing many times in summaries (profile/job description) such as “leadership” when looking for management-related positions. The key terms account for 40% of the relevance to appear on the results list.
   3. Topics. Groups or sentences of related terms that tend to appear together such as “risk management” when searching for health or security positions. This accounts for the remaining 10% of the relevance.
3. Demotion of results: in some cases, the results’ list could include potential candidates that are similar (but not necessarily an exact match) to the position that is being queried, however, the candidate could be living in the exact same area making the candidate a possible choice. Similarly, the perfect candidate could exist but live in a different city far away or even a different continent altogether. To consider these possibilities, the relevance is affected as follows:
   1. Any entities residing between 100-200 km away from the target receive a 10% penalty[[1]](#footnote-2).
   2. Any entities residing over 200 km away from the target receive a 20% penalty.
   3. In the case of the United States of America only, regions (e.g. “the Northwest”) are considered. Candidates leaving outside the target region receive an additional 10% relevance penalty.

The relevance of the position can be found as part of the information provided in the results list. Further drill-down into its relevance provides exact scores for the items detailed above.

#### Entity Details

Upon clicking in an entity in the results page, details of the entity will be shown. Details will depend on whether the entity is a position or a candidate. The details on each one can be adjusted/configured and will depend on the information that is received into the system as per the Dataset documentation (specifically, the Output – Elasticsearch mappings section).

### Application Programming Interface - API

The API is a programmatically-accessible set of commands that will enable myScheduling to access the information as needed by the system.

The implementation libraries are still being migrated, so this area will be completed once the migration is over.

## Operations

### Bulk Processing

On weekly basis, bulk data processing is required. This must be done in order to retrain the models to capture any new information that has been introduced in that time frame.

Re-training the models frequently guarantees that as new terminology appears (for example: technology names/tools), the model captures the relevance of this terms. As they become more prominent, the vectors created with these new models may include the new terms. If we don’t retrain periodically, our models would reflect stalled information.

Re-training the models also imply that all data must be reindexed into Elasticsearch. This, in order to recreate the vectors using the fresh models.

These two processes must be coordinated and synchronized in order to avoid downtime and ensure an adecuate behaviour of the solution while this is happening. For this, a blue/green deployment approach is proposed.

**NOTES:**

* A separate document has been provided with more details of the blue/green deployment approach.
* A separate document has been provided with more details in the data flow for both incremental and bulk processes.

### Incremental Processing

Every 6 hours, updated data within that timeframe (i.e. deltas) will be reprocessed and indexed to ES in order to make this data available for matching. The models won’t be retrained for incremental processing.

### Automated Learning

For future phases, the Search and Match solution will learn from user’s feedback using machine learning techniques. This is something that has been excluded from the phase 1, but it should be integrated into the solution. The following is required:

* Gather data from users’ interaction with the system (thumbs-up/thumbs-down feedback, actual placements, etc)
* Using the aforementioned machine learning techniques, predict optimum hyperparameters.
* Test these hyperparameters.
* Use them for the production system.

1. Penalties are calculated as a percentage of the relevance score and not a total percentage decrease. For example, if a position has a relevance of 80% but is 150km away, it receives a decrease of 8 point for a final 72% relevance. [↑](#footnote-ref-2)