Jobs to Skills

Documentation

Overview of Code Function

This application aims to extract skills from job descriptions. Given a job description, the systems will first recognize skills from the text and then find implicit skills which are not mentioned in the requirement. The output will consist of extracted skills, related skills, and trending skills.

The system includes two modules, called skill extractor and implicit skills extractor.

Skill Extractor

1. Overview

Text clustering method is used to determine the relevant skills in the job descriptions. A group of texts comprised of

- Job description texts mined from LinkedIn (4,907 titles)
- Skill description texts mined from Wikipedia (147 skills) in which the list of relevant texts were taken from Skills and Technology Skills list provided by the Occupation Information Network (O* Net)[1] (version 26.1).

The above texts were then fed into k-means clustering in order to identify skills that are highly similar to the job descriptions by being in the same clusters. Through this method, we would be able to identify the skills that are relevant to the job descriptions.

2. Skill Extractor Module

The skill extractor module are comprised of the following processes

a. **Data cleaning**

- Remove line breaks
- Remove any HTTPs
- Remove numbers
- Remove whitespaces
- Remove empty characters
- Lowercase all texts
- Lemmatize texts to root words

[1] Retrieved from https://www.onetcenter.org/dictionary/26.1/excel/

b. Data vectorisation

Two methods are used to vectorise the texts. They are

- Bidirectional Encoder Representations from Transformers (BERT) embedding
- Term frequency–inverse document frequency

c. K-means clustering training

For each of the method above, a k-means model was trained utilising

- k-means++ initialisation
- Number of clusters to be chosen from range 2 to 500 in incremental steps of 50. Both the elbow and silhouette methods were used to evaluate the suitable number of clusters to be used. We take note that since the job descriptions might share overlapping skills and the clusters might not be as clearly separated; hence, the elbow method is the main method to choose a suitable number of clusters.

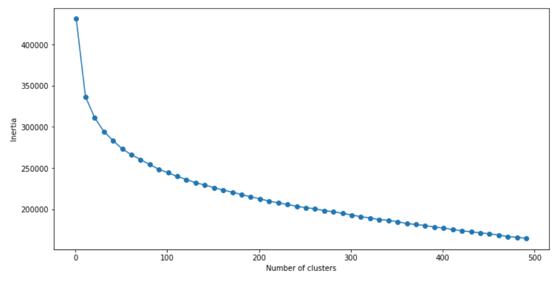

Figure 1: BERT-based Inertia vs Number of Clusters. A reasonable choice of cluster is 50.

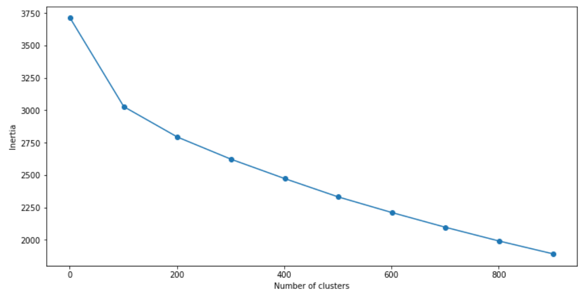

Figure 2: TFIDF-based Inertia vs Number of Clusters. A reasonable choice of cluster is 150.

d. Model saving and result analysis

With the chosen number of clusters, 2 k-means models were trained and saved. We proceeded to do a heuristic test of 30 job descriptions and observed relatively reasonable skills returned.

The following are the top 5 skills (skills that lie in the most populated clusters) extracted by BERT-based and TF-IDF model respectively

No	**BERT-based model** **(in a cluster of size 212)*	**TFIDF model** **(in a cluster of size 305)*
1	Programming	Reading Comprehension

2	Technology Design	Active Listening
3	Project management software	Writing
4	Configuration management software	Critical Thinking
5	Object or component oriented	Coordination

The result of BERT-based model is different from TFIDF model because

- TFIDF is based on mapping of exact words; hence, it tends to capture more common words such as reading, listening, writing in most job descriptions
- BERT is based on the semantic similarity of words so it could help k-means clustering model capture more sophisticated relevant skills to the job descriptions (such as drawing the similarity between a software name and the relevant technology skills)

e. Future improvement

Our training data is highly skewed towards IT and accountancy job scopes; hence, the models might not be able to perform well to cluster job descriptions outside of these fields.

Moreover, the methodology is limited such that there might be clusters without any skills assigned to it. The user might experience a null value returned when querying for the skills in the job descriptions

Finally, a more careful study of choice of clusters could be implemented.

Implicit Skill Extractor

1. Overview

Implicit skills are skills that are not mentioned in the job description. For example, a job description for a technical consultant may not mention specific technical skills which are helpful for this job. Determining implicit skills will help suggest related skills or recommend related jobs to users. This can be helpful for matching resumes and job's descriptions as well. Besides, implicit skills are needed in inference systems to infer useful skills for students in educational systems.

Topic model is chosen because of the nature of a job description. Nowadays, numerous jobs require cross functional skills which belong to different fields. A job itself can belong to multiple industrial sectors. For example, an engineer in the fintech industry should acquire both technical and financial skills. Topic models can satisfy this requirement because they represent the coverage of different topics in text documents. This is similar to field coverage in a job description.

2. Implicit Skill Extractor Module

This module depends on the skill extractor module to find implicit skills in a job description. Usually, those skills will be listed in similar job requirements which belong to the same industrial sector or researching fields.

Based on that, Latent Dirichlet Allocation ('LDA') model is used to transform and determine the topic coverage of job descriptions. Topics can be working areas or research domains such as engineering, software, or finance. The number of topics is chosen by grid-search algorithm to yield the most optimal parameters.

The top ̀most similar jobs of the query will be retrieved based on the distance of topic distribution vectors.

Implicit skills are then extracted from those jobs and added to the query.

The model is built using Scikit-Learn, a Python library. Firstly, job's descriptions are loaded from the dataset and then cleaned by removing punctuation. Those descriptions are then transformed to vectors of term frequencies. In this process, words are converted to lowercase and stop words are filtered based on the existing English stop words list in Scikit-Learn. After that, the vectorized documents are fed to a Scikit-Learn module to train the model.

The model is tuned using grid search, an algorithm used in Scikit Learn. It receives the data and a grid of parameters and then builds multiple models using all combinations of given parameters. The best one is then selected by comparing the score of the models. This score is calculated based on the log likelihood of LDA models. The grid of parameters is a dictionary containing the parameters' names and lists of values. Below is an example:

{ 'n_component': [5, 0, 15, 20],

'learning_rate': [0.5, 0.75] }

In this project, the number of topics, 'n_components' is the only parameter for tuning purposes.

After finding the best model, it will be used for transforming the given job description.

The given description will go through a similar process consisting of vectorization and transformation. After that, similar descriptions are retrieved based on Euclidean distance. The smaller the distance, the more similar the documents. The top five similar texts are then chosen and used for extracting implicit skills.

Skills extractor modules are called to get the skills from the texts. Those skills will be added to the list of skills found in the given job description.

Detailed Usage of the Code

1. Skill Extractor Module

In order to utilize the module, user would follow the following steps

Step 1: Set up the notebook. Make sure that the following files are available by uploading the files to Google Collaboratory at your local drive.

File Name	File Description
/model/TFIDF_JobClustering (2).joblib	Saved TF-IDF model
/model/BERT_JobClustering.joblib	Saved BERT based model
/data/BERT Skill Group.csv	Saved result generated by BERT
/data/TFIDF Skill Group.csv	Saved result generated by TFIDF
/model/TFIDF_Vectorizer.joblib	Saved TFIDF pre-fitted vectoriser
/data/usertest.txt	Testing file listing job descriptions

Package requirement

Package (Python 3.7)	**Version**
sentence-transformers	2.1.0
scikit-learn	1.0.1
spacy	2.2.4
re	2.2.1
joblib	1.1.0

For ease of usage, please run the notebook on Google Collaboratory.

Step 2: Use the usertest.txt file or any custom file containing job description separated by a new line

Step 3: Run all the cells in the notebook 03 Try K-Means Model.ipynb

Step 4: Two types of results will be returned. The result will be returned by both the model based on BERT embedding and TF-IDF vectors.

a. Skills that might be relevant to each job description;

b. Industries that the job description and the associated skills are related to.

2. Implicit Skill Extractor Module

Step 1: Set up.

The module can be run locally or on Google Colab.

To run this module on Google Colab, the following files need to be uploaded.

File Name	**File Description**
/model/best_model.joblib	Best pretrained model found by grid search algorithm.
/model/output.joblib	Prebuilt document/topic matrix transformed by the pretrained model.

/model/vector.joblib	Prebuilt term frequency vector.
/data/dataset.csv	Dataset containing skill's description list.
usertest.txt	List of job's descriptions to test.

Software requirements are the same as section 1.

Note: usertest.txt should followed the below format:

- Each line contains one job's description.
- Only the document's content should be saved in this file.

Step 2: Run the notebook named implicit_skills_extractor.ipynb.

Step 3: The potential implicit skills will be shown for each job description.

Contribution of Team Members

The following table shows the task and the member in charge for the task.

No	**Task**	**Hours allocated**	**Member in-charge**
1a	Crawl job description	5	HG
1b	Crawl skill description	1	TG
2	Dataset preparation and cleaning	4	HG, TG
3	Topic modelling	10	HG
4	Topic clustering	10	TG
5	Analysis of result	5	HG, TG
6	Code clean-up, documentation and voiced presentation	5	HG, TG
Total hours		**40**	