M7003_finalpiece_24000114067

February 3, 2025

1 Intersect – Personalized Job Matching

LIS MASc | The Right Word | Final Piece (Choice 2: NLP)
Student number: 24000114067

Access the web app, PDF, notebook, and data at the GitHub repository.

1.1 Overview

Intersect (web app) is a job-matching tool leveraging NLP to reorder job postings based on semantic similarity rather than traditional keyword searches. Unlike lexical search (BM25), which relies on exact word matches, semantic search uses dense vectors to capture meaning (Boykis, 2023; Mitchell, 2019; Schmidt, 2015), providing more personalized results when used with user-provided text. By comparing different retrieval techniques (semantic search, lexical search, reranking), *Intersect* aims to enhance job discovery and reduce manual effort.

1.2 Data and Methodology

The dataset consists of job postings scraped from *CV-Library*, a job board referenced by the UK Government's Career Advice. Search queries (`ai`, `leadership`, `fun`, etc.) were chosen based on student input. Each job description was embedded into a vector space for comparison with user queries. The process involves:

- Scraping job listings and vectorizing results with OpenAI's `text-embedding-3-small`.
- Generating word clouds with TF-IDF.
- Capturing user input and computing similarity via dot product.
- Visualizing clusters using PCA and KMeans.
- Reordering results using BM25 (lexical search).
- Reranking with Cohere's cross-encoder.

Named entity recognition and LLM-based permutation (Sun et al., 2024) were tested but discarded due to inefficiency and poor perfomance.

1.2.1 Tools

The cluster visualization uses PCA for dimensionality reduction and KMeans for clustering due to their simplicity, with two dimensions for easier interpretation. The number of clusters is flexible, but t-SNE or LSA might be better alternatives. Since the true number of clusters is unknown, MeanShift or DBSCAN could be more suitable.

For lexical search, BM25 —a modified version of TF-IDF— is used, ranking significant words higher by penalizing common terms. Preprocessing includes lowercasing, stopword removal, stemming (preferred over lemmatization for speed), and tokenization, while numbers and special characters are retained for simplicity.

Cross-encoding is similar to embedding, but is computationally expensive since it does not precompute the vectors. Cohere's reranker model is used for convenience.

Embeddings are generated using OpenAI's model, and the website is built with Streamlit. Proprietary models can be replaced with free, local, open-source alternatives, and the site can be self-hosted if needed.

1.3 Results

[ ]:
```python
from datetime import datetime, timezone
import pandas as pd
import streamlit as st
from dotenv import load_dotenv
from openai import OpenAI

from intersect.embedding import get_embedding
from intersect.utils import add_you, add_index
from intersect.read_pdf import get_text_from_pdf
from intersect.semantic_search import similarity_search
from intersect.cluster_viz import pca_df, get_chart, add_clusters
from intersect.lexical_search import lexical_search
from intersect.rerank import rerank_cohere
from intersect.tfidf import wordcloud_tfidf, tfidf_words
from intersect.ner import wordcloud_ner, ner_count
from intersect.permutation import permutation_openai

# code repurposed from the web app source code


def open_and_preprocess_db(_db_name):
    # very clean code!

    def get_db_filepath(db_name: str) -> str:
        return f"intersect/data/{db_name}.feather"

    original_df = pd.read_feather(get_db_filepath(_db_name))
    original_df = original_df.dropna()
    original_df = original_df.drop_duplicates(subset=["description"])
    original_df["i_relevance"] = original_df.index
    original_df["id"] = original_df.index

    # add days since posted
    original_df["timestamp"] = pd.to_datetime(original_df["posted"], utc=True)
```

```
    now = datetime.now(timezone.utc)
    original_df["days_ago"] = (now - original_df["timestamp"]).dt.days  # type:␣
 ↪ignore
    return original_df.copy(deep=True)


def get_input_text(filename: str) -> str:
    path = f"intersect/data/cvs/{filename}.txt"
    with open(path, "r") as f:
        return f.read()


load_dotenv()
table_size = 5
n_clusters = 1
```

While we can't evaluate the quality of this service in a quantitative way (without the time-consuming process of developing a benchmark or using something like BEIR), we can offer some qualitative observations by testing it manually and reporting interesting findings.

An item in the database looks like this:

[31]:
```
open_and_preprocess_db("ai").iloc[0]
```

[31]:
```
title                                                AI Engineer
company                            Randstad Technologies Recruitment
location                                                    London
salary                                        £80,000 - £130,000/annum
type                                                     Permanent
posted                                           2024-12-09T11:34:31Z
job_industry                                             Engineering
abstract        Job Title: AI Engineer. Location: Central Lond…
url             https://www.cv-library.co.uk/job/222767113/AI-…
description     Job Title: AI EngineerLocation: Central London…
embedding       [-0.03973353, 0.009314176, 0.030347656, -0.024…
i_relevance                                                    0
id                                                             0
timestamp                                2024-12-09 11:34:31+00:00
days_ago                                                      56
Name: 0, dtype: object
```

And the original keyword search results are displayed to the user like this:

[32]:
```
def view_relevance(text: str) -> pd.DataFrame:
    df = open_and_preprocess_db(text)
    view_relevance = df[["id", "title", "company", "days_ago", "description",␣
 ↪"url"]]
    return view_relevance.head(table_size)
```

```
view_relevance("ai")
```

[32]:

	id	title	company	days_ago	description	url
0	0	AI Engineer	Randstad Technologies Recruitment	56	Job Title: AI EngineerLocation: Central London…	https://www.cv-library.co.uk/job/222767113/AI-…
1	1	AI Solution Architect	GCS Ltd	49	AI Solution Architect - 18-month Contract - Fe…	https://www.cv-library.co.uk/job/222799912/AI-…
2	2	AI Project Manager	In Technology Group	27	Role: AI Project Manager Location: City of Lon…	https://www.cv-library.co.uk/job/222863903/AI-…
3	3	Senior AI Engineer	Platform Recruitment	42	London- Senior AI Engineer - £60 - 80k - AI An…	https://www.cv-library.co.uk/job/220343944/Sen…
4	4	Head of AI - Robotics	Lawrence Harvey	31	Head of AI - Robotic Autonomy Are you passiona…	https://www.cv-library.co.uk/job/222845101/Hea…

1.3.1 Word cloud

Although TF-IDF is designed to address this issue, we consistently observe that the word clouds are filled with overly generic words which are uninformative.

[40]:
```
from intersect.tfidf import nb_wordcloud_tfidf
import matplotlib.pyplot as plt

%matplotlib inline

df = open_and_preprocess_db('ai')
wc = tfidf_words(df["description"].tolist())
wcdf = pd.DataFrame(list(wc.items()), columns=["Word", "Frequency"])
nb_wordcloud_tfidf(wc)
plt.show();
```

```
<Figure size 28800x16800 with 0 Axes>
```

1.3.2 Semantic search

When searching the `ai` database using a fake cv of a nurse, the first result is for a nursing job lost inside the ai database.

[41]:
```python
def view_semantic(text, db):
    input_text = get_input_text(text)
    df = open_and_preprocess_db(db)

    input_embedding = get_embedding(OpenAI(), input_text)
    df = similarity_search(df, input_embedding)  # type: ignore
    df = add_index(df, "score_semantic", "i_semantic")

    view_semantic = df[
        [
            "id",
            "i_semantic",
            "title",
            "company",
            "days_ago",
            "description",
            "url",
        ]
    ]

    return view_semantic.head(table_size)


df = view_semantic("nurse", "ai")
df
```

[41]:

	id	i_semantic	title	company	days_ago	description	url
0	71	0	Nurse - Private Health Assessments	Zest Business Group	49	Zest Scientific is selecting personable and cl…	https://www.cv-library.co.uk/job/222797744/Nur…
1	113	1	Chief Executive Officer	The British Association of Urological Surgeons	49	Chief Operating Officer\n\nWe are looking for …	https://www.cv-library.co.uk/job/222797264/Chi…
2	17	2	Senior Digital Project Manager, AI, Mainly Remote	Carrington Recruitment Solutions	43	Senior Digital Project Manager, Portfolio, Pro…	https://www.cv-library.co.uk/job/222823191/Sen…
3	120	3	Product Specialist Graduate Level	RedTech Recruitment Ltd	47	Product Specialist - Graduate Level\n\nA brill…	https://www.cv-library.co.uk/job/220945299/Pro…
4	98	4	Senior Contracts Recruiter	Aurora Samuels Associates	61	Aurora Samuels Associates is recruiting for a …	https://www.cv-library.co.uk/job/222423333/Sen…

In general, the semantic search results are very different from the original results. Informal testing seems to indicate that the semantic search is more relevant to the user's query than the keyword search. For example, with my CV, the results from the similarity search talk about python and law - subjects relevant to my experience, while the keyword search has generic AI engineering jobs.

[42]:
```
view_relevance("ai")
```

[42]:

	id	title	company	days_ago	description
0	0	AI Engineer	Randstad Technologies Recruitment	56	Job Title: AI EngineerLocation: Central London…
1	1	AI Solution Architect	GCS Ltd	49	AI Solution Architect - 18-month Contract - Fe…
2	2	AI Project Manager	In Technology Group	27	Role: AI Project Manager Location: City of Lon…
3	3	Senior AI Engineer	Platform Recruitment	42	London- Senior AI Engineer - £60 - 80k - AI An…
4	4	Head of AI - Robotics	Lawrence Harvey	31	

```
4  Head of AI - Robotic Autonomy Are you passiona…

                                                 url
0  https://www.cv-library.co.uk/job/222767113/AI-…
1  https://www.cv-library.co.uk/job/222799912/AI-…
2  https://www.cv-library.co.uk/job/222863903/AI-…
3  https://www.cv-library.co.uk/job/220343944/Sen…
4  https://www.cv-library.co.uk/job/222845101/Hea…
```

[43]:
```
view_semantic("g", "ai")
```

[43]:
```
    id  i_semantic                                              title  \
0   88           0                              Senior Python Developer
1   97           1  Senior Software Engineer - Robotics - Navigation
2   89           2             Lead Software Engineer - Manipulation
3   94           3                     Senior Legal Engineer - GenAI
4  124           4                              Senior Data Scientist

                          company  days_ago  \
0  TalentTrade Recruitment Limited        59
1                Proactive Global        48
2                Proactive Global        61
3   Ignite Digital Search Limited        46
4            Xpertise Recruitment        62

                                        description  \
0  Senior Python Developer\n\n\n£75,000 + Bonus +…
1  The MissionProactive Global have partnered wit…
2  About us:In a world where artificial intellige…
3  Senior Legal Technologist / Senior Legal Engin…
4  Job Title: Senior Data Scientist - HealthAbout…

                                                 url
0  https://www.cv-library.co.uk/job/222760275/Sen…
1  https://www.cv-library.co.uk/job/222803172/Sen…
2  https://www.cv-library.co.uk/job/222748959/Lea…
3  https://www.cv-library.co.uk/job/222664179/Sen…
4  https://www.cv-library.co.uk/job/222743898/Sen…
```

1.3.3 Lexical search

Lexical search gives interesting results as well, although less obviously relevant to the user query.

[44]:
```
def view_lexical(text: str, db_name: str) -> pd.DataFrame:
    input_text = get_input_text(text)
    df = open_and_preprocess_db(db_name)
    df = lexical_search(input_text, df)
```

```
    view_lexical = df.sort_values(by="score_lexical", ascending=False)
    view_lexical = view_lexical[
        [
            "id",
            "i_lexical",
            "score_lexical",
            "title",
            "company",
            "days_ago",
            "description",
            "url",
        ]
    ]
    return view_lexical.head(table_size)


view_lexical("g", "leadership")
```

[44]:

	id	i_lexical	score_lexical	title	company	days_ago	description	url
72	89	0	38.582642	Head of Clinical Services	Castlefield Recruitment	21	Castlefield Recruitment is proud to be partner…	https://www.linkedin.com/jobs/view/3...
201	268	1	37.339005	Computer Science Teacher	Philosophy Education	31	Computer Science Teacher\nFull-time \nEnfield …	https://www.linkedin.com/jobs/view/3...
313	440	2	34.876892	Backend Software Engineer Python AI SaaS	Client-Server	28	Backend Software Engineer / Developer (Python …	https://www.linkedin.com/jobs/view/3...
2	2	3	34.559830	Principal Technologist - [Artificial Intellige…	Summer Browning Associates	24	Summer-Browning Associates are seeking a Princ…	https://www.linkedin.com/jobs/view/3...
314	441	4	34.000317	Professional Services Lead - Data and AI	83zero Ltd	27		https://www.linkedin.com/jobs/view/3...

```
314  We are seeking an experienced and highly motiv…
```

```
                                                   url
72   https://www.cv-library.co.uk/job/222893889/Hea…
201  https://www.cv-library.co.uk/job/222846509/Com…
313  https://www.cv-library.co.uk/job/222852443/Bac…
2    https://www.cv-library.co.uk/job/222888659/Pri…
314  https://www.cv-library.co.uk/job/222861214/Pro…
```

1.3.4 Dimensionality reduction and clustering

Most of the different databases show just a cloud of points that does not suggest anything in particular.

[45]:
```
def view_embedding(text: str, db_name: str, d):
    input_text = get_input_text(text)
    df = open_and_preprocess_db(db_name)

    input_embedding = get_embedding(OpenAI(), input_text)
    df = similarity_search(df, input_embedding)  # type: ignore
    df = add_index(df, "score_semantic", "i_semantic")

    df_without_you = df.copy()
    df_you = add_you(df_without_you, input_text, input_embedding)  # type:␣
↪ignore
    df_pca = pca_df(df_you, "embedding", n_components=2)

    def generate_chart(_df: pd.DataFrame, n_clusters: int):
        _df = add_clusters(df_pca, n_clusters, n_components=2)
        _df.loc[_df["title"] == "Your text", "Cluster"] = " You"
        return get_chart(_df)

    return generate_chart(df_pca, d)
```

[46]:
```
view_embedding("g", "ai", 1)
```

```
/tmp/ipykernel_143332/534672769.py:15: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value ' You' has dtype incompatible with int32, please explicitly cast
to a compatible dtype first.
  _df.loc[_df["title"] == "Your text", "Cluster"] = " You"
```

[46]:
```
alt.Chart(…)
```

…

However, in two cases, there seems to be clear clusters which are not obvious from the keyword search: `fun` and `leadership`

…

[47]:
```
view_embedding("g", "fun", 3)
```

```
/tmp/ipykernel_143332/534672769.py:15: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value ' You' has dtype incompatible with int32, please explicitly cast
to a compatible dtype first.
  _df.loc[_df["title"] == "Your text", "Cluster"] = " You"
```

[47]:
```
alt.Chart(…)
```

[48]:
```
view_embedding("g", "leadership", 3)
```

```
/tmp/ipykernel_143332/534672769.py:15: FutureWarning: Setting an item of
incompatible dtype is deprecated and will raise an error in a future version of
pandas. Value ' You' has dtype incompatible with int32, please explicitly cast
to a compatible dtype first.
  _df.loc[_df["title"] == "Your text", "Cluster"] = " You"
```

[48]:
```
alt.Chart(…)
```

This gives us some insight into how these generic keywords. In `fun`'s case, they correspond to teaching, management and healthcare roles, while in `leadership`'s case, they are related to teaching, management and engineering.

1.3.5 Reranking

Reranking gives different results, but they did not display a remarkable improvement upon other methods.

[49]:
```
def view_reranked(text: str, db_name: str):
    input_text = get_input_text(text)
    df = open_and_preprocess_db(db_name)
    df = rerank_cohere(input_text, df)
    df = add_index(df, "score_reranker", new_index="i_reranker")
    view_reranked = df.sort_values(by="score_reranker", ascending=False)
    view_reranked = view_reranked[
        [
            "id",
            "i_reranker",
            "score_reranker",
            "title",
            "company",
            "days_ago",
            "description",
            "url",
        ]
    ]
```

```
    return view_reranked.head(table_size)

view_reranked("g", "ai")
```

[49]:

	id	i_reranker	score_reranker	title	company	days_ago	description	url
0	137	0	0.170882	Consultant	Vertical Advantage Limited	62	As one of the world's fully diversified data s…	https://www.cv-library.co.uk/job/222741990/Con…
1	37	1	0.070541	Head of Data Science	83zero Ltd	27	Job Title: Head of Data ScienceSalary: £100,00…	https://www.cv-library.co.uk/job/222860781/Hea…
2	74	2	0.066721	Data Engineer	Randstad Technologies Recruitment	56	Job Title: Data EngineerLocation: Central Lond…	https://www.cv-library.co.uk/job/222766927/Dat…
3	76	3	0.065779	Data Scientist/ Analyst Developer	Guidant Global	47	Job Title- Data Scientist/ Analyst DeveloperJo…	https://www.cv-library.co.uk/job/222806046/Dat…
4	39	4	0.057266	Full Stack Developer	SmartSourcing plc	60	**2X FULL STACK DEVELOPER**3 MONTHS WITH POSSI…	https://www.cv-library.co.uk/job/222753572/Ful…

1.4 Conclusion

Intersect uncovers non-obvious job opportunities by enhancing traditional search methods with NLP. The varied outcomes suggest a hybrid approach—combining keyword, semantic, and reranking techniques—could yield optimal results.

Future improvements include real-time scraping, LLM-enhanced reranking, visa sponsorship tagging, and CSV export functionality. With UI/UX refinements and integration with multiple job boards, *Intersect* could evolve into a viable product.

1.5 References

- Boykis, V. (2023). *What are embeddings?*. Retrieved from https://github.com/veekaybee/what_are_embeddings
- Schmidt, B. (2015). *Vector Space Models for the Digital Humanities*. Bookworm. Retrieved from https://bookworm.benschmidt.org/posts/2015-10-25-Word-Embeddings.html

- Mitchell, M. (2019). *Artificial Intelligence: A Guide for Thinking Humans*. Pelican Books.
- Sun, W., Yan, L., Ma, X., Wang, S., Ren, P., Chen, Z., Yin, D., & Ren, Z. (2024). *Is ChatGPT Good at Search? Investigating Large Language Models as Re-Ranking Agents* (No. ArXiv: 2304.09542). ArXiv. https://doi.org/10.48550/arXiv.2304.09542

This work contains around 500 words.

AI was used for summarization purposes.