Progress Report: ConnectGlobal

An AI-Powered Career Support for International Students in the United States of America

Problem Statement

International students face many barriers in the U.S. job search with sponsorship and OPT opportunities being uncertain and not always clear, as well as a lack of connections resulting in a weak network and less opportunities. Cold applications rarely work for anyone, even U.S. citizens, so adding these extra issues on top of that can dishearten many students, leading them to move back home after university.

Primary Goals

3 Primary Goals:

1. Job Matching System
    - Match student resumes with visa-friendly job postings
2. Networking Recommender
    - Graph-based system to connect students with relevant professionals
3. Reduce Job Search Stress
    - Increase interview callback rates using job matching system

Workflow

Data Preprocessing

Job Postings
- Kaggle[5]
- LinkedIn
- Glassdoor

Resumes Samples
- Synthetic or
- Anonymized

Synthetic Networking Data
- Country
- Skills
- Industry

Resume-job similarity model

TF-IDF
- Baseline semantic similarity

BERT Embeddings[4]
- Deep semantic similarity

Model
- Supervised classification
- Outputting top-k matched jobs

Networking Recommender

Graph Construction
- Nodes = people
- Edges = shared factors

Graph Clustering
- Identifying communities of similar professionals

Link Prediction
- Suggest likely helpful connections

Evaluation

Job Matching
- Precision/recall
- Compare TF-IDF to BERT embeddings

Usefulness for networking
- Measured by relevance of suggested connections

Data Collection & Preprocessing

- Used Kaggle for raw datasets
- Job postings
  - Implemented keyword filtering system
  - Looks for words such as “H1-B”, “OPT”, “sponsorship”, etc. in job postings
  - Adds a new column to each posting stating “Yes”, “No”, or “Unknown”
- Professional profiles
  - Created fake profiles with attribute “Country”
- Resumes
  - Generated realistic samples for testing

Data Collection & Preprocessing

```
# select relevant fields
df = df[['job_posted_date', 'company_address_locality', 'company_name', 'company_description', 'job_de

# add a new field, visa_sponsorship
df['visa_sponsorship'] = 'Unknown'

# preview new dataframe
print(df.head(5))

# keyword-based filter
# set our keywords_yes and keywords_no
visa_keywords_yes = ['h1-b', 'h1b', 'visa sponsorship',
                     'sponsorship available', 'work visa', 'opt', 'cpt',
                     'stem opt', 'sponsorship']

visa_keywords_no = ['must be authorized to work in the us', 'no sponsorship',
                    'without sposorship', 'not sponsor', 'usc or gc only',
                    'citizens only', 'green card only']

# combining company description and job description for future use with searching for keywords
df['job_text'] = df[['company_description', 'job_description_text']].astype(str).agg(' '.join, axis=1)
```

Job Matching System

- TF-IDF Model
  - Cosine similarity calculation between resumes and job postings
  - Ranks top-5 job matches per resume
- BERT Model
  - Integrated sentence-transformers library
  - Deeper understanding than keyword matching
  - Ranks top-5 job matches per resume
- Comparison
  - Both process entire dataset
  - TF-IDF faster but BERT deeper understanding

Job Matching System

```
# loops through each resume by index instead of the sparse matrix directly
for i in range(resume_vecs.shape[0]):
    current_resume_vec = resume_vecs[i:i+1, :]
    # computes the similarity of each resume and each job posting
    # returns an array of similarity scores
    sims = cosine_similarity(current_resume_vec, job_vecs)[0]
    # sorts jobs by similarity score, descending
    # picks the top n jobs which is set at 5
    top_indices = sims.argsort()[::-1][:top_n]
    # store the results
    results[i] = {
        'top_jobs': [
            {
                # .iloc[idx] return the row at position idx from the jobs dataframe
                'jobs_title': jobs.iloc[idx]['job_title'],
                'company_name': jobs.iloc[idx]['company_name'],
                'visa_sponsorship': jobs.iloc[idx].get('visa_sponsorship', 'Unknown'),
                # returns the similarity score at idx from the sims array
                'similarity_score': float(sims[idx]),
            }
            for idx in top_indices
        ]
    }
return results
```

Job Matching System


```
TF-IDF results: {'top_jobs': [{'jobs_title': 'Machine Learning Engineer - Santa Clara County, California, United States', '
company_name': 'Sibitalent Corp', 'visa_sponsorship': 'Yes', 'similarity_score': 0.12832756623809208}, {'jobs_title': 'Data
 Scientist Internship', 'company_name': 'Applied Concepts, Inc.', 'visa_sponsorship': 'Yes', 'similarity_score': 0.12320632
19221294}, {'jobs_title': 'Data Scientist Internship', 'company_name': 'Applied Concepts, Inc.', 'visa_sponsorship': 'Yes',
 'similarity_score': 0.1232063219221294}, {'jobs_title': 'Intern, Machine Learning Engineer - VLMs', 'company_name': 'Samsu
ng Semiconductor', 'visa_sponsorship': nan, 'similarity_score': 0.10660656888868918}, {'jobs_title': 'Machine Learning Engi
neer', 'company_name': 'ArcheHealth', 'visa_sponsorship': 'Yes', 'similarity_score': 0.08362526952937267}]}
BERT results: [{'job_title': 'Data Scientist (Python)', 'company_name': 'LogicMatrix', 'visa_sponsorship': nan, 'similarity
_score': 0.5932486653327942}, {'job_title': 'Data Scientist (Python)', 'company_name': 'LogicMatrix', 'visa_sponsorship': n
an, 'similarity_score': 0.5932485461235046}, {'job_title': 'Machine Learning Engineer', 'company_name': 'Givzey', 'visa_spo
nsorship': 'Yes', 'similarity_score': 0.5905258059501648}, {'job_title': 'Data Scientist', 'company_name': 'hackajob', 'vis
a_sponsorship': 'Yes', 'similarity_score': 0.542746901512146}, {'job_title': 'Machine Learning Engineer', 'company_name': '
Inference.ai', 'visa_sponsorship': nan, 'similarity_score': 0.528724193572998}]
```


- Job title, Company name, Visa Sponsorship, Similarity score

Networking Recommendation System

- Graph Construction
  - NetworkX graph with 500 professional nodes
  - Edges created based on skill similarity
    - Cosine Similarity > 0.3
- Recommendation Algorithm
  - Weighted scoring system
    - Skill-based similarity
    - Country-based bones (+0.3)
  - Returns top-3 professional connections

Networking Recommendation System

```
# cosine_similarity computes the cosine similarity between every profile's TF-IDF skill vector proc
# cosine similarity quantifies how close two profiles' skill vectors point in the same direction
sims = cosine_similarity(skill_vectors)

# we loop through each id in df for the length of df
for i in range(len(df)):
    # we loop through again starting at 2 in order to avoid duplicates in order to compare this to
    for j in range(i+1, len(df)):
        # if the cosine similarity is above 0.3 then we add an edge linking the two nodes, storing
        # this shows 1. if they are well connected and 2. how well connected they are
        if sims[i, j] > 0.3:
            G.add_edge(df.loc[i, 'id'], df.loc[j, 'id'], weight=sims[i, j])

return G
```

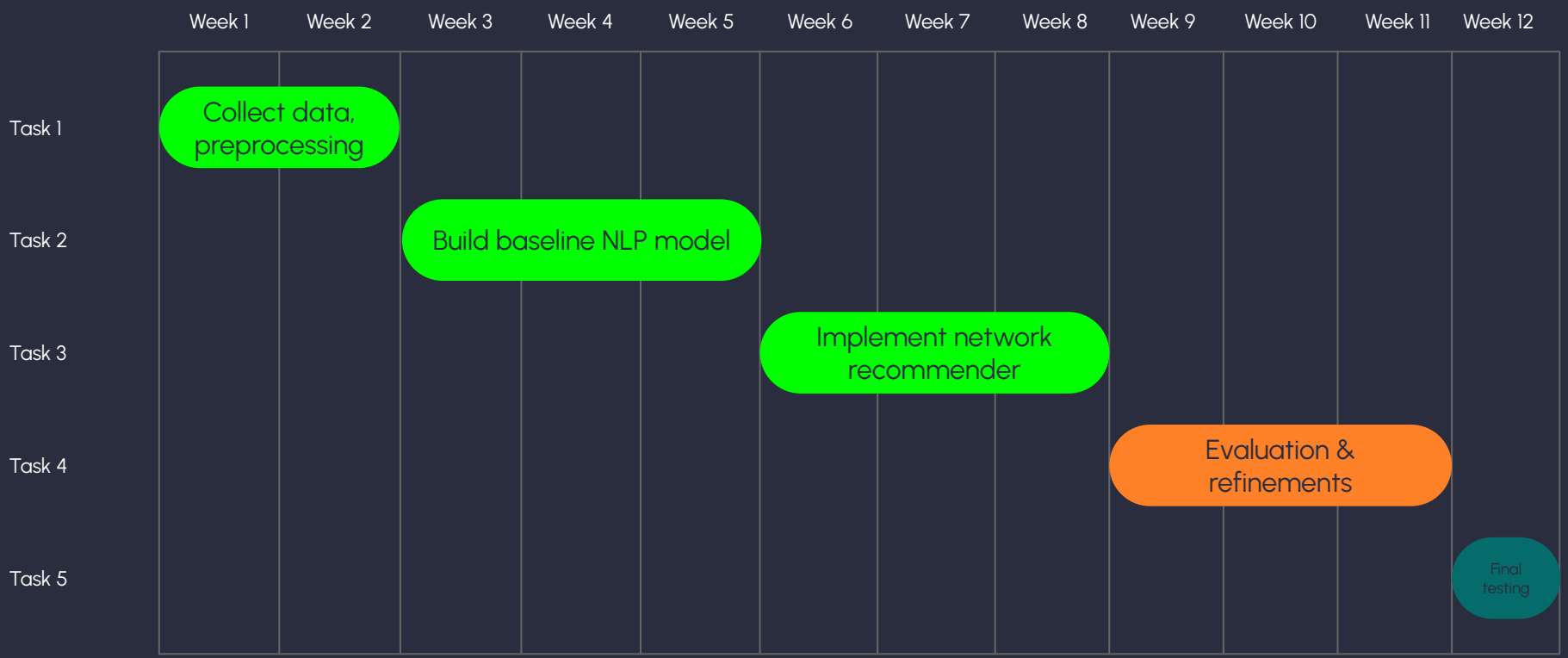
Networking Recommendation System

	name	company	country	skills	score
0	Robert Mcfarland	Lopez-Arroyo	China	Linux,Ethical Hacking,Python,Network Security	1.10
1	Miss Nancy Baird	Chambers-Peterson	China	Linux,Network Security,Ethical Hacking,Python,...	1.03
2	Catherine Burke	Wilson Ltd	Pakistan	Python,Linux,Network Security	1.00

Challenges

- Visa Sponsorship Detection
  - No consistent mention of visa sponsorship
  - Positive and negative keyword check
- Resume Data Availability
  - Limited access to real resumes
  - Synthetic resume dataset created
- Network Graph Sparsity
  - Recommendation quality is reliant on the right number of connections
  - Similarity threshold implemented

Semester Timeline



Up Next

- Possible web scraping in order to use real-world resumes and professional profiles
- Integrate live job postings instead of previous years
- Create user interface

Any questions? Ask away!

Thank you