Job Seeking Helper

Hongming Wang

UNIVERSITY OF WATERLOO

Motivation

- Graduation season, seeking jobs.

- Search vs Recommendation

- Types of Recommendations

  1. Edit Manually
  2. Simple aggregate (popularity-based, recent used)
  3. **User-specific recommendation** (content-based, collaborative filtering)

UNIVERSITY OF WATERLOO

About Dataset

- Provided and sponsored by CareerBuilder.com
- Available in Job Recommendation Challenge at Kaggle
- Data including: users' information, application history, jobs' information

Content-based Recommendation

1. Overview

1. Text preprocessing

1. Data preprocessing

1. Model Training

1. Example of Recommendation Result

UNIVERSITY OF WATERLOO

Overview

- Idea:

  - Recommend Jobs to user **x** similar to previous jobs applied by **x**

- How to tell if jobs are similar -- relevant fields:

1. Job Title
2. Description
3. Requirements
4. Location (State/Province, City)

Are they equally important from the view of similarity?
How can we address their different importance in the implementation?

UNIVERSITY OF
WATERLOO

Text Preprocessing

- Scope: Title, Description, Requirements, City, State/Province

Example: **JobID** = 1; **Titile** = ‘Security Engineer/Technical Lead’;

Raw text of its **Description**:

<p>Security Clearance Required: Top Secret </p>\r<p>Job Number: TMR-447</p>\r<p>Location of Job: Washington, DC</p>\r<p>TMR, Inc. is an Equal Employment Opportunity Company</p>\r<p>For more job opportunities with TMR, visit our website www.tmrhq.com</p>\r

……

Identifies resources and mentors in-house talent to ensure TMR remains responsive to growing initiatives and contracts with qualified personnel.   \r
\r<p> </p>\r<p> </p>

Not readable and analyzable at all …

UNIVERSITY OF WATERLOO

Step 1: Match and remove html tags using Regex

<.*?>

<p>Security Clearance Required: Top Secret </p>\r<p>Job Number: TMR-447</p>\r<p>Location of Job: Washington, DC</p>\r<p>TMR, Inc. is an Equal Employment Opportunity Company</p>\r<p>For more job opportunities with TMR, visit our website www.tmrhq.com</p>\r

....

Identifies resources and mentors in-house talent to ensure TMR remains responsive to growing initiatives and contracts with qualified personnel.   \r
\r<p> </p>\r<p> </p>

UNIVERSITY OF WATERLOO

Step 2: Remove other formatting characters:

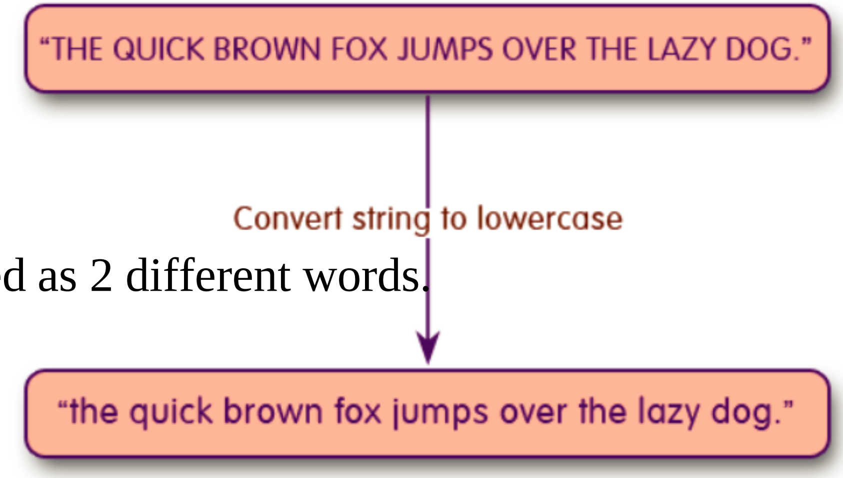
- redundant spaces, '\r'(carriage return), '\n'(newline), ' '(non-breaking space)

<p>Security Clearance Required: Top Secret </p>\r<p>Job Number: TMR-447</p>\r<p>Location of Job: Washington, DC</p>\r<p>TMR, Inc. is an Equal Employment Opportunity Company</p>\r<p>For more job opportunities with TMR, visit our website www.tmrhq.com</p>\r

....

Identifies resources and mentors in-house talent to ensure TMR remains responsive to growing initiatives and contracts with qualified personnel.
\r
\r<p> </p>\r<p><a
href="https://www.tmrhq.com/jobapplicationstep1.aspx"> </p>

Step 3: Convert the string to lowercase



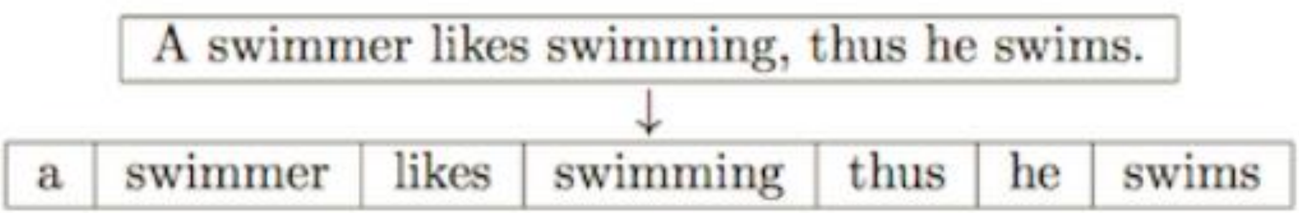
- Python is case sensitive

In this case, '**S**ecurity' and '**s**ecurity' are identified as 2 different words.

Step 4: Tokenize

- Split tokens (in this case, words) with the space delimiter

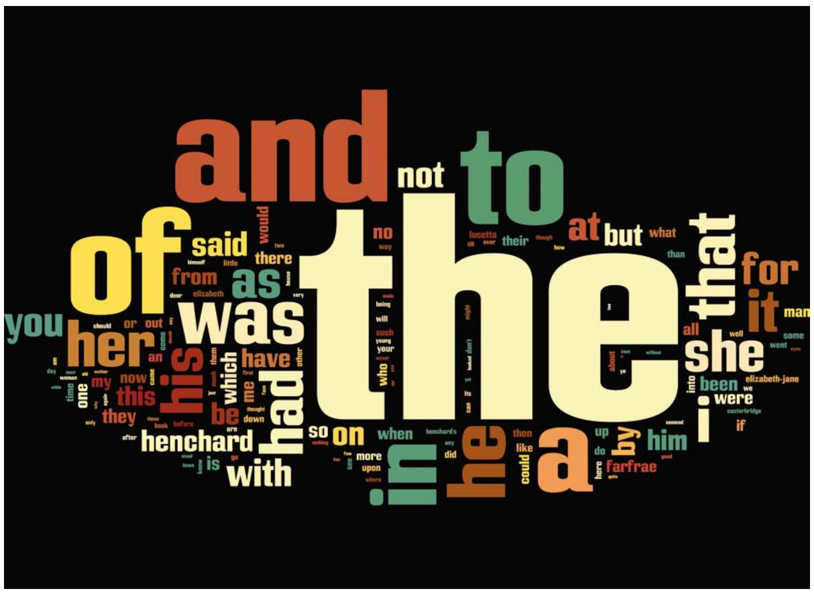
Convert string to a list of words. Also strip all punctuations.



Step 5: Filter out the stop words in English

- Including: a, an, at, the, to, of, in, she, he …

Those words contribute nothing but noise



A swimmer likes swimming, thus he swims.

↓

swimmer	likes	swimming	,	swims	.

a, **thus**, **he** are the removed stop words

Should we also do stemming?

Or should we also do lemmatization?

Step 6: Generate a combined feature for each job

- Will be used for training the model later
- Strategy of concatenation:

combined feature = preprocessed(Description) + preprocessed(Requirement) +

preprocessed(Title) * weight_title + preprocessed(City) * weight_city +

preprocessed(State) * weight_state

where weight_title = (1 + length // 100) , weight_city = (1 + length // 150) ,

weight_state = (1 + length // 250) , and

length = len(preprocessed(description)) + len(preprocessed(Requirements))

Note: (string:**str**) * (int:**n**) means string **str** repeats **n** times.

In our implementation, we assign different fields with different weights to address the importance. The order we define is:

Title > City > State > Description = Requirements

UNIVERSITY OF WATERLOO

Example of the result after preprocessing

JobID = 1; **Title** = ‘**Security Engineer/Technical Lead**’;

Description = ‘<p>Security Clearance Required> … … </p>’

Requirements = ‘<p>SKILL SET</p>… …APPLY HERE</p>’

City = ‘**Washington**’ **State** = ‘**DC**’

JobID = 1;

Feature = ‘security clearance required top secret job number tmr 447 location job washington dc tmr inc equal employment opportunity company job opportunities tmr visit website

……

related certifications education years experience bs computer science related discipline minimum 8 years security minimum 4 years senior lead position apply **security engineer technical lead security engineer technical lead security engineer technical lead washington washington dc**’

UNIVERSITY OF WATERLOO

Data Preprocessing

Concern 1: Low-quality Jobs

- Incomplete information, not unavailable
- Filter Criteria: keep in the jobs with more than 1 application.

Concern 2: Cold Start Problem

- hard do provide personalized recommendations for users with none or a very few number of consumed items
- Filter Criteria: focus on the users with more than 5 applications in the training stage

UNIVERSITY OF WATERLOO

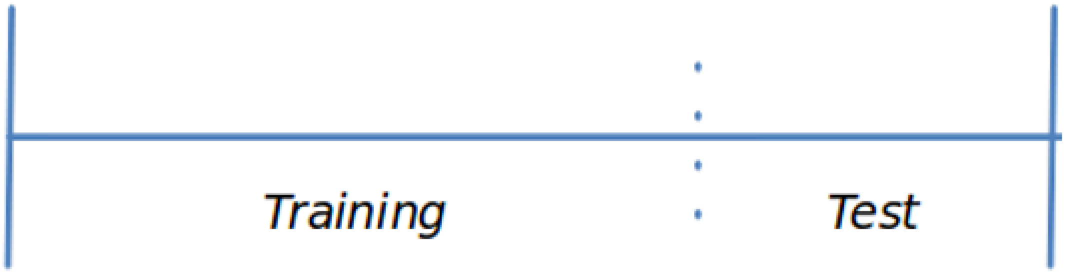
Train/Test Data Splitting

Using Cross-validation techniques?

- Ensure the generalization of data.
- **Not** in the real time.

Splitting according to a reference date

- Ensure the generalization of data.
- In the real time.
- More robust



Model Training

Goal: Create **Job Profile**

- Contains a set of important words capturing the essential
- Convert the **Feature** fields into the profile based on **TF-IDF**

Job Profile = words with high TF-IDF scores + their TF-IDF scores

TF-IDF (**T**erm **F**requency – **I**nverse **D**ocument **F**requency)

- An algorithm judging the importance of a word to a document (i.e. job) in a collection (i.e. training set).
- TFIDF("resume"), TFIDF("skill"). What about TFIDF("security")?

$$\mathrm{tfidf}_{i,j} = \mathrm{tf}_{i,j} \times \log\left(\frac{N}{\mathrm{df}_i}\right)$$

$\mathrm{tf}_{i,j}$ = total number of occurences of i in j

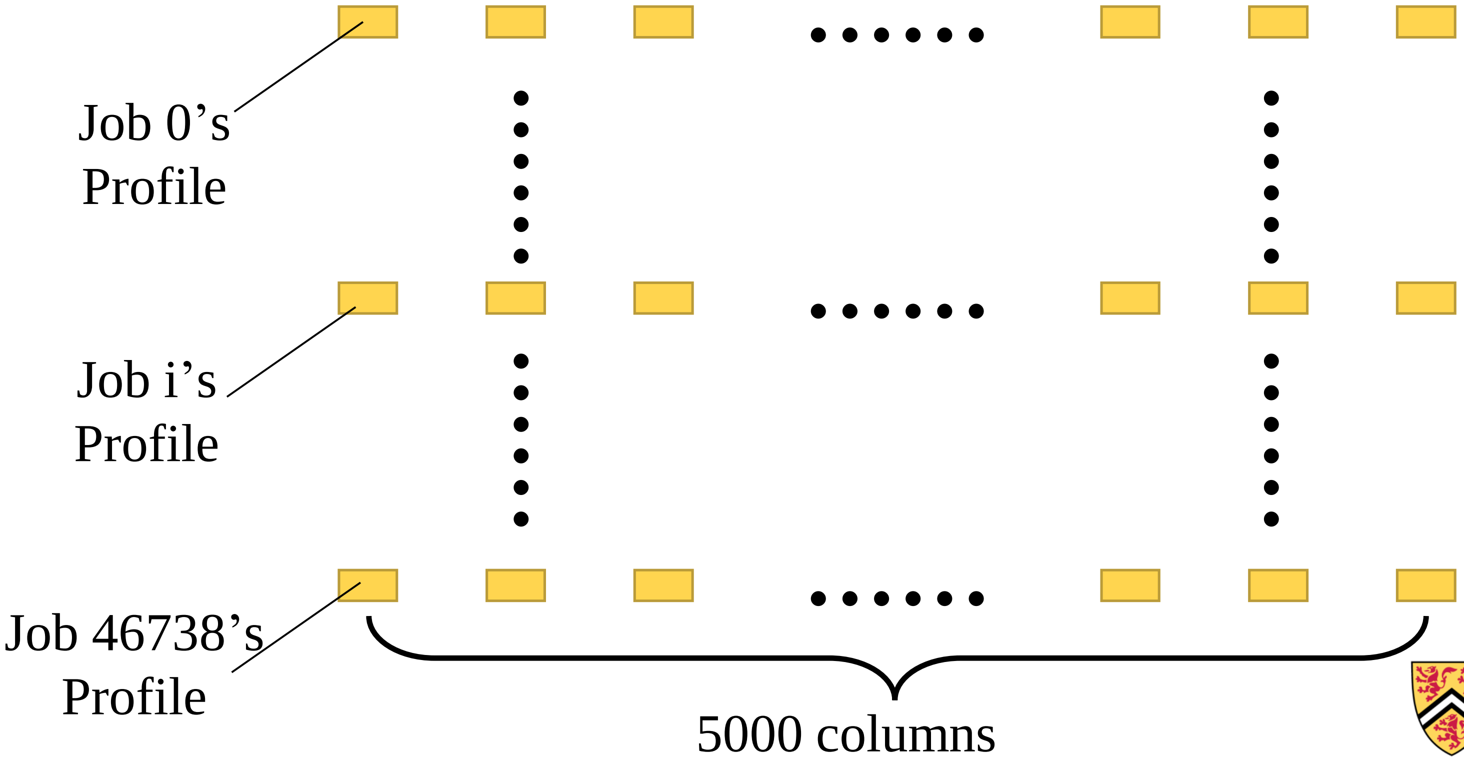
df_i = total number of documents (speeches) containing i

N = total number of documents (speeches)

UNIVERSITY OF WATERLOO

Details about the model

- A (46739, 5000) sparse matrix, where 46739 is the number of jobs, 5000 is the specified capacity of words to keep in.
- Composed by the unigrams (like "security") and bigrams (like "sales manager") in the corpus of training set.



Build User Profile

- Aggregate all profiles of the jobs applied by the user **x**.
- Then average the result.
- A row unit vector with the shape of (, 5000)

Prediction

- Calculate cosine similarity for the vectors of user **x** with every job **i**
- Sort the result list in descending order.

Example of Recommendation Result

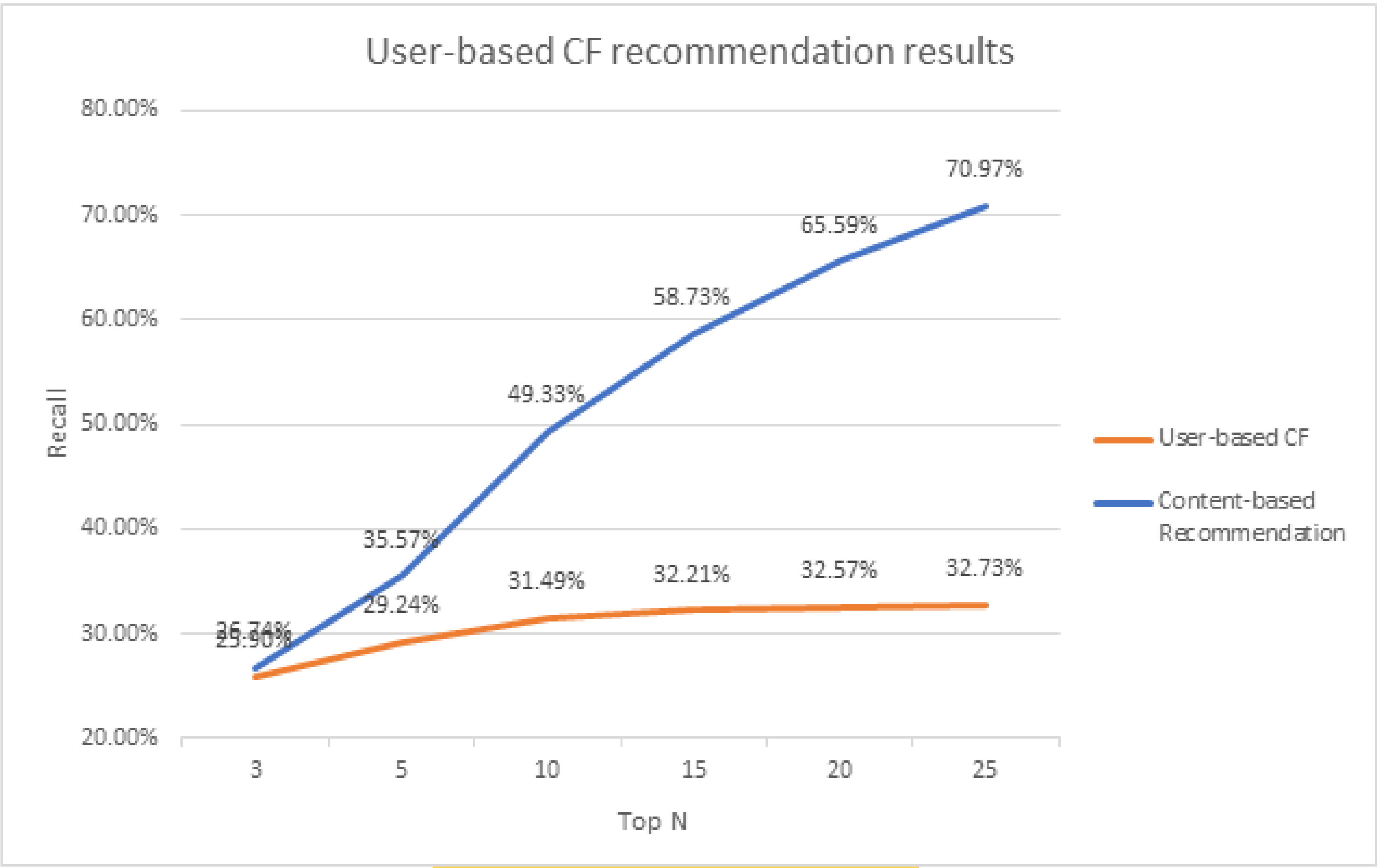
- UserID = 8189.
- Jobs applied in the training time:

1. JobID = 406104 “Online Marketing Manager”
2. JobID = 758462 “Canon Insights Program - Social Media Internship”
3. JobID = 472338 “Event Coordinator- Sales & Marketing”

	recStrength	JobID	Title
0	0.642109	563167	Administrative Assistant/ Research Assistant
1	0.632758	262678	Executive Assistant/ Marketing Assistant
2	0.611017	862963	Recent College Graduate/Administrative Assistant
3	0.611017	563172	Recent College Graduate/ Administrative Assistant
4	0.593569	562659	Administrative Assistant/ Client Services
28	0.462925	233594	Administrative Assistant, Senior Administrativ...
29	0.462643	1039146	Administrative Assistant
..	...	...	...
70	0.427255	198270	Japanese Bilingual Administrative Assistant
71	0.426223	269682	Administrative Assistant
72	0.425989	569560	Administrative Assistant
73	0.425549	917783	Administrative Assistant

Evaluation

Definition of Recall = (number of recommended jobs that are applied by user) / (total number of jobs applied by user)



Recall at TopN Recommendation

UNIVERSITY OF WATERLOO

Evaluation and Comparison

'modelName': 'Content-Based’

Pros:

1.No need for other users’ data （diffrent from collaborative filtering).

2.Able to commend new job postings

3.Explicit explanation.

Cons:

1. Difficult to create a proper **Feature**

UNIVERSITY OF WATERLOO

Future Work

1. Detection of negative rating (no interest in some jobs)
2. Expanding the set of stop words specially for the task of job recommendation

UNIVERSITY OF WATERLOO

Reference

1. http://www.grroups.com/blog/naive-bayes-and-text-classification-introduction-and-theory
2. https://www.kaggle.com/c/job-recommendation/data
3. https://www.kaggle.com/gspmoreira/recommender-systems-in-python-101