DataJobs

By Seth Chart

Business Understanding

The Problem with Job Titles

Data job titles are not well defined.

- Identical roles have different titles.
- Distinct roles have identical titles.

Data job titles are ineffective.

- Job seekers cannot trust job titles when vetting job postings.
- Employers cannot communicate roles effectively

The market needs more information.

Objectives

1. Use job descriptions to classify jobs, without referencing the job title.
2. Provide a tool that can classify a job based on its description.
3. Evaluate the accuracy of classifying jobs by their titles.

Available Resources

- LinkedIn and Indeed have recently taken steps to limit access to job postings.
  - Closing APIs
  - Adversarial web design
- Careerjet provides reasonably open access to job postings.
  - Public API
  - Accessible web design
- There are still technical barriers to accessing Careerjet job postings
  - Official Python API package is not functional
  - Reasonable limits on search (2000 results)

Technical Goals

1. Obtain a reasonably large representative corpus of job postings.
2. Produce a topic model to detect prominent topics within job descriptions.
3. Represent job descriptions as a mixture of DataSkills.
4. Use the representation of job descriptions to cluster jobs into distinct DataRoles.
5. Build a classification model that predicts DataRole from the job title.
6. Evaluate the accuracy of job titles in terms of the classification model.
7. Build DataSkills and DataRole pipelines that can be applied to unseen job postings.
8. Serve a publicly available RESTful api to provide DataSkills and DataRoles in a software as a service model.

Project Plan

Defining a way forward

Data Collection

Data Cleaning

Feature Engineering

Modeling

Deployment

Data Collection

Workflow

1. Selenium webdriver is used to navigate careerjet.com.
2. Job postings are scraped and parsed using Beautiful Soup.
3. Job title, job description, and job posting url are saved to an sqlite database for storage.

Data Cleaning

Workflow

1. Lowercase and remove newline characters.
2. Tokenize documents into sentences.
3. Tokenize sentences into words.
4. Tag words with parts-of-speech tags.
5. Lemmatize words based on parts-of-speech tags.
6. Remove stopwords and special characters.

Feature Engineering

Workflow

1. Combine common phrases into n-grams.

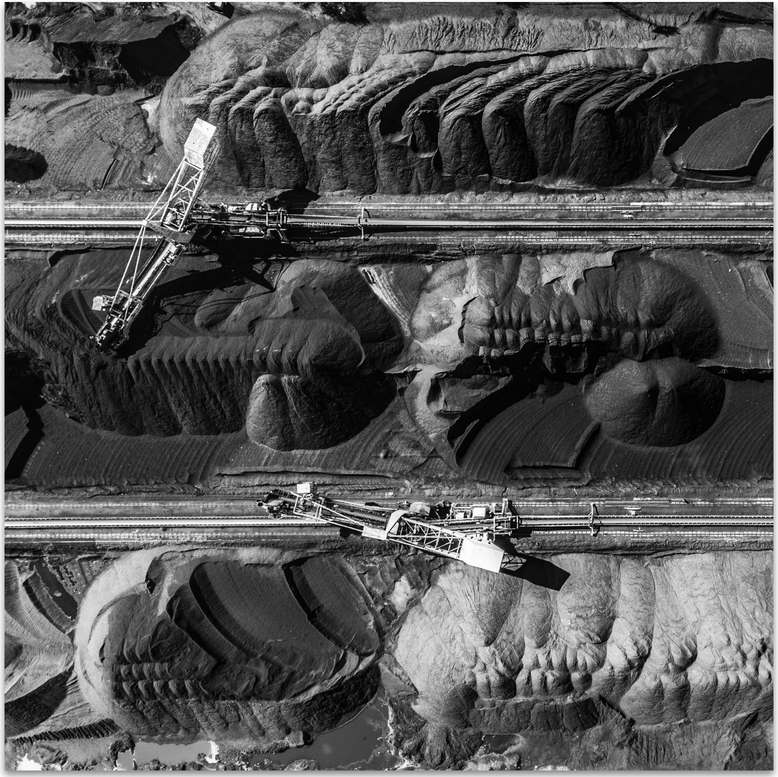
Modeling

Workflow

1. Topic model applied to job descriptions to produce DataSkills.
2. Clustering applied to DataSkills to produce DataRoles.
3. Classifier applied to job titles to predict DataRoles.

Data Collection

Obtaining the raw materials



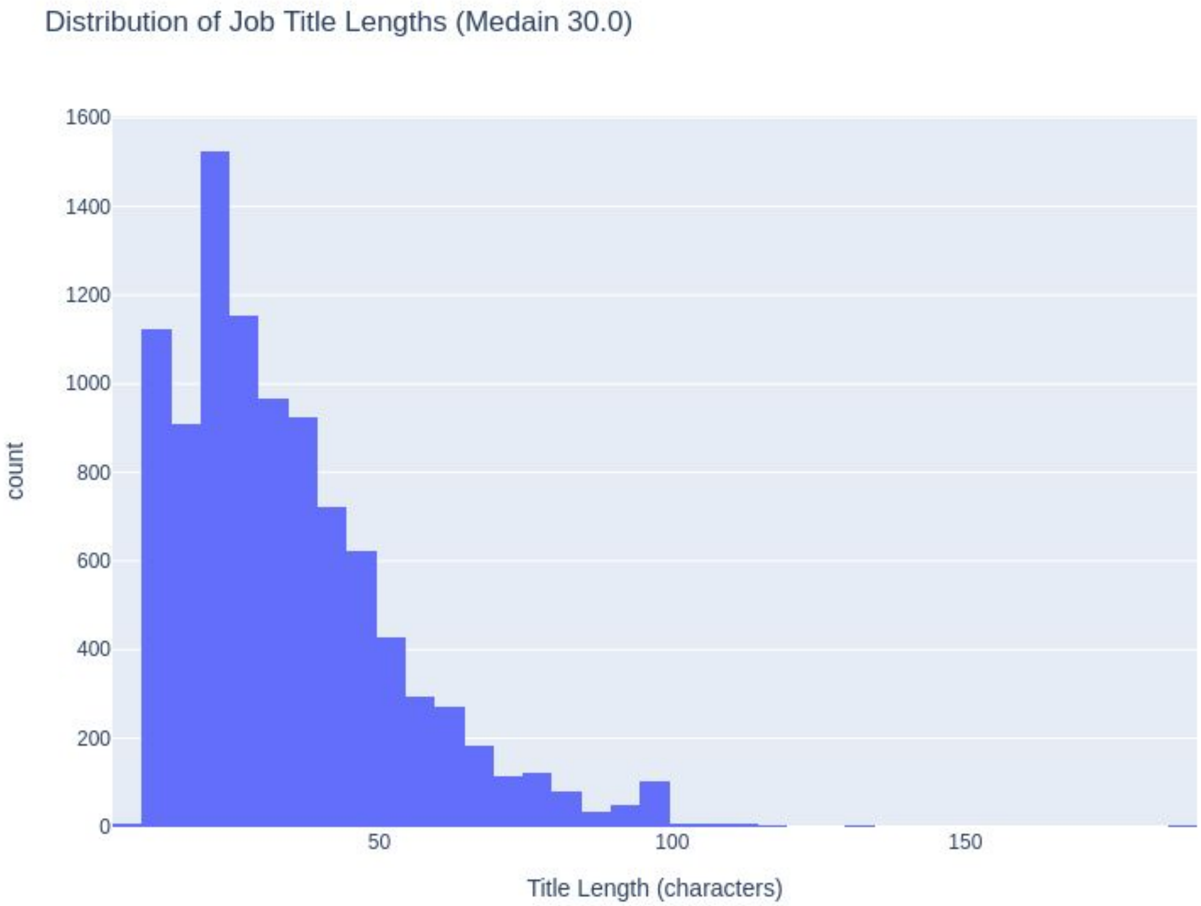
Implementation

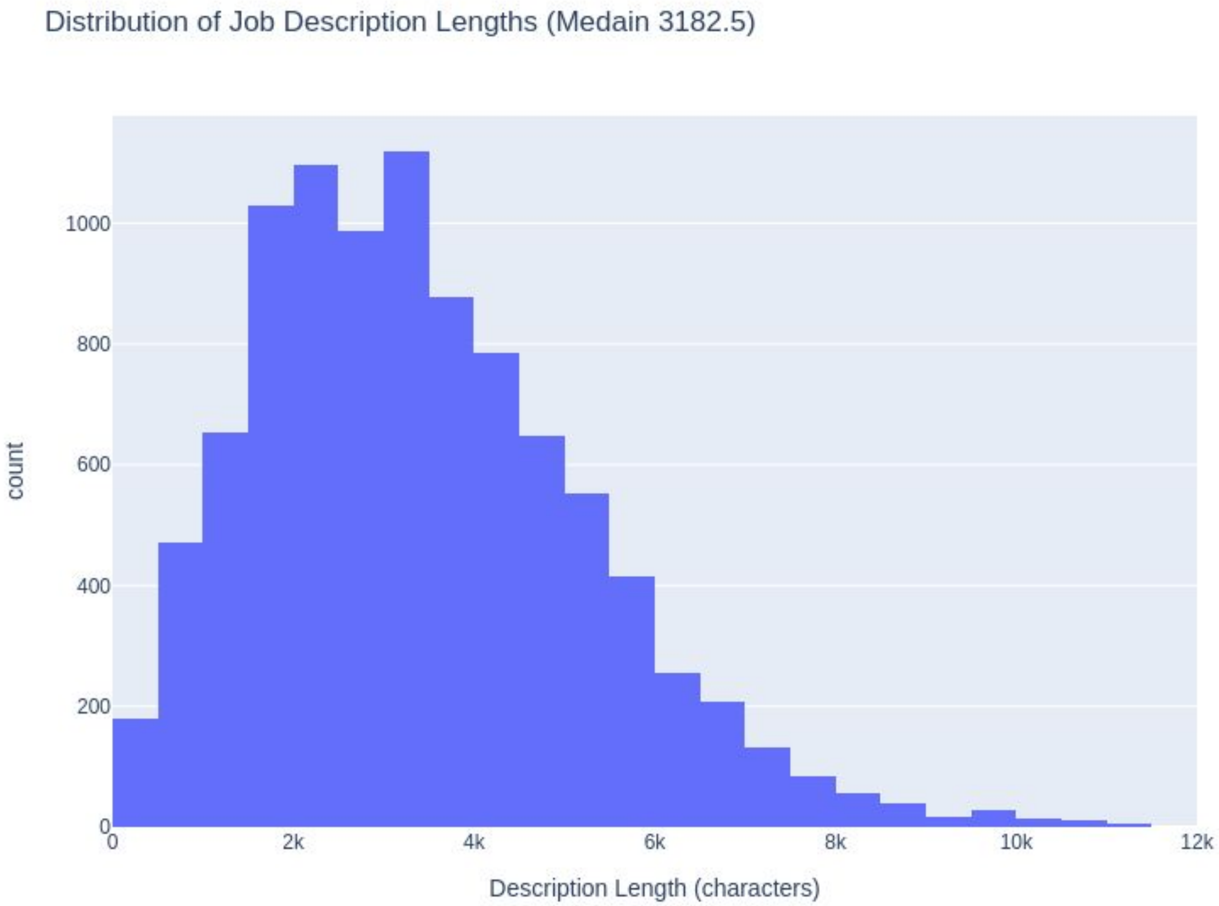
- Interaction with careerjet.com is handled by the custom careerjet module.
  - Wraps a Selenium webdriver
- Data storage is handled by the JobsDb module.
  - Wraps a SQLite3 database
- Data collection is automated by the scrape script.

Results

- Two scraping sessions produced 19,166 job postings.
- We selected 9,666 postings that contained the keyword ‘data’ for further analysis.
- The median length of a job title was 30 characters.
- The median length of a job description was 3,182.5 characters.
- All data was encoded as strings.
- There were no missing values.

This step satisfies Technical Goal 1


Distribution of Job Title Lengths (Medain 30.0)
1600
1400
1200
1000
800
600
400
200
0
count
50
100
150
Title Length (characters)



Distribution of Job Description Lengths (Medain 3182.5)

Data Cleaning

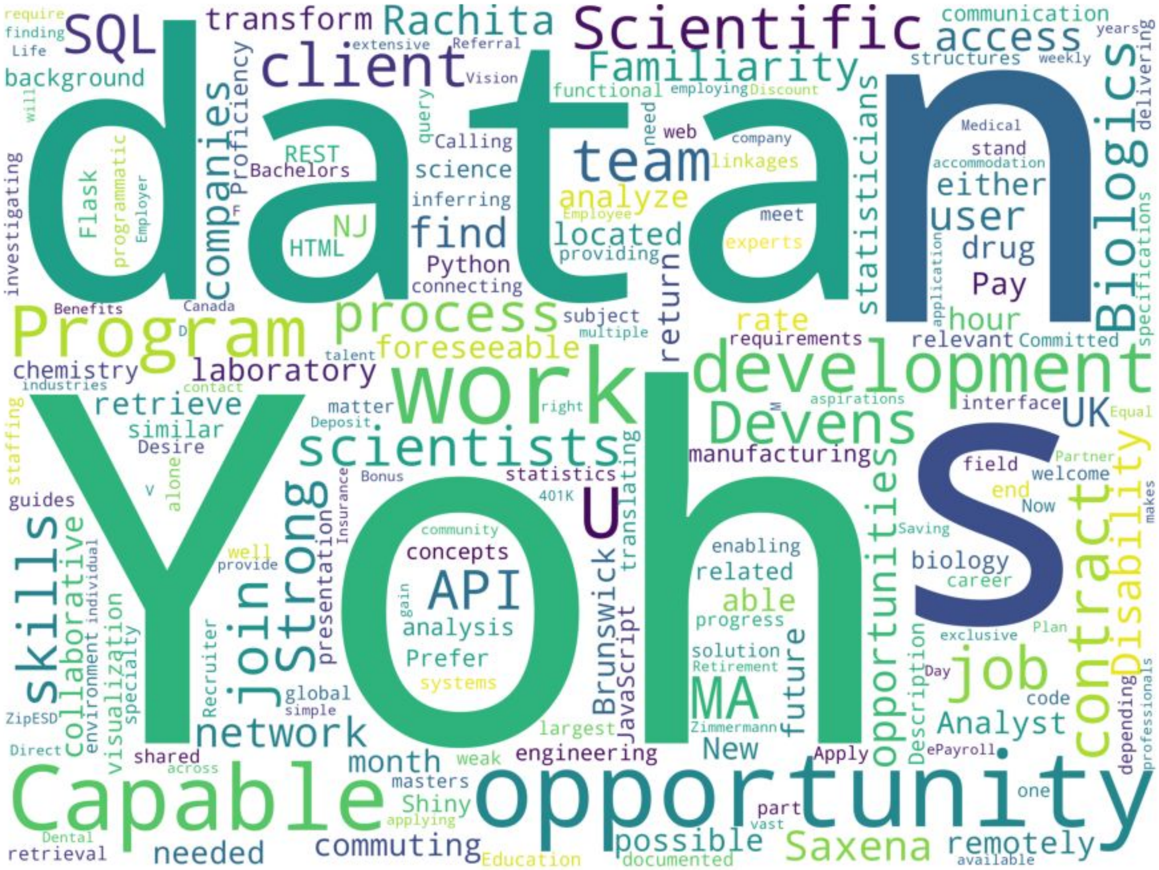
Refining the materials

Implementation

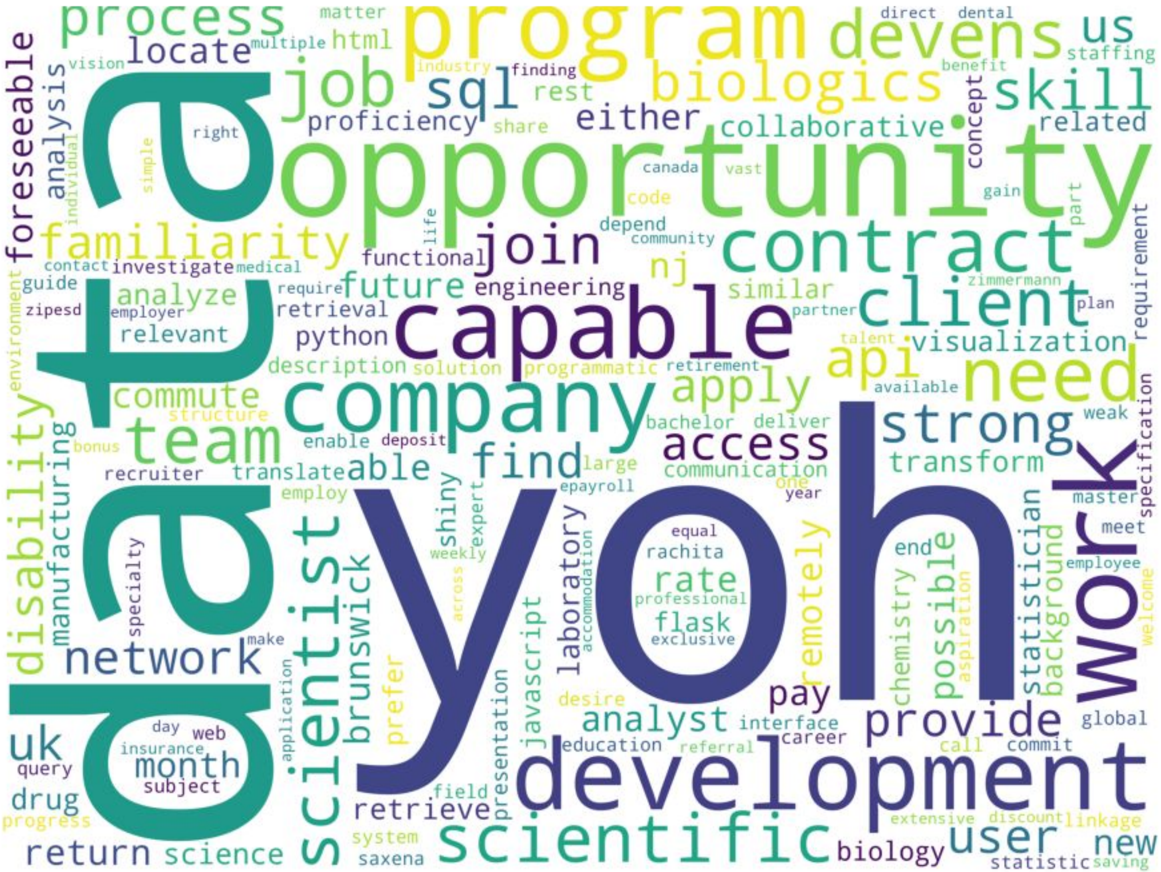
- All data cleaning is handled by the DataProcessor module.
- Tokenization depends on sent_tokenize and word_tokenize from nltk.tokenize.
- Parts of speech tagging depends on pos_tag from nltk.
- Lemmatization depends on WordNetLemmatizer from nltk.stem.wordnet.
- Final cleanup depends on stopwords from nltk.corpus.

Results

- All records were successfully processed.
- Raw text inputs are returned as a list of tokens.
- Before and after word clouds for an example job posting are displayed below.



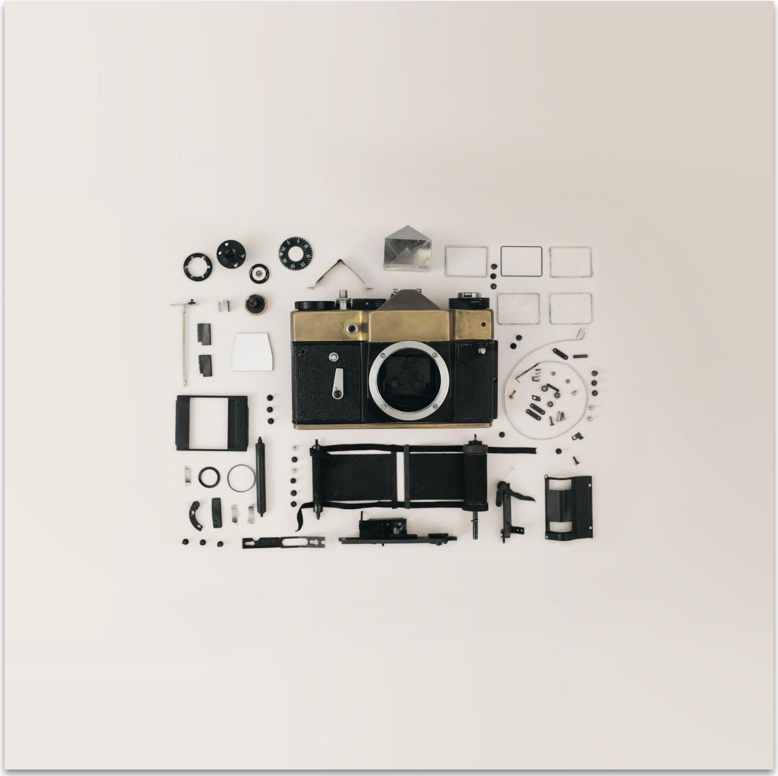
Word cloud for raw job description



Word cloud for clean job description

Feature Engineering

Making the parts



Implementation

- All feature engineering is handled by the DataProcessor module.
- Identifying and combining common phrases depends on Phrases from gensim.models.
- Phrase identification runs on the full corpus.
- The phrase model is specific to our training corpus.

Results

- After feature engineering common phrases are combined into n-grams for n = 1, 2, 3, and 4.
- Adjacent tokens are combined into a single token when they match a common phrase.
    - Ex: Two token input [‘machine’, ‘learning’] is transformed to single token output [‘machine_learning’].
- We display a word cloud for the feature engineered example on the next slide.

data
opportunity
capable
work
devens
team
need
client
job
process
biologics_development
company
familiarity
yoh
skill
yoh_scientific
provide
apply
analyst
contract
locate
work_remotely_foreseeable_future
month_contract_pay_rate
able
scientist_statistician
new_brunswick_nj
possible
chemistry_biology
engineering
investigate
desire
require
guide
meet_career_aspiration
drug
statistic
program
gain
join
simple
retrieve
html_javascript
talent
science
related
functional
access_yoh
make
vast
sql_query
web
user
community
structure
call
us
network
code
shiny
zimmermann_company
either
concept
interface
weekly_epayroll
transform
subject_matter
across_multiple
education_bachelor
similar
year
insurance_direct_deposit
scientific
saxena_welcome
scientist
translate
retirement_saving_plan
rest_api
depend
linkage
commute
deliver
employee_discount_program
end_user
uk_canada
right
zipesd
partner
join_yoh
visualization
return
yoh_find
uk_find
recruiter_rachita
accommodation_application
environment
progress
part
prefer
strong
expert
requirement
available
contact_us_individual_disability
one_large
relevant
extensive
laboratory
yoh_day
share
industry
programmatic
referral_bonus_program
master
specialty
weak
employ
medical_dental
vision_benefit
finding
system
sql
enable
retrieval
life_disability
analysis
solution
professional
background
proficiency_python
flask
description
field
yoh_network
staffing
global
analyze
commit
api
development
specification
manufacturing
access_exclusive
strong_communication_presentation
equal_opportunity_employer
collaborative

Word cloud for feature engineered job description

Modeling

Assembling the tool

Implementation

Topic Model

- A Latent Dirichlet Allocation model with ten topics.
- Takes cleaned and feature engineered job descriptions as input.
- Returns ten dimensional probability vectors representing a mixture of topics.
- The topics from this model are our DataSkills.

This step satisfies Technical Goals 2 and 3

Implementation

Clustering model

- A k-Means model with ten centers.
- Takes DataSkills vectors as input.
- Returns an integer cluster label ranging from 0 to 9,
- Clusters from this model are our DataRoles.

This step satisfies Technical Goal 4

Implementation

Classification Model

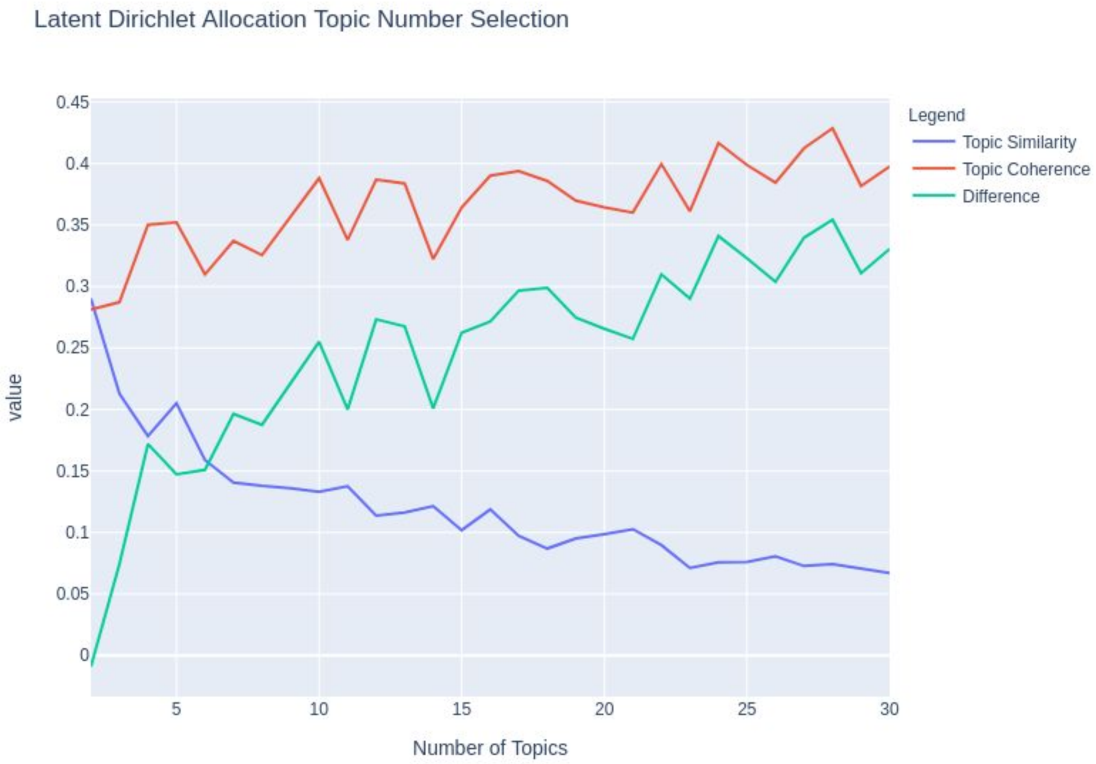
- A Multinomial Naive Bayes' classifier with Lidstone smoothing parameter alpha = 0.1
- Takes word count vectorized job titles as input.
- Returns a DataRole.

This step satisfies Technical Goal 5

Model Tuning

Topic Model

- Hyperparameter
  - Latent Dirichlet Allocation models detect a **number of topics** that must be selected by the user.
- Quality Measures
  - Topic Coherence measures the internal consistency of topics. Higher is better.
  - Topic Similarity measures how much topics overlap. Lower is better.
- Selection Criteria
  - We will select a number of topics where the difference between coherence and similarity transitions to slower growth.

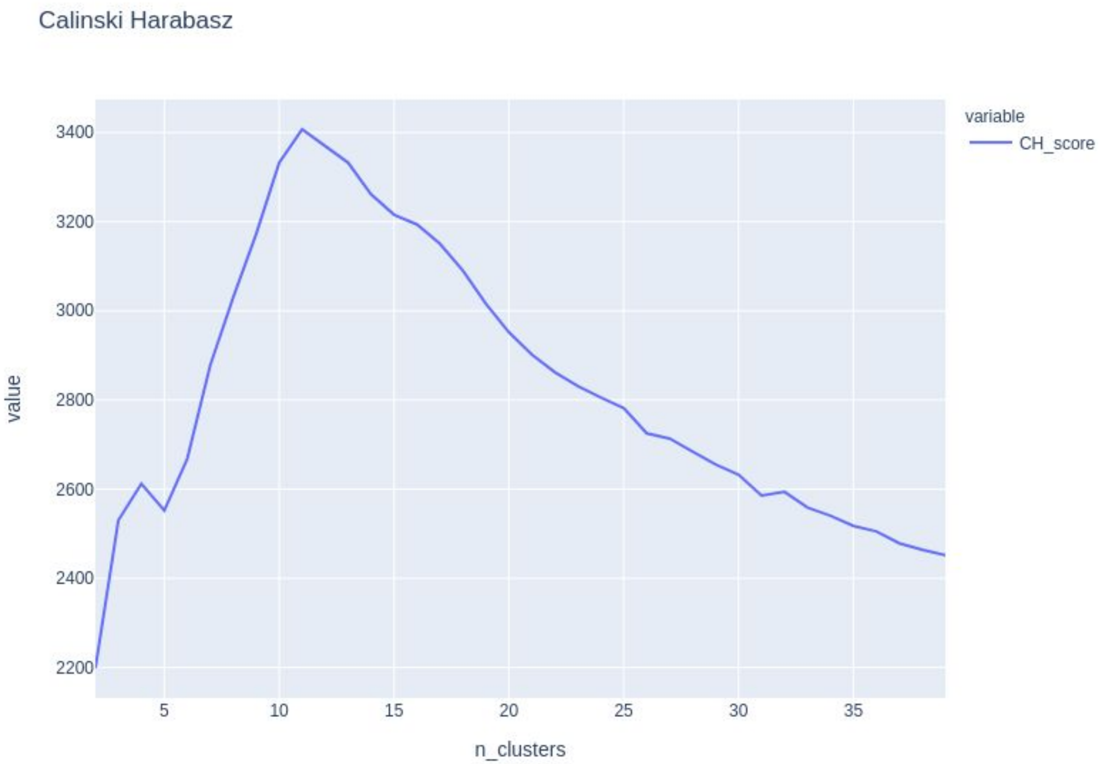


After ten topics the difference between similarity and coherence grows more slowly.

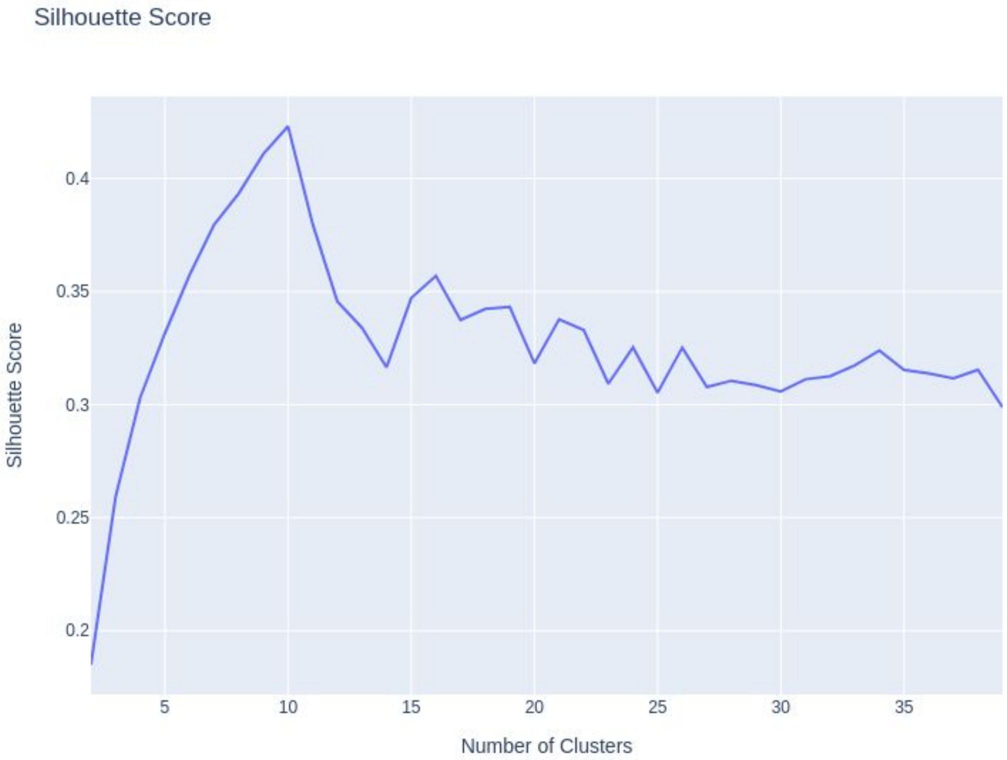
Model Tuning

Cluster Model

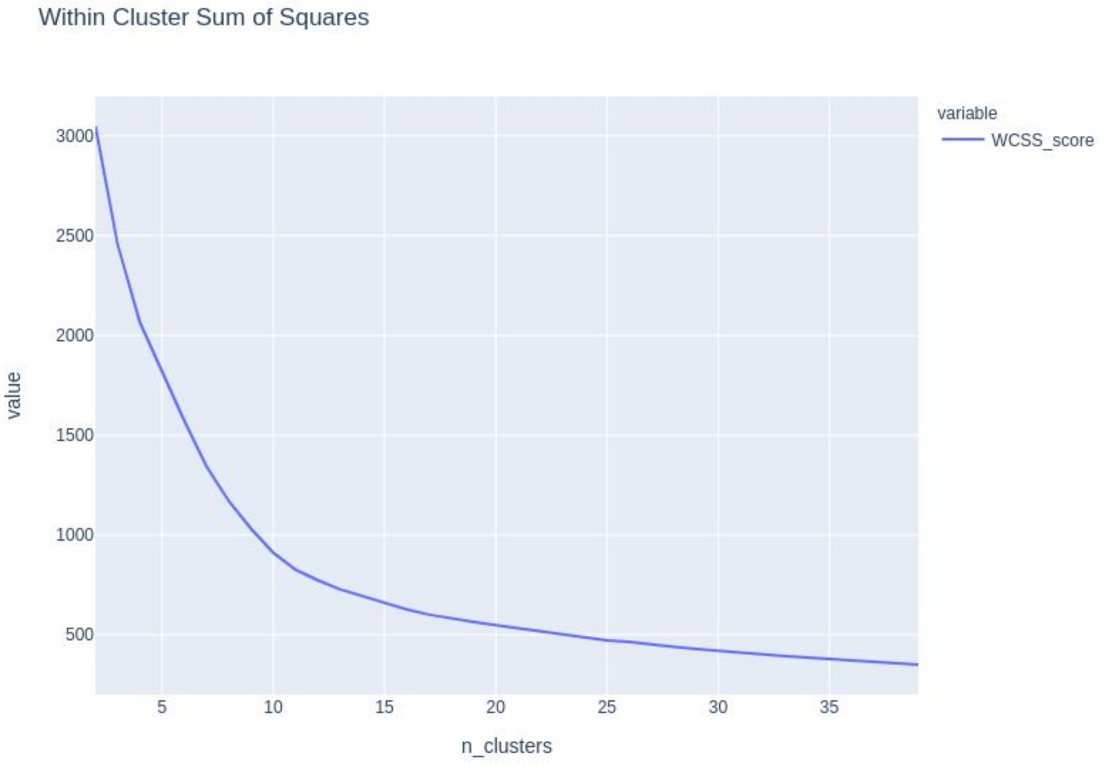
- Hyperparameter
    - k-Means clustering finds a number k of cluster centers that best represent the data. We must select k.
- Quality Measures
    - Calinski-Harabasz Index and Silhouette Score both measure the ratio of within cluster to between cluster dispersion.
    - Within Cluster Sum of Squares measures the within cluster dispersion.
- Selection Criteria
    - We will select a value of k where CHI or SS are maximized or WCSS has an elbow. We prefer smaller k.



The Calinski-Harabasz Index indicates that approximately eleven clusters is optimal.



The Silhouette Score indicates that approximately ten clusters is optimal.

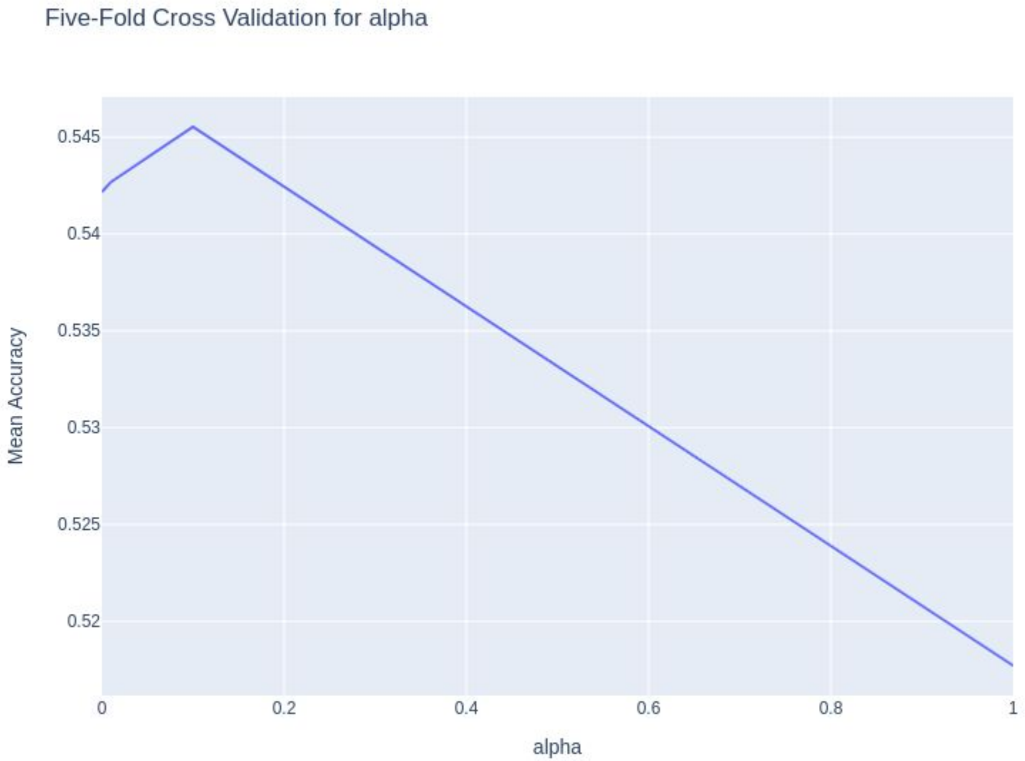


The Within Cluster Sum of Squares has a smooth elbow, between nine and fifteen clusters are optimal.

Model Tuning

Classifier Model

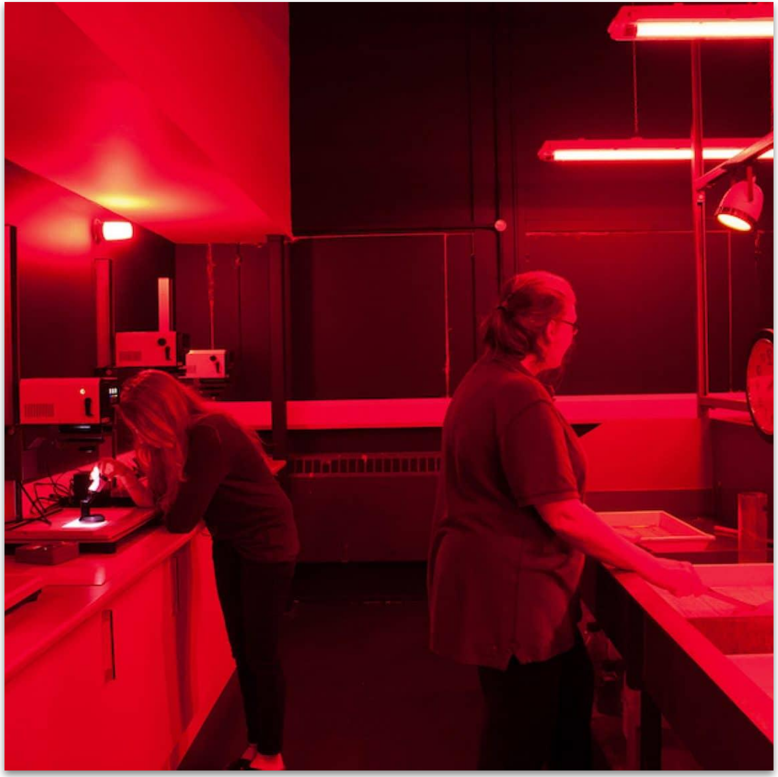
- Hyperparameter
  - Multinomial Naive Bayes' works best with some amount (alpha) of smoothing to avoid issues with probabilities near zero.
- Quality Measures
  - We will evaluate our model in terms of accuracy, so we will also us accuracy for our hyper-parameter tuning.
- Selection Criteria
  - We perform five-fold cross validation and select that value of alpha that produces the best average accuracy.



Setting alpha to 0.1 produces the highest mean accuracy across validation folds.

Evaluation

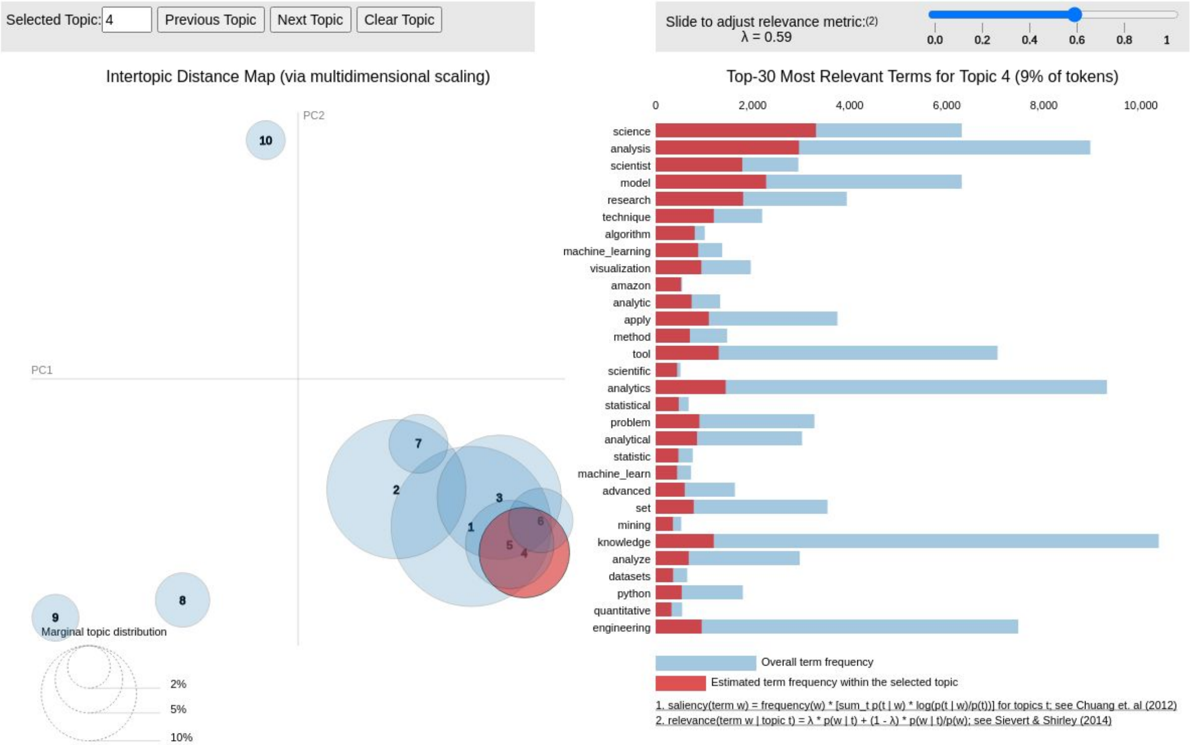
Ensuring the quality of our work



DataSkills

- Analysis
- Process Management
- System Administration
- Data Science
- Big Data
- Technical Leadership
- Compliance
- Company Culture
- Part Time
- Boilerplate

- Each named skill represents a topic from our model
- Names were selected using a visualization tool
- The first eight DataSkills are fairly consistent with intuition.
- The last two are more more requirements than skills.

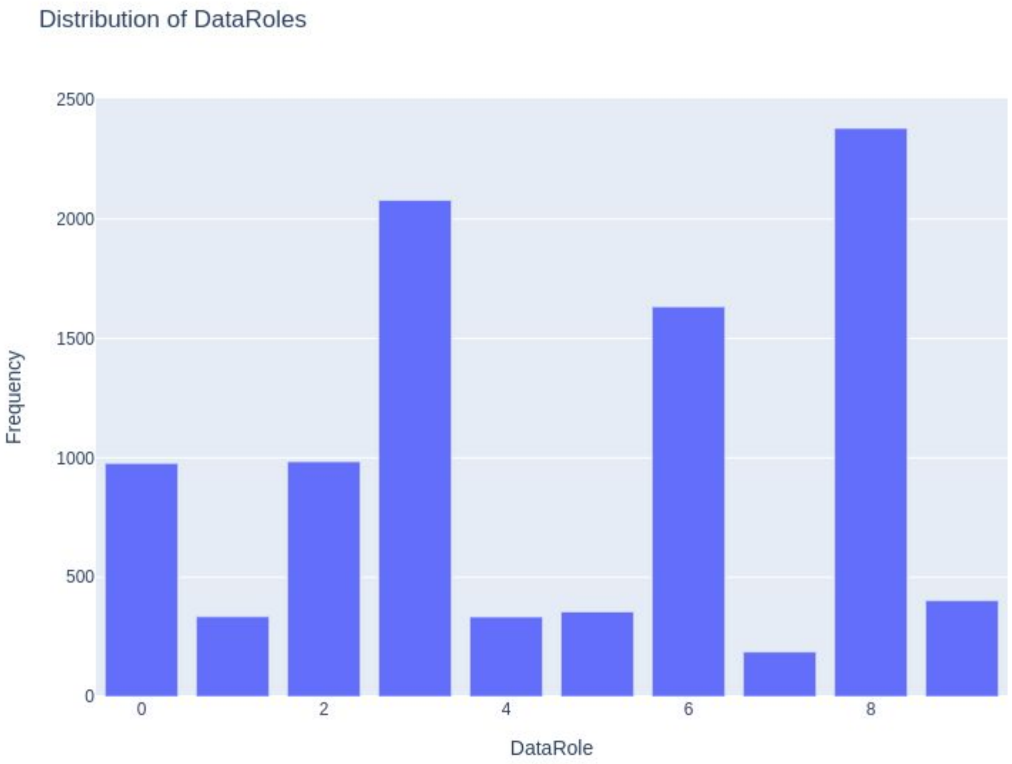


1. saliency(term w) = frequency(w) * [sum_t p(t | w) * log(p(t | w)/p(t))] for topics t; see Chuang et. al (2012)
2. relevance(term w | topic t) = λ * p(w | t) + (1 - λ) * p(w | t)/p(w); see Sievert & Shirley (2014)

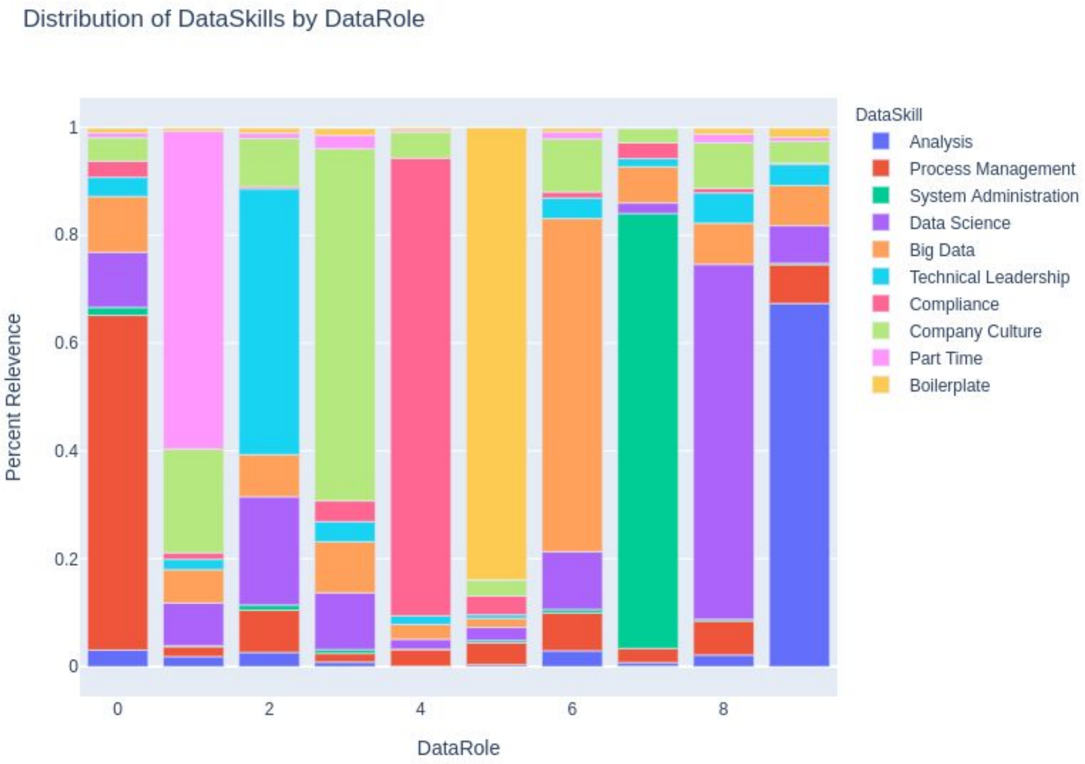
The topic above became our Data Science DataSkill based on the thirty most relevant words above.

DataRoles

- We identified ten DataRoles using our clustering model.
- Each DataRole is a cluster of jobs with similar DataSkills requirements.
- Based on the DataSkills required for each DataRole we were able to identify several familiar roles.
- There were five dominate DataRoles that represented the majority of the data job market.



Classes 0, 2, 3, 6, and 8 make up the majority of DataRoles in the job market.



The figure above shows the mixture of DataSkills required by each DataRole.

DataRoles and DataSkills

Based on the DataSkills requirements we do our best to identify the five most common DataRoles with existing job titles.

- DataRole 0 describes a Data Engineer, incharge of deployment.
- DataRole 2 describes a Lead Data Scientist, managing a data science team.
- DataRole 3 seems to be a role at a company with a strong commitment to company culture.
- DataRole 6 describes a Big Data Engineer, incharge of managing big data.
- DataRole 8 describes a Data Scientist, incharge of machine learning and statistical analysis.

Accuracy of Job Titles

- Our classification model tried to predict the DataRole of a job posting using only the job title.
- This simulates the process of vetting a list of job postings for further review based on the title.
- We found that our best model was able to produce 55% accuracy.
- The model performs substantially better than random guessing, which should have 10% accuracy.
- Correctly predicting the type of job (DataRole) from the title 55% of the time is still not efficient.
    - Incorrectly rejected postings are a missed employment opportunity.
    - Incorrectly accepted postings cost the job seeker additional time that could be better used.

Deployment

Putting our product in your hands

Implementation

Coming Soon

- RESTful API to provide DataSkills and DataRoles evaluation of a raw text job description in Software as a Service model.
- Dashboard to demonstrate on the fly computation of DataSkills and DataRoles.

Recommendations

What should you do?

Job Seekers

- Would benefit from job postings supplemented with a numerical DataSkills summary.
- Would benefit from the ability to filter job postings by DataRole.
- Should avoid vetting data industry jobs by job title alone and this is not sufficiently accurate.

Employers

- Should include DataSkills summaries in job postings to improve relevance of applicants.
- Should use DataRoles to clearly define roles on their team and communicate needs when hiring.
- Should use both DataSkills and DataRoles as tools to guide standardization in job titles.

Job Forums

- Can add value for both employers and job seekers by incorporating DataSkills and DataRoles into job listings.
- Can use both DataRoles and DataSkills as inputs for further analysis of job postings.
- Can improve job market efficiency by increasing the availability and accessibility of information by leveraging DataSkills and DataRoles.

Thank You

I appreciate your attention!

Contact Information

- Website: sethchart.com
- Twitter: @sethchart
- LinkedIn: sethchart
- GitHub: sethchart
- Email: seth.chart@protonmail.com
- Project Repository: DataJobs

Appendix

Just in case you have questions

List of Images

- Photo by Curioso Photography on Unsplash
- https://www.processsensors.com/industries/steel/molten-steel-handling-and-casting-temperature
- Photo by Alexander Andrews on Unsplash
- Photo by Alexander Andrews on Unsplash
- https://www.photoion.co.uk/blog/encyclopaedia/darkroom/
- Photo by Klaudia Piaskowska on Unsplash
- Photo by 🇸🇮 Janko Ferlič on Unsplash