D Mitriy, CC BY-SA 3.0, via Wikimedia Commons

AI-based Model for Closed Job Detection

Mentor:Yusi Zhang, SongTao Guo

Zixuan Li

Software Engineering Intern 21'

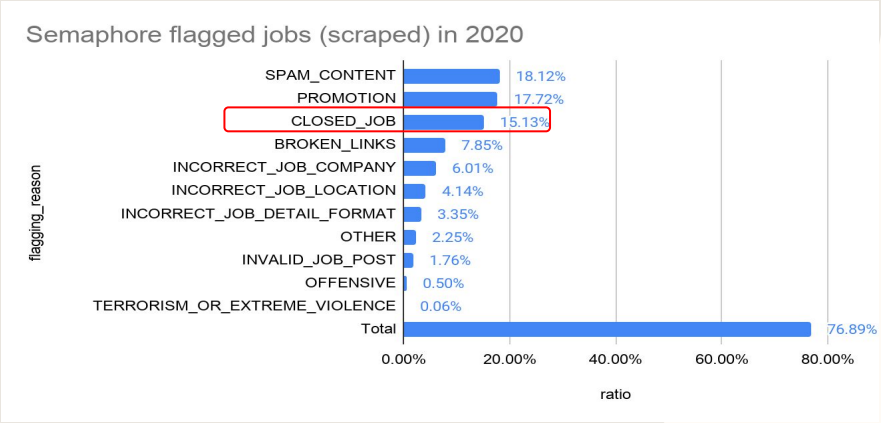
LinkedIn

Agenda

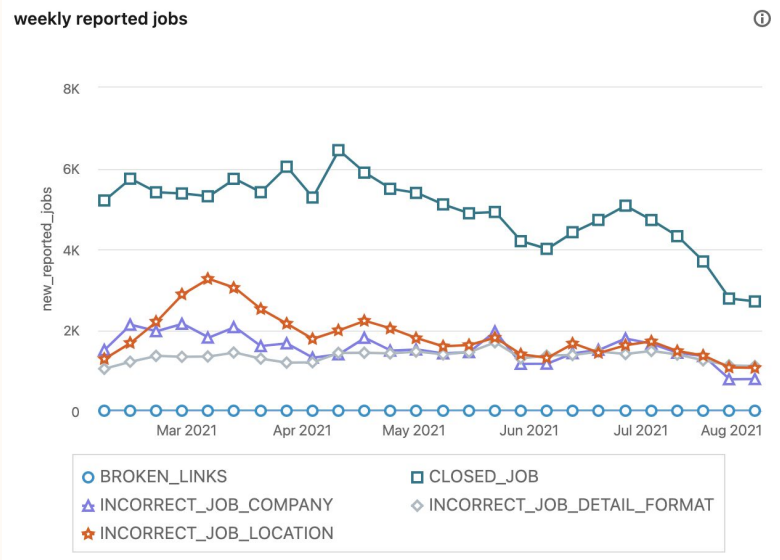
1. Problem Statement
2. Data Preparation & Feature Extraction
3. Modeling & Evaluation
4. Challenges & Takeaways

15%+

Of user complaints in 2020 is contributed by closed jobs



CLOSED_JOB is one of the top job quality issues of the ingested jobs reported by our members.



$$P(s(j) = closed \mid \langle j, p_t, c_t \rangle) = ?$$

$$\Downarrow$$

$$f: R^n \rightarrow \{open(0), closed(1)\}$$

Binary Classification Problem

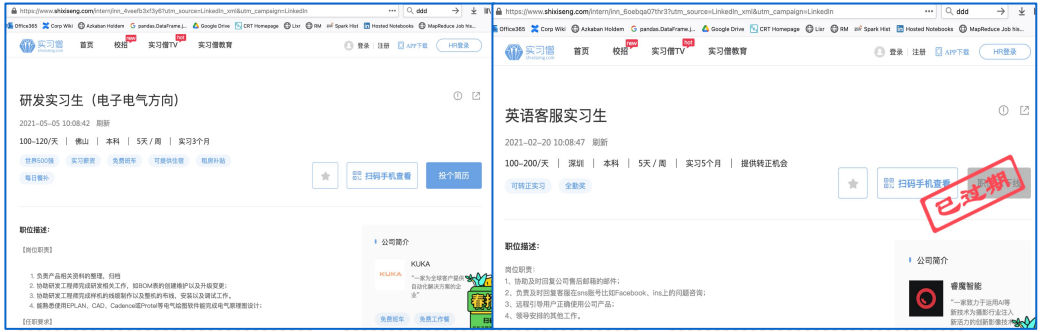
Footnotes

1. j as job from all ingested jobs with external apply links
2. s(j) as binary status of job j
3. p_t as web content of apply page p given time t
4. c_t as context given time t

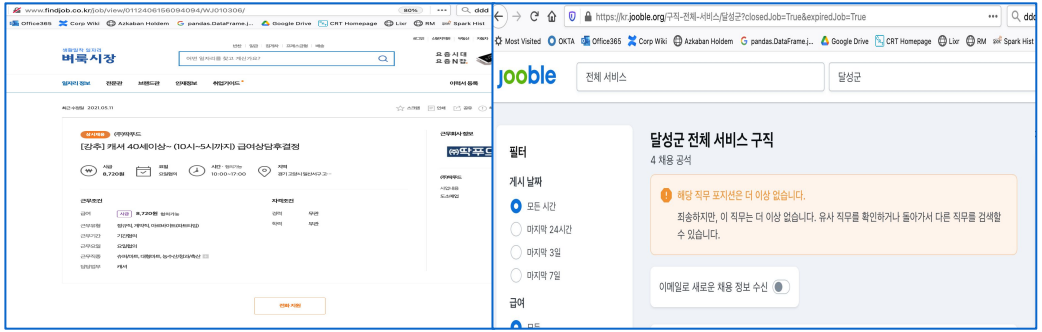
Closed/Open Jobs

examples

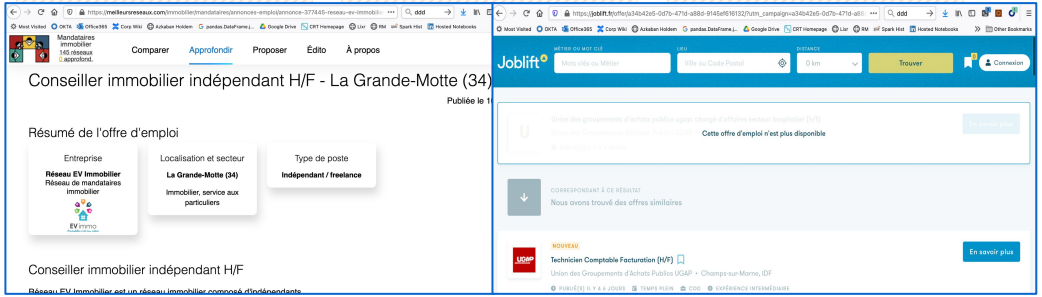
shixiseng.com

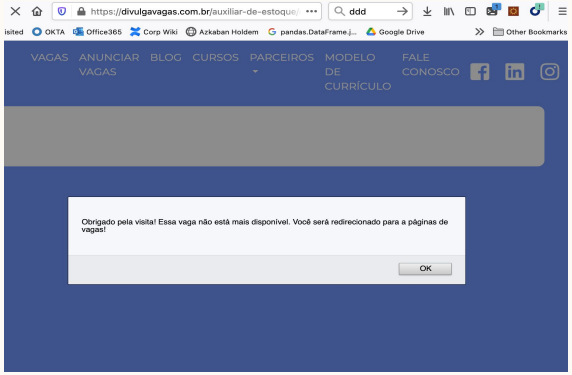


kr.jooble.org

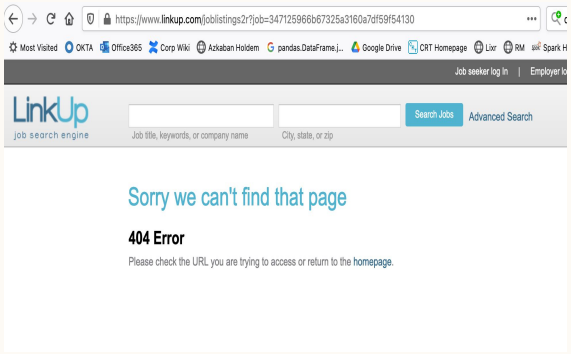


joblift.fr

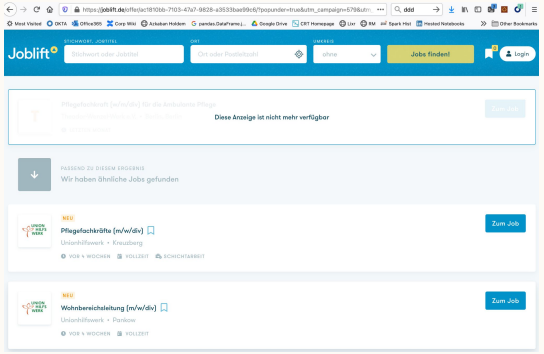




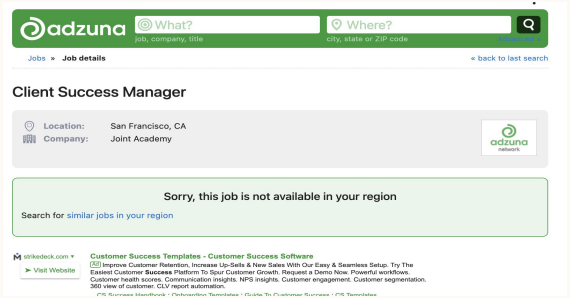
Alert message



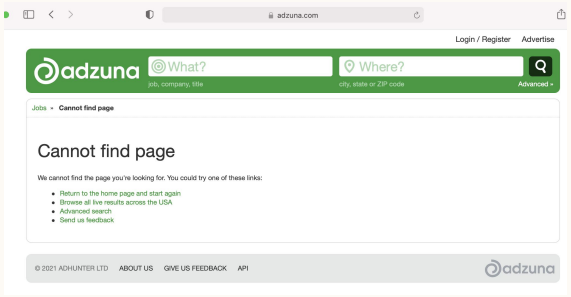
404 error, redirect or not, “cannot find”



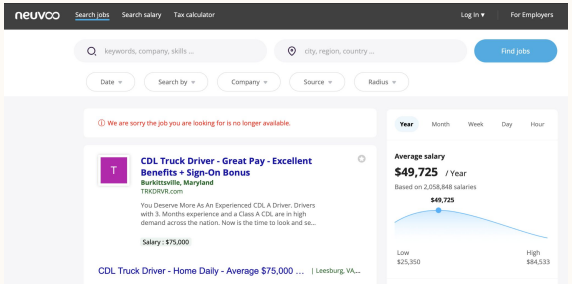
Single job vs. job list, “no longer”



Redirect or not, “not available”



Redirect or not, “cannot find”



Redirect or not, “not available”

Solutions

We should consider an investment of a hybrid approach.

Job title matching

Build a static domain knowledge base and perform job title matching against crawled content.

- Pros: Easy to understand
- Cons:
  - Job quality issues of the static domain knowledge base
  - Job title mismatches due to format difference

RegEx Rule-based

Use manually extracted rules to achieve high precision

- Pros:
  - High precision
  - Easy to understand/deploy directly in the remote scrapers
- Cons:
  - Significant manual effort needed
  - Less adaptive to changes
  - Hard to cover the long tail

AI-based model

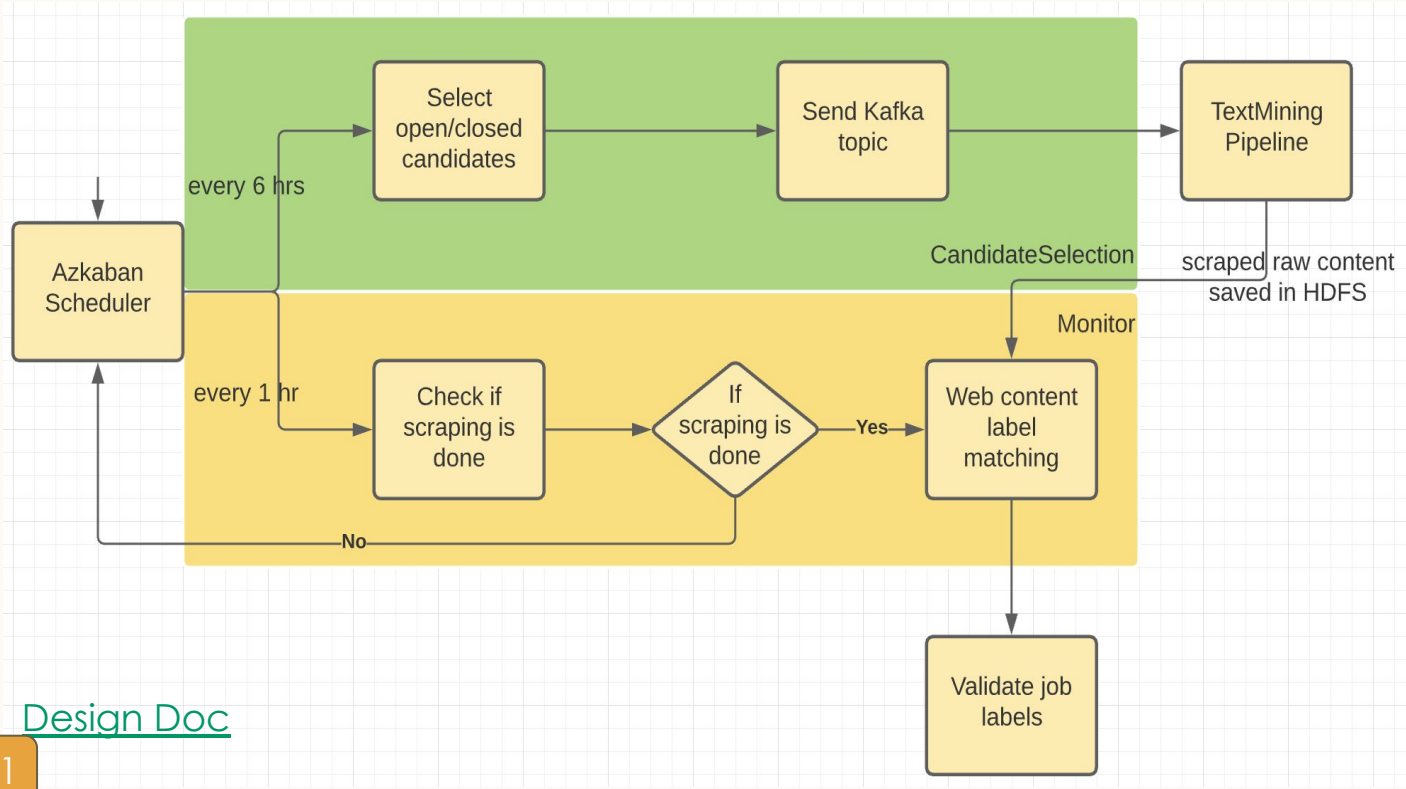
Use AI model to learn hidden rules from the raw data

- Pros:
  - Less manual effort to update and maintain the rules
  - Easy update
- Cons:
  - Hard to achieve high precision
  - Large and precise training dataset needed
  - Deployment constraints

Data Preparation Iteration 1

Rb 2667131

Data Preparation -- Iteration 1



Design Doc

Rb 2667131

Iteration 1 Blocked & Learning from Iteration 1

Why blocked?

Since the design in iteration 1 depends on the Text Mining pipeline to automatically scrape and save raw content in HDFS, it was blocked when the Text Mining pipeline wasn't launched on time.

Experience gained

- Program in Scala, Presto SQL
- General idea on how to select open/closed candidates for training data → **reuse in Iteration 2**
- Basic knowledge on LinkedIn internal tools

Data Preparation Iteration 2

Initial query rb 2675696.

Thank you Songtao for providing detailed feedback and scraped web content.

Data Preparation -- Iteration 2

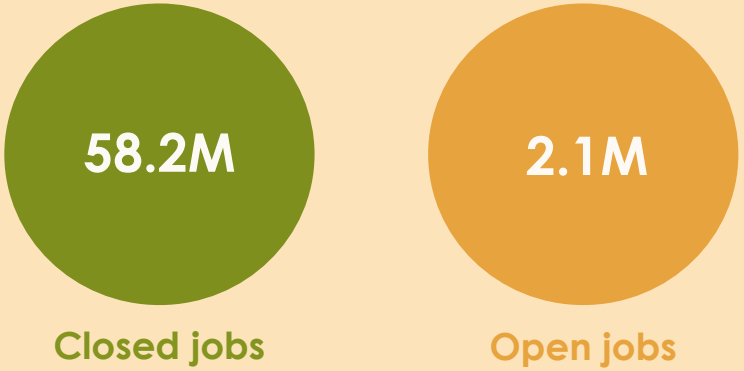
Candidate Selecting Heuristic

Open jobs = non-JA jobs created in last 2 days & never flagged

Closed jobs = jobs closed by job ingestion in last 90-30 days & closed in raw jobs & existed in job posting + flagged closed job in recent 90 days

Additional filters minimize 3m+ noises

How many jobs we select?



How about sampling strategy?

Things to be aware of when sampling

For high coverage and representation, we should ensure...

- Each domain in the training data has enough samples representing both open and closed jobs.
- Acceptable amount of samples from each domain to avoid dominance.
- Cover representative domains.

Sampling methods

Based on proportion

- The amount of samples selected from every domain is based on its contribution to all jobs. Round up if needed.

  Ex. We want 60k open/closed jobs. Suppose domain *D* has *d* jobs and *k* jobs in total, we should select 60k x *d* / *k* for *D*.

- The amount of samples selected from each domain is the same. Calculating performance score based on proportion (weight). Round up in the end if needed.

  Ex. We select 1 job from every domain. We times the weight (*d*/*k*) when calculating score.

From top domains

Total 85,867 domains covered by 27,593,114 non-JA sourced jobs created in the past 1 month

Top domains	Job coverage
1630	**90%**
4357	**95%**
20469	99%

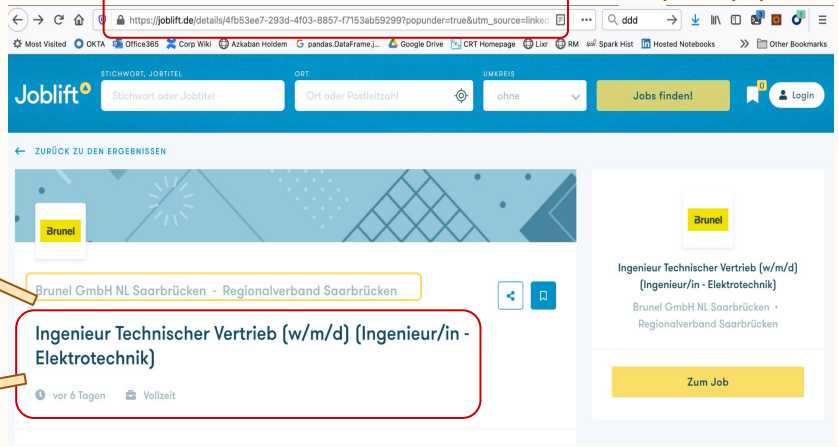
Stats

sample size	pre_label	domains	jobs	Total jobs
Top 2k domains, max 100 samples/domain-label	closed	1,951	189,984	339,720
	open	1,922	149,736	
Top 5k domains, max 100 samples/domain-label	closed	4,210	404,884	598,462
	open	4,062	193,578	
Top 2k domains, max 50 samples/domain-label	closed	1,951	95,615	177,985
	open	1,922	82,370	
Top 5k domains, max 50 samples/domain-label	closed	4,210	205,124	333,241
	open	4,062	128,117	

Sample Size: top **2k** domains, max **100** samples/domain-label

Feature Extraction

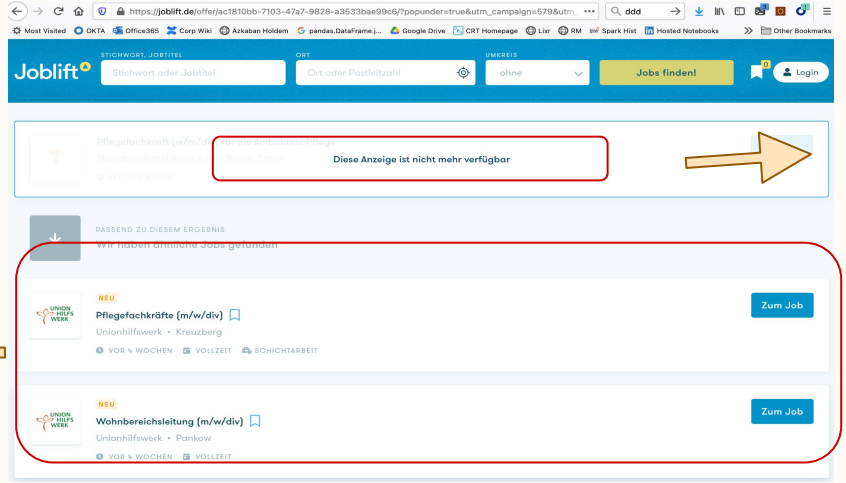
URL query params



Company
Job location

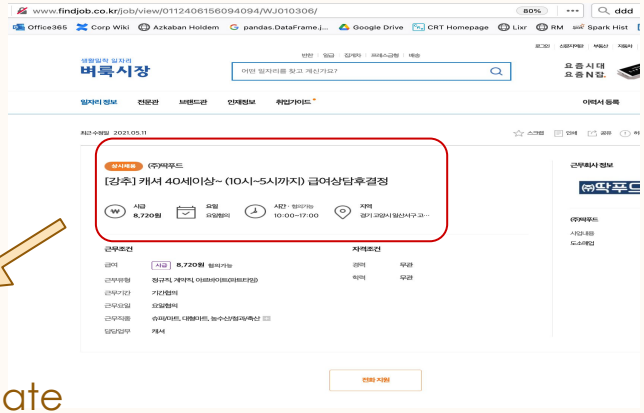
Language

Job title
Job created date
Job type (full/part time)
Job description

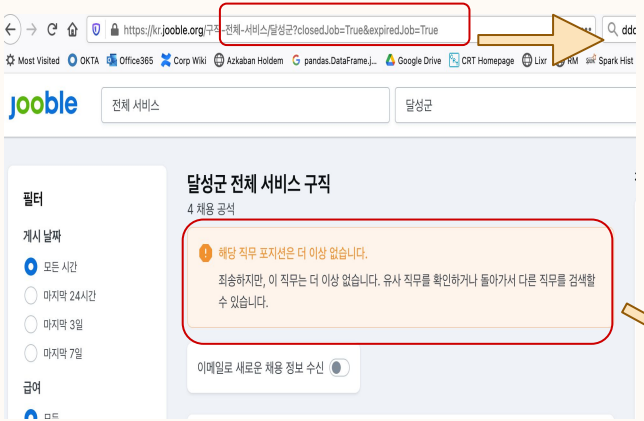


Keywords “no longer”

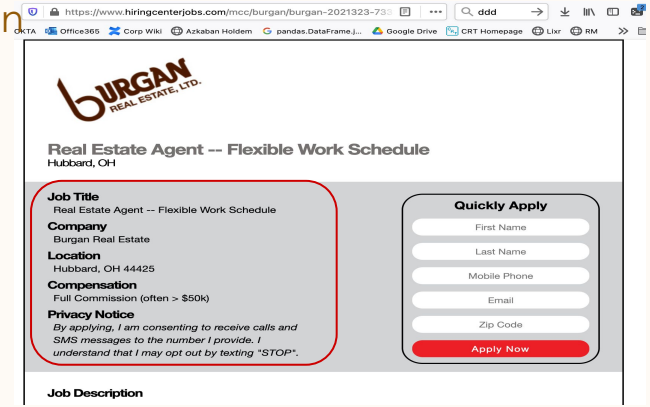
Embedded links

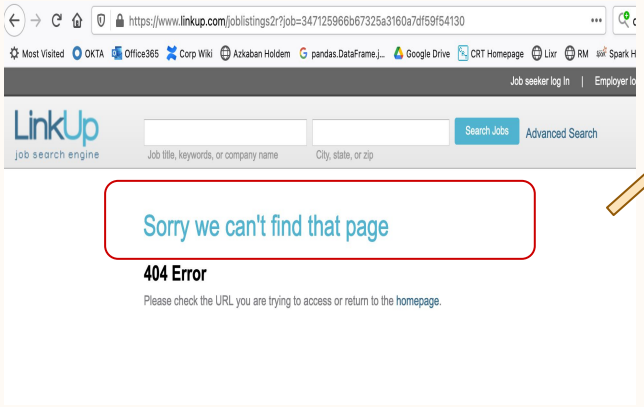


Job title
Job created date
Job type (full/part time)
Job description
Company
Salary
etc.



URL params
expiredJob=True
closedJob=True





Alert message

Open job

Closed job

Potential features we can explore

Web features

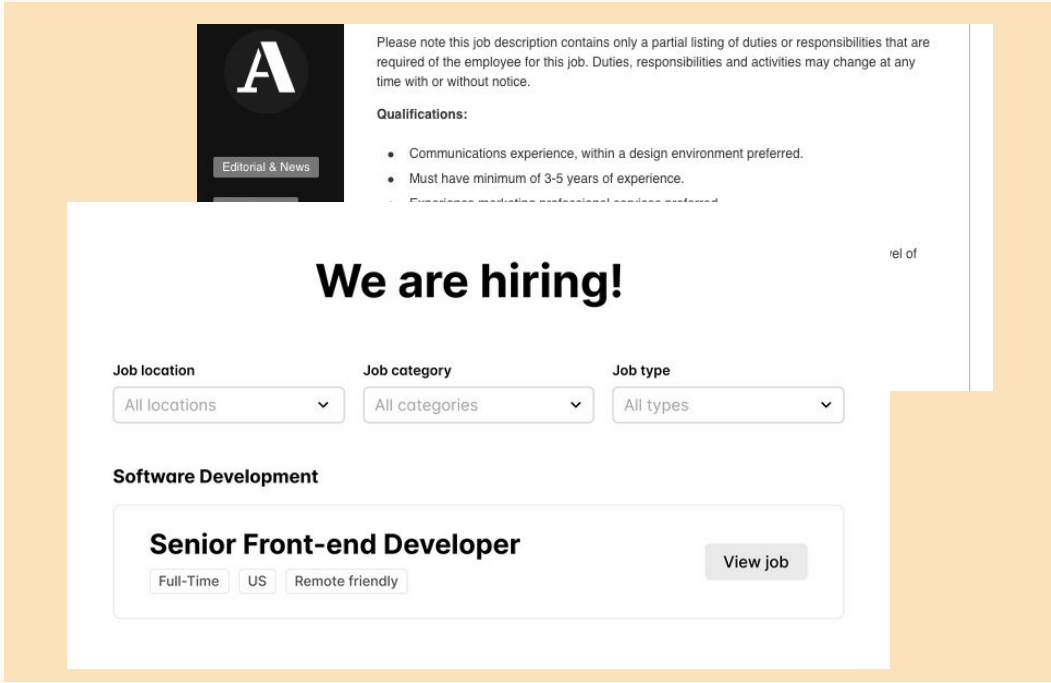
- Seed domain/final URL domain(redirected)
- DOM structures (tag, class names)
- URL query parameters (keywords)

Job-web matching features

- Job title/description/location match

Job features

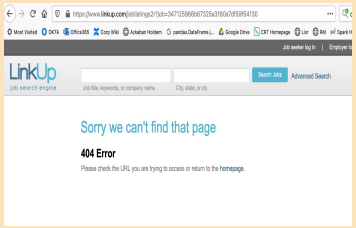
- Job engagement (clicks, applies, favs)
- Job created date



Features chosen for modeling

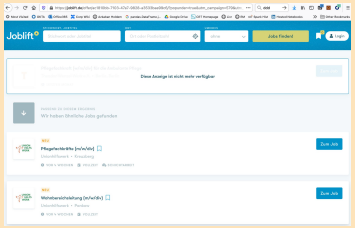
1. Length of raw html

Some closed job might only have a message saying something similar to "this job is already closed" while open jobs usually include detailed job description, responsibility, company introduction etc and thus longer.



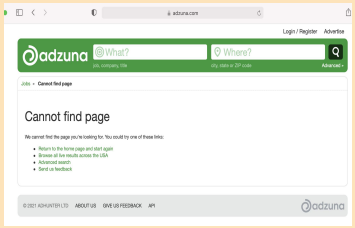
2. Totals of links

Some domains suggest other opportunities to the job seekers if the job they are currently looking at is closed. Those domains usually include a list of embedded links to other jobs.



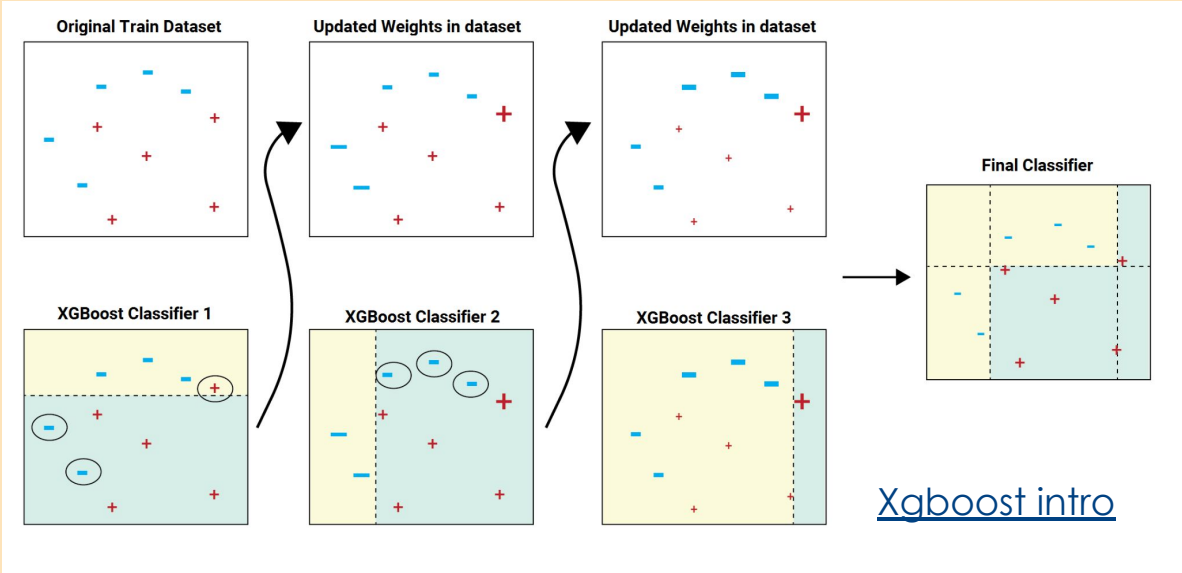
3. Job title matching

If the job title doesn't match, this job, with high probability, is closed.



Modeling & Evaluation

DEMO notebook



Xgboost intro

70.5%	Accuracy	78.2%
66%	Precision	78.3%
81%	Recall	78.5%

Before: This represents the performance of xgboost model with 2 features w/o tunings.

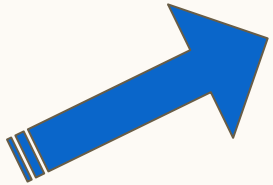
After: This represents the performance of xgboost model as a desired outcome of tuning.

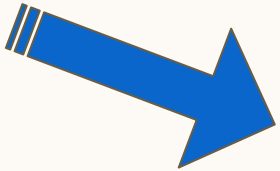
For who to take it over,

1. Manage a clean training dataset
2. Increase sample size + Tuning
3. Review failure cases & enrich features
4. Transport to PCV2

Challenges & Takeaways

- **Everything is new.** New programming language. New tools. New knowledge.
- **Limited related background**. Learn everything by myself from the very beginning.
- **Dependency challenge**. Depending pipeline wasn't launched on time. Had to switch to another approach.





- **Technological takeaways**
  - Basic knowledge on ML modeling
  - Code/Debug in Scala, SQL, Python
  - Basic understanding on internal tools including PCV2, Frame, avro-schema, Azkaban, etc.
- **Non-technological takeaways**
  - Be chill when blocked and quickly proceed to solve problem actively
  - Analyze problems independently and support my statements with solid stats
  - Ask questions. Reach out to experienced engineers for suggestions professionally

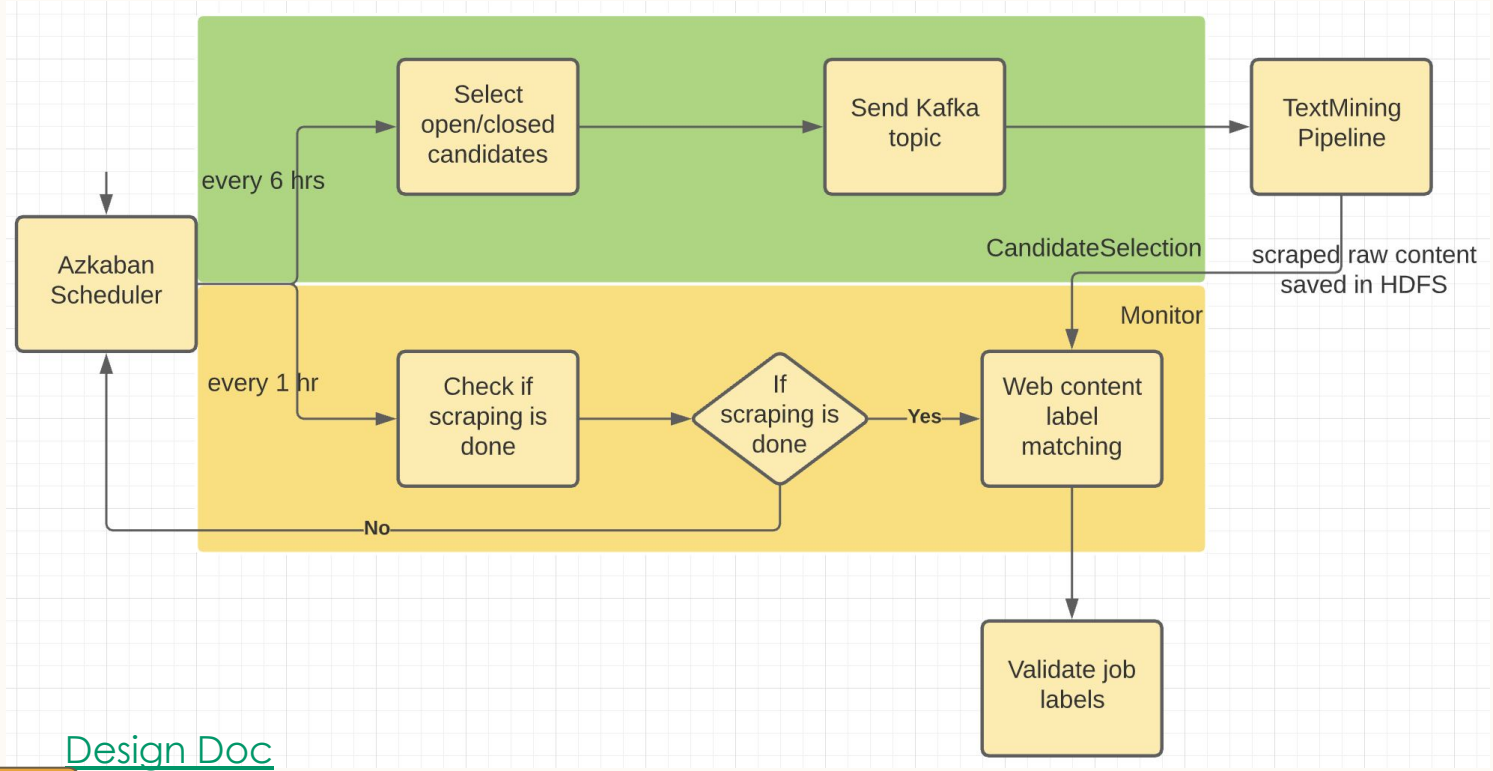
Q & A

Special thanks to Songtao and Yusi

Thank you!

Appendix

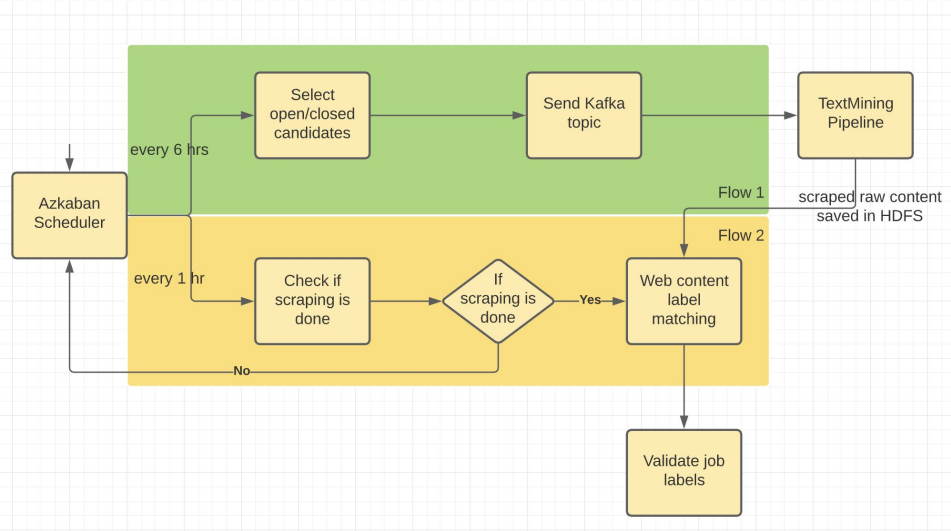
Data Preparation -- Iteration 1



Design Doc

Rb 2667131

Data Preparation -- Iteration 1



Flow 1

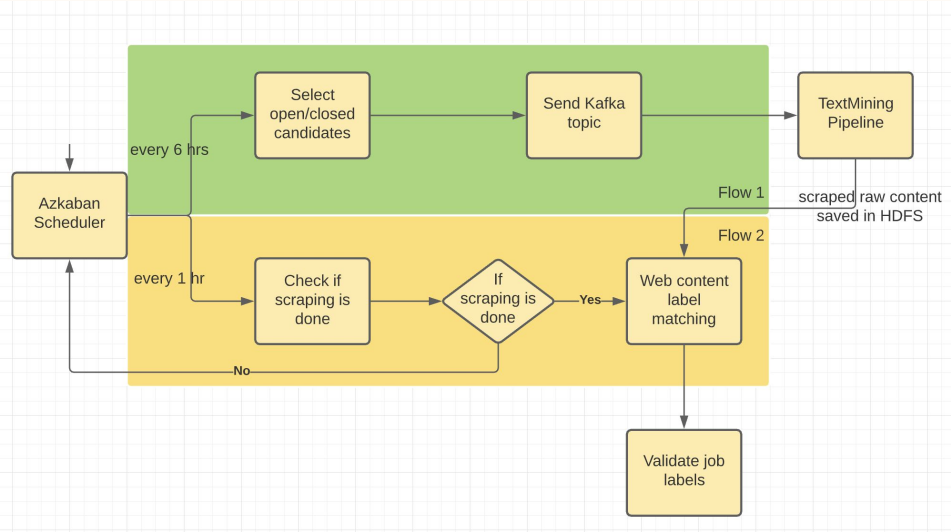
- Input: source, raw_jobs, jobs_column_prod, user_flagging_union, datepartition, sample size
- Output:
  - Save job_id, url and label in HDFS with batch_id as folder name
  - Send a TextMiningRequestMessage for every selected candidate

What happens next

1. Select urls(urls is labeled) -> send to Kafka topic1
2. Ubiquity-backend consumes Kafka messages, and then schedules the scraping which is done by Scraper system
3. Scraper service sends scraped web content to Kafka topic2
4. Messages in Kafka topic2 will be materialized to HDFS automatically and further processed by flow 2

Rb 2667131

Data Preparation -- Iteration 1



Flow 2

- Input: workflow dir of flow1 and flow2, TEXT_MININGJobIngestionSourceMessageV2
- Output: raw_content with labels, job_id, url, datapartition registered in a partition table

Now we should have raw html paired with its open/closed label and should be ready to proceed to feature extraction for modeling.

Rb 2667131