Contents

1 work environment

1.1 yas

1.1.1 #+name:

1.2 org babel

1.2.1 ob : print tables

1.2.2 ob : imports at the top

1. use noweb ?
2. **TODO** preamble

1.2.3 TODO tag subtrees to trigger different action on eval / tangling

1. tag filter view / generated file/buffer

1.2.4 change prefix key

1.2.5 TODO go to discussion on mailing list / stack exchange

1.2.6 code fonts

1. scimax

1.2.7 org database

1.3 TODO python kernel interaction

1.3.1 TODO python debugging

1.3.2 TODO ob-ipython / jupyter kernel

1. add scimax extensions `https://github.com/jkitchin/scimax/blob/master/scimax-org-babel-ipython.el`
2. access running kernels
3. find a way to make jupyter run on python3
   (a) for now, using ipython3 console in ob-ipython
   (b) add something to :session
   (c) add :kernel
   (d) run jupyter in different envs then do :
      i. start in a shell jupyter-console
      ii. copy the json filename from /run/user/1000/jupyter to :session argument in the source code header.

1.3.3 TODO set env

1. **TODO** choose for each session
   (a) run jupyter kernels in different envs
2. **DONE** manage different environments with envwrapper

1.4 Scimax test / merge

1.4.1 run scimax as standalone

1.4.2 implement whole scimax and remove bit by bit what is buggy

1.4.3 implement scimax piece by piece

2 Work directions

2.1 Visualization

2.2 Big Data NLP

2.2.1 Clean data

2.2.2 spark / hive

2.2.3 Clustering

2.3 TODO store, browse and classify offers from different time/origin

2.3.1 TODO scraper pipeline

1. store in a database and gain acces to it
2. check date before crawling page
3. test matcher on title before crawling page
4. if match but different query/website add to "original query/website" list
5. new variables
   (a) date of scrap
   (b) post dates (if different matches / reposts)
      i. new / repost
      ii. most recent date
   (c) original website
   (d) queries
   (e) read / unread

2.3.2 Matcher program

1. run benchmark test to compare speed of different programs
   (a) pandas dataframes
   (b) sql
   (c) no sql
   (d) c
2. comparison methods
   (a) similarity rate
   (b) hash tables / id
3. matching criterium
   (a) is date & firm & title
   (b) title is not too common
   (c) run tests

2.3.3 news alert system

1. filter : rating value of a job
   (a) criterium
      i. short term
         A. contract
         B. salary
         C. domain
         D. location
      ii. long term
         A. career
         B. knowledge
   (b) find data from other sources
   (c) concentrate on available data for now
2. UX
   (a) overview
      i. number of new offers

            ii. print titles
    (b) browse offers
    (c) rate
    (d) features valuation
    (e) keywords valuation
        i. banned
        ii. needed
        iii. quckly give a weight to each word

3. Process
    (a) Add new interesting offer to a queue
    (b) News / RSS / Mail model ?
4. run as a daemon on a server
5. send sms with a link
6. generate html ?

2.4 **TODO** browse offers

2.4.1 sql generated table

2.4.2 generated on view by a server ?

`http://kitchingroup.cheme.cmu.edu/blog/2017/01/03/Find-stuff-in-org-mode-anywhere/`

2.4.3 show in a browser

1. java app ?

2.4.4 browse on by one

2.4.5 export to a file / hyperlinked filesystem

1. pdf one job a page
2. Custom column/agenda view ?
3. org file ?
    (a) **TODO** generate html

2.4.6 emacs mode / standalone lightweight emacs distribution

2.4.7 index page with redirect links to original offers ?

2.5 Environment

2.6 SQL database

2.7 scrapper

2.8 frontend

3 TODO set PATH env