Amazon SageMaker Studio File Edit View Run Kernel Git Tabs Settings Help Feedback

/ cskg / ESWC2021 /

Name	Last Modified
Analyze CSKG.ipynb	7 minutes ago
CSKG Embeddings.ipynb	12 hours ago

j_CSKG.ipynb | Analyze CSKG.ipynb | CSKG Embeddings.ipynb

Markdown git 2 vCPU + 4 GiB Python 3 (Data Science) Share

Analyzing CSKG

This notebook performs various analyses on CSKG

Parameters are set up in the first cell so that we can run this notebook in batch mode.

Parameters for invoking the notebook

- `cskg_path` : a folder containing the CSKG edges file and all the analysis products.
- `kg` : the name of the edge file.
- `nkg` : the name of the normalized edge file.
- `delete_database` : whether to delete the SQL database before running the notebook: "" or "no" means don't delete it.

Tip:

1. We have prepared the `kg` and `nkg`, you can download them from https://drive.google.com/drive/u/1/folders/16347KHSloJJZIbgC9V5gH7_pRx0CzjPQ.
2. if you don't want to the default setting, please make sure that you type `jupyter notebook` under our root floder(the path of the whole repository), which is `../cskg`

Preamble

Set up paths and environment variables, make sure those varible are pointing to the correct file(path)

[2]:
```
# Parameters
cskg_path = "../output"
kg = "cskg.tsv.gz" #"cskg_connected.kgtk.gz"
nkg = "cskg_connected_normalized.tsv.gz" #"cskg-normalized.kgtk.gz"
delete_database = "yes"
```

[3]:
```
import io
import os
import subprocess
import sys

import numpy as np
import pandas as pd

import altair as alt
```

[4]:
```
os.environ['CSKG'] = cskg_path
os.environ['KG'] = "{}/{}".format(cskg_path, kg)
os.environ['NKG'] = "{}/{}".format(cskg_path, nkg)
os.environ['STORE'] = "{}/wikidata.sqlite3.db".format(cskg_path)
os.environ['kypher'] = "kgtk query --graph-cache " + os.environ['STORE']
# os.environ['kypher'] = "time kgtk --debug query --graph-cache " + os.environ['STORE']
```

[5]:
```
!echo $CSKG
!echo $KG
!echo $NKG
!echo $STORE
!echo $kypher
```

```
../output
../output/cskg.tsv.gz
../output/cskg_connected_normalized.tsv.gz
../output/wikidata.sqlite3.db
kgtk query --graph-cache ../output/wikidata.sqlite3.db
```

[6]:
```
cd $cskg_path
```

```
/root/cskg/output
```

[7]:
```
if os.path.exists(delete_database) and delete_database != "no":
    print("Deleted database")
    !rm $STORE
else:
    print('No database exists')
```

```
No database exists
```

Utilities

[8]:
```
def bar_chart(data, x_column, y_column, title="", width=800):
    """Construct a simple bar chart with two properties"""
    bars = alt.Chart(data).mark_bar().encode(
        y=alt.Y(y_column, sort='-x'),
        x=x_column
    ).properties(
        title=title,
        width=width
    )

    text = bars.mark_text(
        align='left',
        baseline='middle',
        dx=3  # Nudges text to right so it doesn't appear on top of the bar
    ).encode(
        text=x_column
    )

    return (bars + text)
```

[9]:
```
import io
import pandas
import subprocess

def shell_df(command, shell=False, **kwargs):
    """
    Takes a shell command as a string and and reads the result into a Pandas DataFrame.

    Additional keyword arguments are passed through to pandas.read_csv.

    :param command: a shell command that returns tabular data
```

```
    :type command: str
    :param shell: passed to subprocess.Popen
    :type shell: bool

    :return: a pandas dataframe
    :rtype: :class:`pandas.dataframe`
    """
    proc = subprocess.Popen(command,
                            shell=shell,
                            stdout=subprocess.PIPE,
                            stderr=subprocess.PIPE)
    output, error = proc.communicate()

    if proc.returncode == 0:
        if error:
            print(error.decode())
        with io.StringIO(output.decode()) as buffer:
            return pandas.read_csv(buffer, **kwargs)
    else:
        message = ("Shell command returned non-zero exit status: {0}\n\n"
                   "Command was:\n{1}\n\n"
                   "Standard error was:\n{2}")
        raise IOError(message.format(proc.returncode, command, error.decode()))
```

Poking around

Print some lines to see what we have

[10]:
```
!zcat < "$KG" | head | column -t -s $'\t'
```

```
/bin/sh: 1: column: not found
head: write error: Broken pipe

gzip: stdout: Broken pipe
```

Normalize the file so that it is easier to process with Kypher

[11]:
```
!kgtk normalize --verbose -i $KG -o $CSKG/temp.cskg.normalize.1.kgtk.gz --columns-to-lower 'relation;dimension' source sentence 'node1;label' 'r
```

```
Opening the input file: ../output/cskg.tsv.gz
KgtkReader: File_path.suffix: .gz
KgtkReader: reading gzip ../output/cskg.tsv.gz
header: id      node1   relation        node2   node1;label     node2;label     relation;label  relation;dimension      source  sentence
input format: kgtk
KgtkReader: Special columns: node1=1 label=2 node2=3 id=0
KgtkReader: Reading an edge file.
Node1 column name: node1
Label column name: relation
Node2 column name: node2
Id column name: id
The following columns will be lowered or normalized
 node1;label from node1 (label 'label')
 node2;label from node2 (label 'label')
 relation;label from relation (label 'label')
 relation;dimension from relation (label 'dimension')
 source from id (label 'source')
 sentence from id (label 'sentence')
The output columns are: id node1 relation node2
Opening the output file: ../output/temp.cskg.normalize.1.kgtk.gz
File_path.suffix: .gz
KgtkWriter: writing gzip ../output/temp.cskg.normalize.1.kgtk.gz
header: id      node1   relation        node2
Read 6001531 rows, wrote 15117009 rows with 9115478 labels.
```

[12]:
```
!zcat < $CSKG/temp.cskg.normalize.1.kgtk.gz | head | column -t -s $'\t'
```

```
/bin/sh: 1: column: not found
head: write error: Broken pipe

gzip: stdout: Broken pipe
```

Rename the columns to the standard `node1/label/node2` and add ids

[13]:
```
!kgtk rename-columns --mode NONE -i $CSKG/temp.cskg.normalize.1.kgtk.gz --output-columns id node1 label node2 \
/ add-id --id-style node1-label-node2 -o $NKG
```

Count the number of edges and nodes

[14]:
```
!$kypher -i $NKG \
--match '(n1)-[e]->(n2)' \
--return 'count(e) as num_edges, count(distinct n1) as num_nodes, count(distinct e.label) as num_relations, count(distinct n2) as num_values' \
| column -t -s $'\t'
```

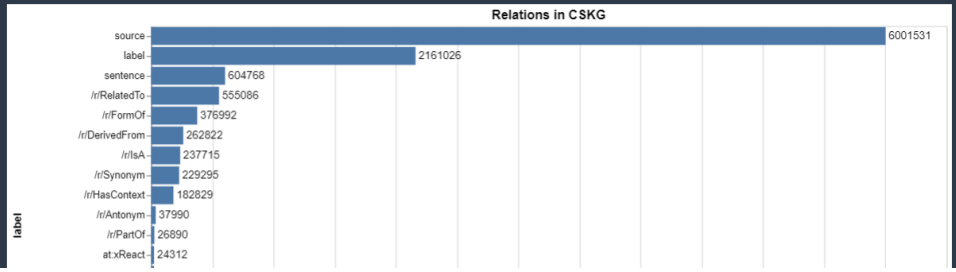
```
/bin/sh: 1: column: not found
[Errno 32] Broken pipe

Exception ignored in: <_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>
BrokenPipeError: [Errno 32] Broken pipe
```

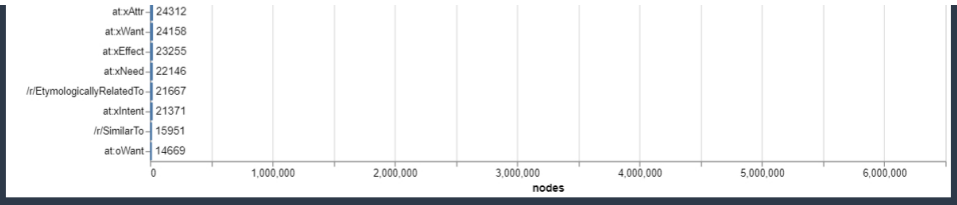
Some Statistics ¶

[15]:
```
command = "$kypher -i $NKG \
--match '(n1)-[e]->(n2)' \
--return 'distinct e.label, count(distinct n1) as nodes' \
--order-by 'count(distinct n1) desc'"
stats = shell_df(command, shell=True, sep='\t')
```

[16]:
```
bar_chart(stats[:20], 'nodes', 'label', title="Relations in CSKG")
```

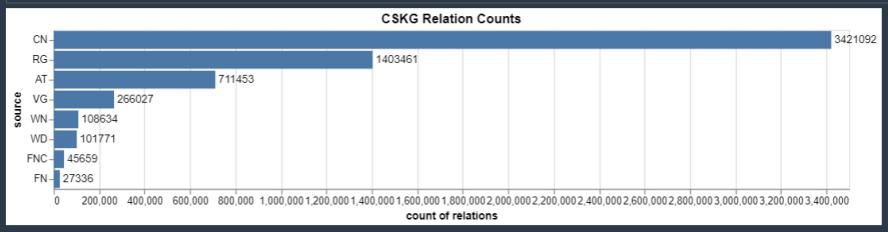
[16]:





Distribution of edges in each data source

[17]:
```
command = "$kypher -i $NKG \
--match '(n1)-[r]->(n2), (r)-[l:source]->(s)' \
--return 's as source, count(distinct r) as `count of relations`' \
--order-by 'count(distinct r) desc'"
data = shell_df(command, shell=True, sep='\t')
bar_chart(data, 'count of relations', 'source', title="CSKG Relation Counts")
```

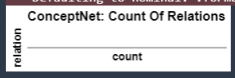
[17]:


Compute the distribuiton of relations in each data source

[18]:
```
command = "$kypher -i $NKG \
--match '(n1)-[r {label: label}]->(n2), (r)-[:source]->(source:`\"SOURCE\"`)' \
--return 'label as relation, count(distinct n1) as count' \
--order-by 'count(distinct n1) desc'"
datasets = []
for source in ["CN", "WN", "AT", "VG", "FN", "WD", "RG"]:
    data = shell_df(command.replace("SOURCE", source), shell=True, sep='\t')
    datasets.append(data)
```

[19]:
```
bar_chart(datasets[0], 'count', 'relation', title="ConceptNet: Count Of Relations", width=200)
```

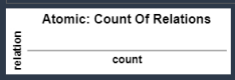
```
/opt/conda/lib/python3.7/site-packages/altair/utils/core.py:187: UserWarning: I don't know how to infer vegalite type from 'empty'.  Defaulting to nominal.
  "Defaulting to nominal.".format(typ)
```

[19]:


[20]:
```
bar_chart(datasets[1], 'count', 'relation', title="WordNet: Count Of Relations", width=200)
```

[20]:

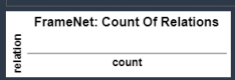

[21]:
```
bar_chart(datasets[2], 'count', 'relation', title="Atomic: Count Of Relations", width=200)
```

[21]:


[22]:
```
bar_chart(datasets[3], 'count', 'relation', title="Visual Genome: Count Of Relations", width=200)
```

[22]:


[23]:
```
bar_chart(datasets[4], 'count', 'relation', title= "FrameNet: Count Of Relations", width=200)
```

[23]:


[24]:
```
bar_chart(datasets[5], 'count', 'relation', "Wikidata: Count Of Relations", width=200)
```

[24]:


[25]:
```
bar_chart(datasets[6], 'count', 'relation', title="Roget: Count Of Relations", width=200)
```

[25]:


ConceptNet nodes that contain `catch` and `throw`

[26]:
```
catch = !$kypher -i $NKG \
--match '(n1)-[r {label: label}]->(n2), (r)-[:source]->(source)' \
--where 'source = $s and n1 =~ ".*/catch/.*"' \
--return 'distinct n1 as node1' \
--spara s='CN' \
--limit 100

for n in catch[1:-1]:
    print("https://www.conceptnet.io"+n)
```

[27]:
```
throw = !$kypher -i $NKG \
--match '(n1)-[r {label: label}]->(n2), (r)-[:source]->(source)' \
--where 'source in [$cn] and n1 =~ ".*/throw?\\b.*"' \
--return 'distinct n1 as node1' \
--spara cn='CN' --spara wd='WD' --spara vg='VG'\
--limit 100
```

```
for n in throw[1:-1]:
    print("https://www.conceptnet.io"+n)
```

ConceptNet nodes that contain `dog`

[28]:
```
dogs = !$kypher -i $NKG \
--match '(n1)-[r {label: label}]->(n2), (r)-[:source]->(source)' \
--where 'source in [$cn] and n1 =~ ".*/dogs?\\b.*"' \
--return 'distinct n1 as node1' \
--spara cn='CN' \
--limit 100

for n in dogs[1:-1]:
    print("https://www.conceptnet.io"+n)
```

[29]:
```
frisbee = !$kypher -i $NKG \
--match '(n1)-[r {label: label}]->(n2), (r)-[:source]->(source)' \
--where 'source in [$cn] and n1 =~ ".*/frisbees?\\b.*"' \
--return 'distinct n1 as node1' \
--spara cn='CN' --spara wd='WD' --spara vg='VG'\
--limit 100

for n in frisbee[1:-1]:
    print("https://www.conceptnet.io"+n)
```

Get all the nodes in ConceptNet, Wikidata-CS and VisualGenone that contain `frisbee`

[30]:
```
!$kypher -i $NKG \
--match '(n1)-[r {label: label}]->(n2), (r)-[:source]->(source), (n1)-[:label]->(n1_label)' \
--where 'source in [$cn, $vg, $wd] and n1 =~ ".*frisbees?.*"' \
--return 'distinct source as source, n1 as node1, n1_label as `node1 label`, label as relation, n2 as node2' \
--order-by 'source, n1' \
--spara cn='CN' --spara wd='WD' --spara vg='VG'\
-o $CSKG/frisbee.tsv
```

We have many edges that relate to `frisbee`

[31]:
```
!wc $CSKG/frisbee.tsv
```

```
 1  6 40 ../output/frisbee.tsv
```

[32]:
```
!head $CSKG/frisbee.tsv | column -t -s $'\t'
```

```
/bin/sh: 1: column: not found
head: write error: Broken pipe
```

[ ]: