Raaghav_94_NLP6_Keyword

March 3, 2024

1 Keyword Extraction

- Develop a model for identifying and extracting keywords/ key phrases from the input sequence.
- Use Named Entity Recognition and PoS tagging as a feature that contribute to the extraction.

1.0.1 Import Dependencies

[ ]:
```
import os
import re
import pandas as pd
from collections import defaultdict
import spacy
```

[ ]:
```
# !python -m spacy download en_core_web_lg
```

1.0.2 Load the Data

[ ]:
```
file_name = os.listdir("SemEval2017/docsutf8")
key_name = os.listdir("SemEval2017/keys")
```

[ ]:
```
data = defaultdict(list)
for f in file_name:
    with open(f"SemEval2017/docsutf8/{f}", "r") as fi, open(f"SemEval2017/keys/
 ↪{f.split('.')[0] + '.key'}", "r") as key:
        data['file'].append(fi.read())
        data['key'].append(key.read().splitlines())
df = pd.DataFrame(data)
```

[ ]:
```
def remove_citations(text):
    st = r"\[\d+\]"
    return re.sub(st,'', text)

df['file'] = df['file'].map(remove_citations)
df.head()
```

[ ]:
```
                                                file  \
0  These results demonstrate that SW-SVR predicts…
```

```
1  As already discussed, in dilute flows the choi…
2  Fig. 7 shows the relationship between the test…
3  The Magnox reactors represent the first genera…
4  Aeroengine turbine disks often consist of para…

                                                 key
0  [combination of several algorithms, D-SDC, dyn…
1  [collision model, dilute flows, fluid, hard an…
2  [Al2O3 nanoparticles, anodizing coatings, anod…
3  [air, carbonaceous deposits, carbon dioxide, c…
4  [Aeroengine turbine disks, decrease the mechan…
```

1.1 Model Building

```
[ ]: nlp = spacy.load('en_core_web_lg')
```

1.1.1 Extracting Named Entities

```
[ ]: def extract_named_ents(text):
         return [ent.text for ent in nlp(text).ents]

     def add_named_ents(dataframe):
         dataframe['named_ents'] = dataframe['file'].apply(extract_named_ents)

     add_named_ents(df)
     df['named_ents'].head()
```

```
[ ]: 0    [SW-SVR, D-SDC, SW-SVR, Firstly, 6 h, Fig, 3, …
     1                    [14, one, Fig, 15, Figs, 16, 17]
     2                                        [7, 24,25]
     3                          [Magnox, first, UK, Magnox]
     4                                                  []
     Name: named_ents, dtype: object
```

1.1.2 Extracting Nouns

```
[ ]: def extract_nouns(text):
         keep_pos = ['PROPN', 'NOUN']
         return [tok.text for tok in nlp(text) if tok.pos_ in keep_pos]

     def add_nouns(dataframe):
         dataframe['nouns'] = dataframe['file'].apply(extract_nouns)

     add_nouns(df)
     df['nouns'].head()
```

```
0    [results, SW, SVR, data, prediction, performan…
1    [dilute, choice, sphere, sphere, models, time,…
2    [Fig, ., relationship, testing, time, friction…
3    [Magnox, reactors, generation, gas, reactors, …
4    [turbine, disks, paramagnetic, Nickel, alloys,…
Name: nouns, dtype: object
```

1.1.3 Extracting Noun Phrases

```
def extract_noun_phrases(text):
    return [chunk.text for chunk in nlp(text).noun_chunks]

def add_noun_phrases(dataframe):
    dataframe['noun_phrases'] = dataframe['file'].apply(extract_noun_phrases)

add_noun_phrases(df)
df['noun_phrases'].head()
```

```
0    [These results, SW-SVR, complicated micrometeo…
1    [dilute, the choice, the hard sphere, soft sph…
2    [Fig, the relationship, the testing time, fric…
3    [The Magnox reactors, the first generation, ga…
4    [Aeroengine turbine disks, paramagnetic, that,…
Name: noun_phrases, dtype: object
```

1.1.4 Extracting Compound Words

```
def extract_compounds(text):
    comp_idx = 0
    compound = []
    compound_nps = []
    tok_idx = 0
    for idx, tok in enumerate(nlp(text)):
        if tok.dep_ == 'compound':

            children = ''.join([c.text for c in tok.children])
            if '-' in children:
                compound.append(''.join([children, tok.text]))
            else:
                compound.append(tok.text)

            try:
                tok_idx = [c for c in tok.children][0].idx
            except IndexError:
                if len(compound) == 1:
                    tok_idx = tok.idx
            comp_idx = tok.i
```

```
        if tok.i - comp_idx == 1:
            compound.append(tok.text)
            if len(compound) > 1:
                compound = ' '.join(compound)
                compound_nps.append(compound)

            tok_idx = 0
            compound = []

    return list(set(compound_nps))

def add_compounds(dataframe):
    dataframe['compounds'] = dataframe['file'].apply(extract_compounds)

add_compounds(df)
df['compounds'].head()
```

[ ]:
```
0    [training periods, kernel approximation, predi…
1    [sphere models, collision model, particle equa…
2    [testing time, Al2O3 nanoparticles, AZ31 magne…
3    [honeycomb network, carbon monoxide, weight lo…
4    [turbine disks, detection principle, remanence…
Name: compounds, dtype: object
```

1.1.5 Combining Compound Words and Entities

[ ]:
```
def extract_comp_nouns(row, cols=[]):
    return {noun for col in cols for noun in row[col]}

def add_comp_nouns(df, cols=[]):
    df['comp_nouns'] = df.apply(extract_comp_nouns, axis=1, cols=cols)

cols = ['nouns', 'compounds']
add_comp_nouns(df, cols=cols)
df['comp_nouns'].head()
```

[ ]:
```
0    {training periods, periods, SVR, feature mappi…
1    {profiles, sphere models, sphere, particle, co…
2    {anodizing, Al2O3 nanoparticles, nanoparticles…
3    {layer, generation, carbon monoxide, weight lo…
4    {direction, density, flux gate magnetometer, p…
Name: comp_nouns, dtype: object
```

[ ]:
```
def drop_duplicate_np_splits(ents):
    drop_ents = set()
    for ent in ents:
```

```
        if len(ent.split(' ')) > 1:
            for e in ent.split(' '):
                if e in ents:
                    drop_ents.add(e)
    return list(ents - drop_ents)

df['pred_keys'] = df['comp_nouns'].apply(drop_duplicate_np_splits)
```

[ ]:
```
df['pred_keys'].head()
```

[ ]:
```
0    [training periods, SVR, feature mapping, algor…
1    [sphere models, collision model, Figs, quantit…
2    [Al2O3 nanoparticles, coatings, fluctuation, r…
3    [layer, generation, carbon monoxide, weight lo…
4    [direction, flux gate magnetometer, field, mat…
Name: pred_keys, dtype: object
```

1.2 Comparing with the Given Keys

[ ]:
```
df[['key', 'pred_keys']]
```

[ ]:
```
                                                   key  \
0    [combination of several algorithms, D-SDC, dyn…
1    [collision model, dilute flows, fluid, hard an…
2    [Al2O3 nanoparticles, anodizing coatings, anod…
3    [air, carbonaceous deposits, carbon dioxide, c…
4    [Aeroengine turbine disks, decrease the mechan…
..                                                 …
488  [adding dipole contributions, algorithm, corre…
489  [biochemical and mechanical conditions, compos…
490  [agricultural producing, agricultural products…
491  [coarse high-order prisms, dense linear bounda…
492  [ARS(2,3,2) scheme, atmospheric motion, HEVI s…

                                             pred_keys
0    [training periods, SVR, feature mapping, algor…
1    [sphere models, collision model, Figs, quantit…
2    [Al2O3 nanoparticles, coatings, fluctuation, r…
3    [layer, generation, carbon monoxide, weight lo…
4    [direction, flux gate magnetometer, field, mat…
..                                                 …
488  [algorithm, interaction potential, subtraction…
489  [sensitivity features, importance, parameters…
490  [demand goods, Pareto improvement, profit allo…
491  [layer, interpolation, mapping, isoparametric,…
492  [Newton solver, Strang splitting, Ascher et al…
```

[493 rows x 2 columns]