

KG Improvement: Embeddings

Credit

This assignment is identical to the one in the City uni course.

Ontology Embeddings (Task Vector)

This task consists in creating embeddings capturing the rich semantics of the created ontology and generated data. We will use the tool OWL2Vec*. These embeddings can be used in a subsequent Machine Learning model that requires as input the encoding of the ontology entities. In this coursework we are computing clusters of the ontology entities.

Subtask Vector.1 Run OWL2Vec* with the created ontology and generated data. Test three different configurations. Save the generated vectors in both binary and textual format (20%).

Subtask Vector.2 Select 5 pairs of entities (for any of the tested configurations) and discuss the similarity of their vectors (e.g., compare the vectors of the concept pizza:Margherita and the word “pizza”). (20%).

Subtask Vector.3 Compute clusters (e.g., using K-means) for the embeddings of the ontology concepts (i.e., the URI embeddings). Test the algorithm with different number of clusters, visualize the clusters and discuss the results (50%).

Subtask Vector.4 Correctness and documentation of the codes (10%).

In the first step, to download the "owl2vec" program, we go to its GitHub repository and download its zip file.

cache	7/1/2023 12:09 PM	File folder	
case_studies	7/1/2023 2:27 PM	File folder	
docs	4/9/2023 3:28 PM	File folder	
onto_complete	4/9/2023 3:28 PM	File folder	
output	7/1/2023 5:36 PM	File folder	
owl2vec_star	7/1/2023 11:59 AM	File folder	
tests	4/9/2023 3:28 PM	File folder	
AUTHORS.rst	4/9/2023 3:28 PM	Restructured Text ...	1 KB
CONTRIBUTING.rst	4/9/2023 3:28 PM	Restructured Text ...	4 KB
default.cfg	7/1/2023 5:29 PM	Configuration Sou...	3 KB
default_multi.cfg	4/9/2023 3:28 PM	Configuration Sou...	2 KB
HISTORY.rst	4/9/2023 3:28 PM	Restructured Text ...	1 KB
jupyter_notebook_owl2vec.ipynb	4/9/2023 3:28 PM	Jupyter Source File	17 KB
LICENSE	4/9/2023 3:28 PM	File	12 KB
Makefile	4/9/2023 3:28 PM	File	3 KB
MANIFEST.in	4/9/2023 3:28 PM	IN File	1 KB
OWL2Vec_Standalone.py	7/1/2023 5:05 PM	Python Source File	12 KB
OWL2Vec_Standalone_Multi.py	4/9/2023 3:28 PM	Python Source File	10 KB
README.rst	4/9/2023 3:28 PM	Restructured Text ...	7 KB
requirements_dev.txt	4/9/2023 3:28 PM	Text Document	1 KB
requirements_owl2vec.txt	4/9/2023 3:28 PM	Text Document	1 KB
setup.cfg	4/9/2023 3:28 PM	Configuration Sou...	1 KB
setup.py	4/9/2023 3:28 PM	Python Source File	2 KB

Now it is necessary to install the required libraries, namely numpy, nltk, gensim, and scikit-learn.

Furthermore, to work with this program, we need Python 3.7 or 3.8 installed. We have installed Python 3.7. To install these libraries, you can use the command ``pip install owl2vec_star``. However, installing these libraries using this method may result in a lower version of gensim (below version 4) installed on Python 3.7. This can cause errors during the model training stage because the ``vector_size`` argument is not used in versions below 4 which is defined in codes (in fact codes are updated and they use latest gensim version but default libraries are not the same as codes). To resolve this issue, you can manually upgrade the gensim library to the latest version.

```
Train the embedding model ...
Traceback (most recent call last):
  File "OWL2Vec_Standalone.py", line 218, in <module>
    min_count=int(config['MODEL']['min_count']), seed=int(config['MODEL']['seed']))
TypeError: __init__() got an unexpected keyword argument 'vector_size'
```

Another issue we encountered during the program's execution was an error that occurred while reading the ontology file. This error was resolved by adding `encoding='utf-8'` to the following code when reading the file.

```
Extract axioms ...
Traceback (most recent call last):
  File "OWL2Vec_Standalone.py", line 86, in <module>
    f.write('%s\n' % ax)
  File "C:\Users\Alireza\AppData\Local\Programs\Python\Python37\lib\encodings\cp1252.py", line 19, in encode
    return codecs.charmap_encode(input,self.errors,encoding_table)[0]
UnicodeEncodeError: 'charmap' codec can't encode characters in position 187-192: character maps to <undefined>
```

```
# Extract axioms in Manchester Syntax if it is not pre_axiom_file is not set
if 'pre_axiom_file' not in config['DOCUMENT']:
    print('\nExtract axioms ...')
    projection.createManchesterSyntaxAxioms()
    with open(os.path.join(config['DOCUMENT']['cache_dir'], 'axioms.txt'),
        'w', encoding='utf-8') as f:
        for ax in projection.axioms_manchester:
            f.write('%s\n' % ax)
```

Now, by configuring the default.cfg file and using the following code, the program is executed, and the embedding process is carried out. The ontology created in the previous stages is used as input in the file.

python OWL2Vec_Standalone.py -config_file default.cfg

configurations:

```
[BASIC]
# the file of input ontology; mandatory; you can also set it as a projected
ontology
ontology_file= ./case_studies/1.owl
#ontology_file = ./case_studies/data/helis_v1.00.origin.owl
# ontology_file = ./cache/projection.ttl
#ontology_file = ./case_studies/pizza/pizza.owl
# ontology_file = ./case_studies/simple/test_ontology.owl

# the output director for the embedding; if not set, it uses the default:
$cache_dir/output
embedding_dir = output
```

Additionally, at the end of the program file, the code for saving the gensim model was facing an issue due to version compatibility between the code and the library. This issue has been fixed and the code was modified as follows.

```

embedding_dir = config['BASIC']['embedding_dir']

# Save vectors in binary format
binary_path = embedding_dir + '.bin'
model_.wv.save_word2vec_format(binary_path, binary=True)

# Save vectors in textual format
text_path = embedding_dir + '.txt'
model_.wv.save_word2vec_format(text_path, binary=False)






# Save the model
model_save_path = embedding_dir + '.model'
model_.save(model_save_path)

import pandas as pd
# Save vectors as CSV
csv_path = embedding_dir + '.csv'
word_vectors = model_.wv.vectors
word_list = model_.wv.index_to_key
df = pd.DataFrame(word_vectors, index=word_list)
df.to_csv(csv_path)






print(f"Vectors saved in binary format: {binary_path}")
print(f"Vectors saved in textual format: {text_path}")
print('Time for learning the embedding model: %s seconds' % (time.time() -
start_time))
print('Model saved. Done!')

```

Now, the model has been executed with three different configurations, and the output has been saved in the formats of txt, csv, bin, and model.

 output1	7/1/2023 4:02 PM	File folder	
 output2	7/1/2023 4:09 PM	File folder	
 output3	7/1/2023 4:13 PM	File folder	
 output4	7/1/2023 5:36 PM	File folder	
 output.txt	7/1/2023 3:50 PM	Text Document	24,290 KB

The output for each model has been placed in separate directories, and the configuration that led to the creation of each model has also been provided.

Name	Date modified	Type	Size
 default.cfg	7/1/2023 5:29 PM	Configuration Sou...	3 KB
 output.bin	7/1/2023 5:32 PM	BIN File	12,295 KB
 output.csv	7/1/2023 5:32 PM	Microsoft Excel C...	33,980 KB
 output.model	7/1/2023 5:32 PM	MODEL File	24,290 KB
 output.txt	7/1/2023 5:32 PM	Text Document	33,947 KB

Csv output of the model:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	words	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	of	-0.40977	0.886151	0.182495	-0.62767	0.59314	0.551254	-1.06692	0.045021	-0.6336	0.151441	0.231199	0.009622	0.161356	0.097154	0.229495	-0.45412	-0.1863	-0.70867	-0.23732	0.184143	-0.57764
3	type	-0.73668	0.565664	0.254811	-1.107	0.791414	0.929359	-1.5268	0.268393	-0.6316	0.236123	-0.12997	0.266547	0.312406	0.014154	0.274333	-0.20282	0.170158	-0.53041	-0.89649	-0.24838	-0.37708
4	pizza	-0.01081	0.176755	0.279866	-0.73813	0.162676	0.454871	-1.01089	0.155637	-0.4107	0.048421	0.027632	0.292053	-0.33195	0.234242	0.176699	-0.32366	-0.24088	-0.49175	-0.03846	0.048141	-0.37799
5	http://www.0.610288	0.142879	-0.76999	0.828864	-0.03921	-0.1811	-0.91399	-0.00649	-0.06747	0.049554	-0.06016	0.072863	0.098257	0.467962	-0.4837	0.098411	-0.08078	-0.20696	-0.1631	0.579198	-0.22223	
6	http://www.0.556674	-0.13644	-1.05098	1.159625	-0.1217	0.046027	-0.98741	-0.04484	0.084513	0.853886	-0.19046	0.403194	0.281881	-0.19766	-0.31335	0.08632	-0.21492	-0.57313	0.355524	0.027932	0.643414	
7	http://www.0.387974	-0.24986	-0.51401	0.87242	0.703754	0.179437	-0.10345	-0.10056	0.50811	0.101474	0.052492	0.776553	0.097578	0.440165	-0.21942	-0.21054	-0.14724	-0.51138	-0.41488	-0.08771	-0.11171	
8	menu	-0.44961	0.167869	0.185687	-0.97275	0.471624	0.688158	-1.12542	0.228717	-0.21751	0.361685	0.502492	0.645963	-0.17913	-0.21628	0.198061	-0.18517	-0.50083	-0.42973	-0.34254	0.131241	-0.27365
9	http://www.0.421021	-0.19439	-0.71086	0.273302	0.078534	-0.79913	-0.94164	0.28268	0.294667	0.545244	0.233551	0.416025	0.55149	0.015817	-0.0626	0.589919	-0.23517	-0.04749	-0.73561	0.548355	-0.22336	
10	value	-0.4899	0.733624	0.91233	-0.04361	0.140819	1.15574	-0.80192	-0.27264	-0.68897	0.06377	0.352576	0.172204	-0.04903	0.13969	0.367567	-0.75604	-0.46585	-1.03022	0.167025	0.359872	-0.37463
11	http://www.0.37676	-0.87488	-1.36845	1.635548	0.226707	0.029118	-0.82401	-0.01628	0.156976	1.000743	-0.24087	0.024707	0.928222	0.433183	-0.18787	-0.09541	-0.08784	-0.50503	-0.01994	0.829937	-0.44208	
12	city	-0.40229	0.548306	0.442274	-0.5109	0.046345	1.922639	-0.93978	0.528863	-0.16487	0.271114	0.755251	0.366814	0.143689	-0.1571	-0.23315	0.252984	0.151025	-0.65631	-0.99917	-0.57776	-0.10713
13	pname	-0.12274	1.259735	0.369255	-0.42882	0.259054	0.545669	-0.43919	0.637095	-0.14668	-0.27116	-0.34082	0.481447	-0.4243	0.231643	-0.05725	-0.05819	-0.26111	-0.00065	-0.44015	0.283424	-0.74448
14	http://www.0.604746	-0.3484	-0.93411	0.914367	0.667263	-0.24626	-1.04261	0.60747	0.173006	1.286604	0.278834	0.35174	0.717208	-0.133	-0.40816	0.20246	-0.25718	-0.06825	0.425223	0.552382	0.028583	
15	http://www.0.45588	-0.12066	-0.46386	0.448709	0.075588	-0.07647	-1.95728	0.421824	0.52142	0.540224	-0.03763	0.26987	0.887546	0.365435	0.835865	-0.17509	0.268396	0.118397	-0.29742	0.006818	0.098943	
16	food	0.059029	1.21769	-0.39891	-0.9042	0.480706	1.051973	-0.96529	0.384375	-0.37196	0.249649	0.065176	0.673844	-0.63348	0.38472	-0.46947	0.477567	-0.84028	-0.6316	-0.07111	-0.27247	-0.61131
17	store	-0.18506	0.852413	-0.42891	-0.59768	0.693898	0.9596	-0.50421	0.235608	-0.0426	0.312952	-0.37265	0.053502	-0.16282	-0.34934	-0.00042	0.927674	-1.22861	-0.94101	-0.21025	-0.57604	-0.19192
18	http://www.0.11674	0.874818	0.104804	-0.30387	-0.99322	0.124855	-0.96748	0.82697	-0.44926	0.229985	-0.36773	0.066842	1.048256	-0.54951	-0.29163	0.63702	0.144575	0.060453	-0.6577	0.65214	-0.29119	
19	state	-0.79614	0.440926	0.406818	-0.32676	0.893798	1.662459	-0.46149	0.784271	-0.43179	-0.49956	-0.1096	0.269868	0.716704	0.334198	-0.52204	0.444737	0.155464	-0.27112	-0.53363	-0.53195	-0.90219
20	category	0.099413	-0.05913	-0.38659	-1.2332	1.086625	0.327634	-1.40374	0.469413	-0.85555	0.76909	-0.25878	0.893746	-0.21487	-0.02583	1.118016	-0.0324	-0.68428	-1.0509	-0.71625	-0.37163	-0.35895
21	http://www.0.355632	0.790838	0.11987	0.5799	-0.8767	-0.87271	-0.62447	-0.13122	-0.92852	0.747044	-0.75876	0.746888	-0.48645	-0.37112	-0.37931	0.760576	-0.23585	0.021295	-0.5839	0.174783	-0.47652	
22	Type	0.728399	-0.15343	-0.898	1.314398	-0.00239	-0.53341	-0.93616	-0.13924	0.042364	0.314591	-0.34086	0.550325	-0.13007	0.213477	-0.30047	0.030064	-0.05643	-0.31273	0.294354	0.039779	0.090667
23	USD	0.188554	-0.81793	-0.101048	1.16917	0.567489	0.579918	-1.24115	-0.47576	0.077457	0.393889	0.38656	0.347631	0.363097	0.71248	-0.44636	-0.30958	0.07459	-0.31895	-0.52756	0.34678	-0.28857
24	usd	-0.45085	0.969379	0.115139	-0.59167	0.42513	0.681927	-1.26119	-0.36087	-0.28076	-0.20468	0.292386	0.42975	-0.0722	-0.22146	0.20793	-0.39406	-0.59527	-0.42733	-0.30601	0.397994	-0.2816
25	http://www.0.118469	-0.75616	-0.24455	1.81579	-0.33906	-1.47735	-0.44865	0.608548	0.352218	0.341292	-0.713	-0.63022	0.827122	-0.13069	-0.53768	0.048639	-0.16799	-0.64745	-0.0129	-0.35769	-0.108907	
26	description	-0.00046	0.475216	1.29022	0.357286	0.537077	0.096309	-0.57841	0.295676	-0.53416	-0.50565	-0.36862	0.162553	0.428963	-0.05121	-0.49534	-0.10543	-0.57909	-0.64235	-0.23493	-0.10276	-0.38485
27	http://www.0.507536	0.418554	-0.35577	0.431276	-1.34946	-0.25798	-0.21602	0.645655	-0.36531	-0.07126	-1.14329	0.373295	0.918292	0.454901	-0.43941	0.860794	0.541731	-0.1123	-0.55211	0.415896	-0.28957	
28	name	-0.6254	1.091474	0.575748	-0.16774	1.215371	1.318692	-1.0949	0.183614	-0.36334	0.361426	0.635202	0.264958	0.099526	0.353343	-0.0735	0.349393	-0.59472	-0.96552	-0.91291	-0.15468	-0.57106
29	http://www.0.675073	1.076281	-0.56827	0.741599	-1.20159	0.717645	-0.11152	0.006955	-0.60233	0.558131	-0.83382	0.335633	0.615282	0.210535	0.251033	0.212856	-0.09826	-0.77586	-0.56404	0.014644	-0.28072	

Text output of model:

```
output.txt - Notepad
File Edit Format View Help
29094 100
of -0.40976885 0.8861513 0.18249455 -0.62766606 0.5931397 0.5512542 -1.0669173 0.045021478 -0.6336036 0.15144074 0.23119926 0.009621853 0.16135617 0.097153954 0.229495 -0.4541245 -0.1863024
852 0.17205898 -0.43116653 0.7438861 -0.68053055 0.22374398 0.24158706 0.3981598 -0.05885362
type -0.73667884 0.5656641 0.25481087 -1.10700895 0.7914141 0.9293587 -1.526802 0.26839286 -0.6331557 0.23612344 -0.12996525 0.26654693 0.3124058 0.01415407 0.2743326 -0.20282061 0.17015752
71302 1.3231885 -0.5218812 0.5181832 -0.36254933 0.37287998 0.39847177 0.15506582 0.006518373
pizza -0.010809262 0.1767548 0.2798657 -0.73813385 0.16267647 0.45487127 -1.0108865 0.15563673 -0.41070148 0.04842101 0.027632307 0.29205298 -0.33195052 0.23424244 0.17669937 -0.32366353 -0.26021737 0.7805855 -0.64835674 1.064181 -0.59966993 0.014865542 0.35025194 0.15217692 -0.07618414
http://www.semanticweb.org/owl2vec?typeOf 0.6102876 0.14287904 -0.7699911 0.8288636 -0.039205074 -0.18110473 -0.91399026 -0.006494744 -0.06747401 0.049553927 -0.06016171 0.072862916 0.09825
0.6737642 0.20030238 0.67225575 -0.5653844 0.23872358 -0.16969305 0.93224484 0.6125878 0.122182295 1.190319 -0.72457546 -0.43648645
http://www.w3.org/1999/02/22-rdf-syntax-ns#type 0.6566743 -0.13643858 -1.0509828 1.1596246 -0.1217001 0.04602724 -0.98741096 -0.044839446 0.08451332 0.85388553 -0.1904567 0.40319356 0.28188
1.1609011 0.62124985 0.4175284 0.014977509 0.60754555 -0.19566147 0.6978066 0.13674946 -0.026572146 1.2710829 -0.13946784 -0.6934504
http://www.city.ac.uk/ds/inm713/hadi_ghasemi/food/pizza 0.38797405 -0.24985752 -0.51401323 0.87241954 0.7037544 0.17943671 -1.0134532 -0.100555934 0.50811005 0.10147439 0.19287485 0.7765525
-0.6211591 -0.9332045 0.63003893 -0.7094472 0.048329514 0.16753562 -0.5297951 1.2330211 0.6471834 -0.3511887 0.7297898 -0.8159478 -0.37119907
menu -0.44961333 0.1678694 0.18568712 -0.9727537 0.4716235 0.6881579 -1.1254205 0.22871718 -0.2175098 0.3616852 0.5024923 0.64596266 -0.17913355 -0.21627948 0.19806074 -0.1851655 -0.5008259
6383 -0.47292188 0.8038057 -0.6472678 -0.19566764 0.53195256 0.18871942 0.10273157
http://www.city.ac.uk/ds/inm713/hadi_ghasemi/relations/menu 0.4210205 -0.19438748 -0.7108562 0.27330175 0.078533836 -0.7991273 -0.9416435 0.28267992 0.29466653 0.5452437 0.23355111 0.416024
4162934 -0.69032353 0.10353554 -0.3071478 0.05814553 0.4285911 -0.33258298 0.7799909 0.4224167 -0.051076267 1.1344305 -0.82821083 0.16919726
value -0.48990238 0.7336237 0.19123329 -0.04360752 0.14081861 1.1557397 -0.80192167 -0.27263916 -0.688967 0.06376977 0.35257638 0.17220367 -0.049030058 0.13968977 0.367567 -0.7560378 -0.465
7 1.3250171 -0.38389757 0.9765255 -0.21719168 0.11258478 0.2257276 0.23659045 0.004035162
http://www.city.ac.uk/ds/inm713/hadi_ghasemi/relations/value 0.3767624 -0.8748753 -1.3684491 1.635548 0.22670747 0.02911817 -0.8240063 -0.016275406 0.15697606 1.0007428 -0.24087416 0.02470
4235138 -1.3185296 -0.112131004 -0.5850179 0.7149102 0.5458315 -0.038655624 0.9086619 0.7184745 0.14434996 0.3537129 -0.16166587 -0.19359381
city -0.40229356 0.5483056 0.44227415 -0.5109041 0.0463448 1.9226394 -0.9397813 0.5288627 -0.16487134 0.2711137 0.75525147 0.36681443 0.1436885 -0.1570974 -0.23314728 0.2529844 0.1510252 -0.
02 0.72093534 -0.5535232 1.1337849 -1.0441345 0.47631916 0.23169488 -0.8568012 -0.8151422
pname -0.122743264 1.259735 0.369255 -0.4288196 0.25905362 0.5456692 -0.43919012 0.6370954 -0.14667879 -0.2711625 -0.34082144 0.4814468 -0.4242966 0.23164295 -0.05724933 -0.05819223 -0.261
36094 -0.64563936 -0.06444471 -0.62607014 0.9383693 0.10620838 -0.38439858 0.4634069 0.4297934 -0.25614318
```

	Conf 1	Conf 2	Conf 3
Embed size	100	150	80
Iterations	10	15	8
Window	5	10	2
Negative	2	35	15
Seed	42	100	25

Next we use the csv file and read it into a pandas dataframe and using that we trained kmeans for k between 2 and 20 and measure SSE and Silhouette Score and then plot them for each k:

```
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import matplotlib.pyplot as plt

# Read the CSV file
data = pd.read_csv('/content/drive/MyDrive/output.csv')

# Extract the embeddings
embeddings = data.iloc[:, 1:].values

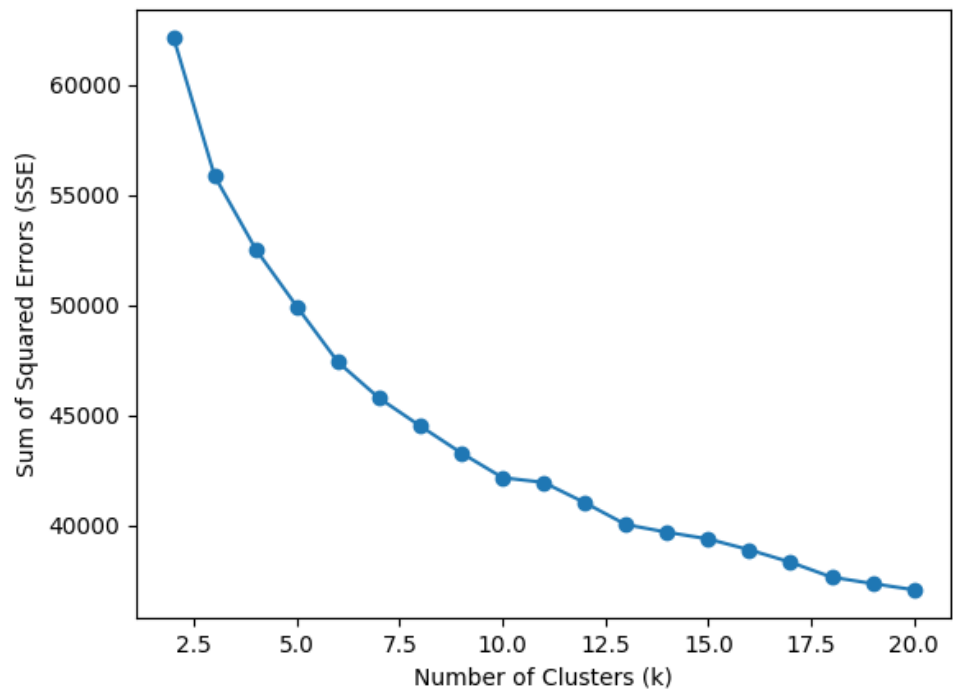
# Initialize lists for storing SSE and Silhouette scores
sse = []
silhouette_scores = []

# Perform clustering for k between 2 and 20
for k in range(2, 21):
    # Initialize KMeans with the current value of k
    kmeans = KMeans(n_clusters=k)
    # Fit the embeddings to the model
    kmeans.fit(embeddings)
    # Append the SSE value to the list
    sse.append(kmeans.inertia_)
    # Calculate the Silhouette score and append it to the list
    silhouette_scores.append(silhouette_score(embeddings, kmeans.labels_))

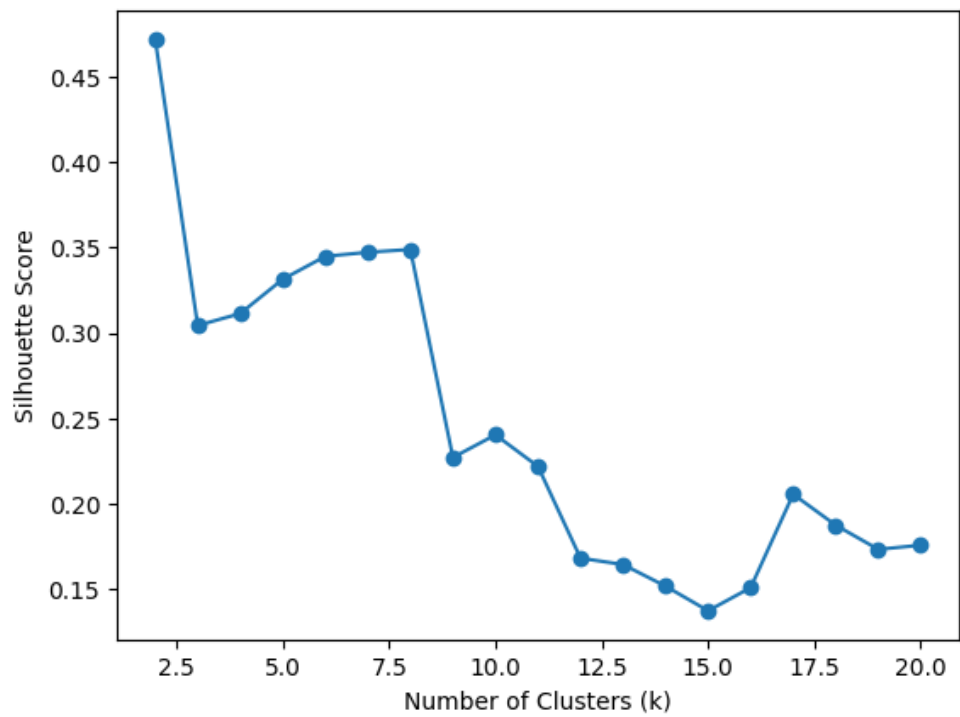
# Plot SSE values
plt.plot(range(2, 21), sse, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Sum of Squared Errors (SSE)')
plt.title('Elbow Method')
plt.show()

# Plot Silhouette scores
plt.plot(range(2, 21), silhouette_scores, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score')
plt.show()
```

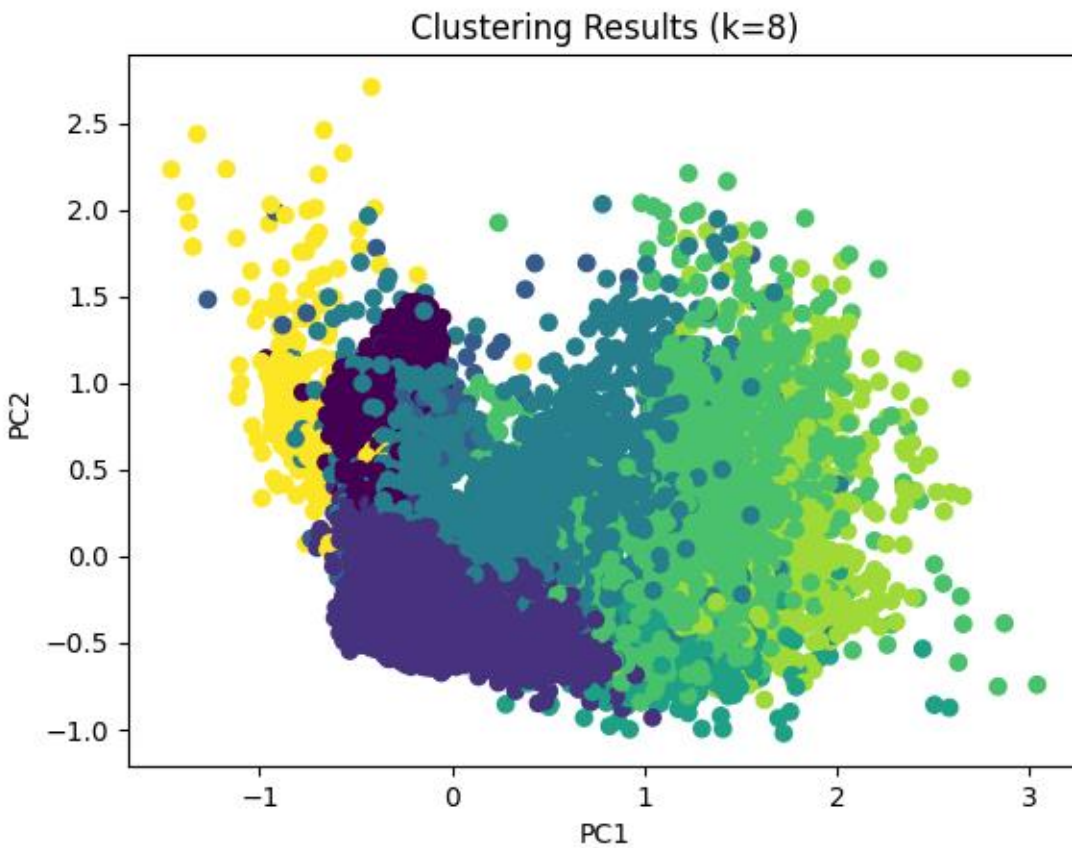
Elbow Method



Silhouette Score



Based on the uniformity of the SSE (Sum of Squared Errors) graph, it may be challenging to determine the optimal number of clusters using the elbow method. That's why the Silhouette graph is also plotted alongside it. Now, considering these two graphs, we can conclude that 8 seems to be an appropriate number for the cluster count.



After performing clustering with $k=8$, we apply Principal Component Analysis (PCA) to transform the data into a 2-dimensional space. But why do we use PCA and how does it work?

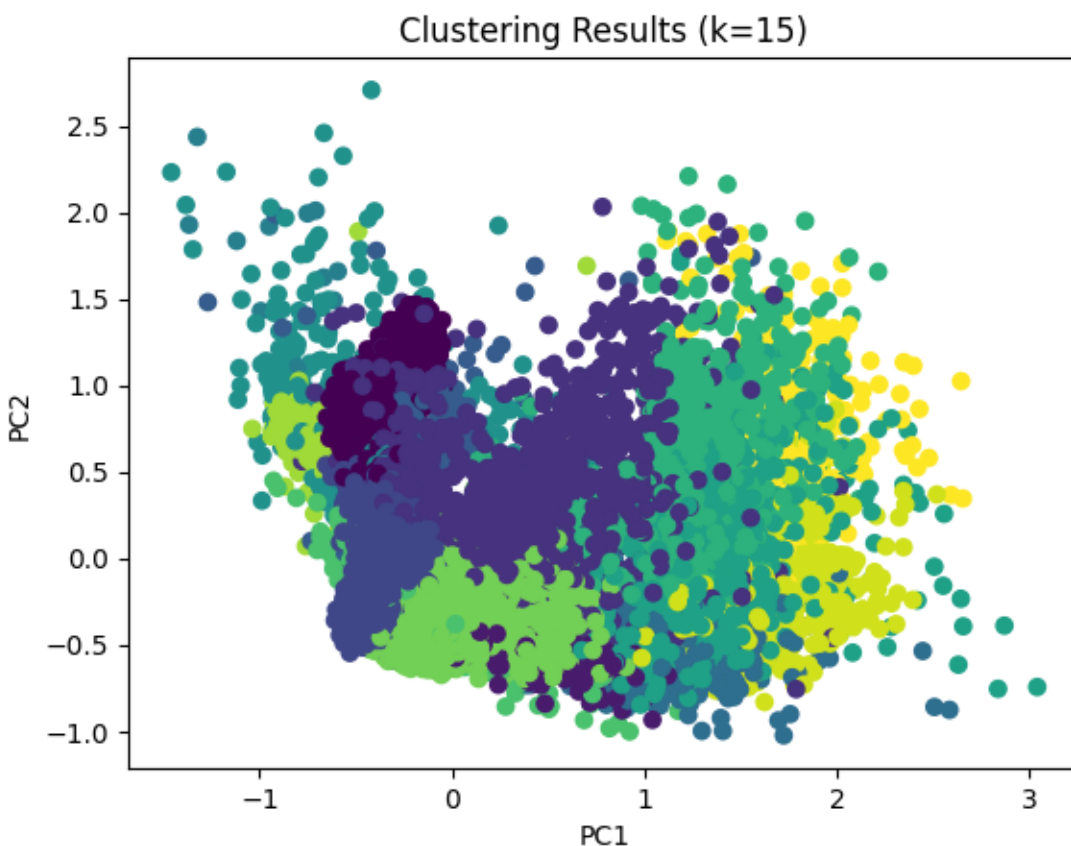
PCA is a dimensionality reduction technique that is commonly used in data analysis and visualization. It helps in reducing the number of variables in a dataset while preserving the most important information. By doing so, it simplifies the complexity of the data and allows us to visualize it in a more manageable way.

In our case, we choose to reduce the dimensionality of the data to 2 because it is easier to represent and visualize data in a 2-dimensional space. While the original data may have multiple dimensions, representing it in 2D allows us to plot it on a graph and easily observe patterns, clusters, or relationships between the data points.

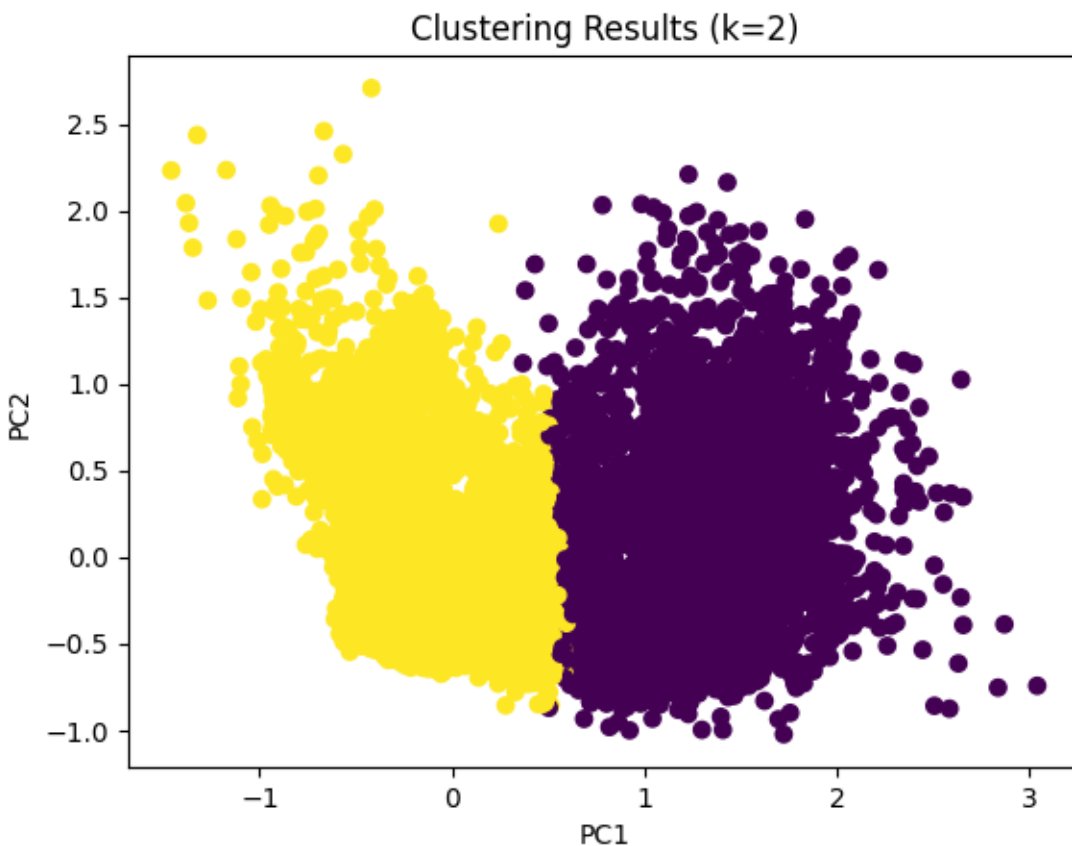
Furthermore, PCA is preferred for dimensionality reduction because it finds the directions (principal components) in the data that capture the most significant variance. By projecting the data onto these principal components, we can retain a significant portion of the original data's information while reducing the dimensionality.

Therefore, utilizing PCA to transform the clustering results into 2D allows us to visualize and analyze the data more effectively, enabling easier interpretation and understanding of the underlying patterns and structures within the data.

The plot on the previous page was the result of dimensionality reduction to 2D and then visualizing the clustering outcome. As it appears, the clustering has been well-performed, and the clusters are well-separated. This outcome was predictable from the high values of the Silhouette coefficient. For example, for the number 15 with the lowest Silhouette coefficient, we expect an inappropriate clustering outcome, as illustrated below.



Indeed, the highest Silhouette coefficient and the maximum break in the elbow plot are associated with the number of clusters being 2. The clustering outcome for this configuration is depicted below.



It is

speculated that clustering with 2 clusters divides the entities into two categories: food and location. Words related to menus, food, pizza, prices, etc., are grouped together in one cluster, while other words associated with cities, countries, states, addresses, zip codes, etc., are placed in the other cluster.

Considering that there were 7 classes in the ontology, it seems that an 8-cluster clustering corresponds to these 7 classes, plus additional relationships. However, it is not certain in this regard, and this is just a possibility.

Furthermore, the requested task involved using one of the results and selecting 5 pairs of entities to discuss the similarity of their vectors.

Two functions were implemented for this task: one function searches and retrieves the vector representation of a word, while the other calculates the cosine similarity between two vectors and returns the similarity score.

Cosine similarity is a measure of similarity between two vectors in a high-dimensional space. It determines the cosine of the angle between these vectors, which represents their directional similarity.

To compute the cosine similarity, the vectors are first normalized to unit length. Then, the dot product of the normalized vectors is calculated, which indicates the alignment of their directions. Finally, the cosine similarity is obtained by dividing the dot product by the product of the vector magnitudes.

A cosine similarity score ranges from -1 to 1, where a score of 1 indicates that the vectors are identical, 0 indicates no similarity, and -1 indicates perfect dissimilarity. It is commonly used in various applications such as text analysis, recommendation systems, and information retrieval.

In the context of the given task, the cosine similarity function allows us to quantify the similarity between pairs of entities based on the similarity of their vector representations. These entity vectors are typically obtained through techniques like word embeddings or other semantic models. By comparing their cosine similarity scores, we can gauge the level of similarity or relatedness between the entities in question.

```
import pandas as pd

# Read the CSV file into a DataFrame

# Define a function to retrieve the embedding for a given entity
def get_embedding(entity):
    embedding = data[data['embedding'] == entity].values[:, 1:].flatten()
    if len(embedding) > 0:
        return embedding
    else:
        return None

# Test the function
entity_name = 'margherita'
embedding = get_embedding(entity_name)

if embedding is not None:
    print(f"Embedding found for entity '{entity_name}': {embedding}")
else:
    print(f"No embedding found for entity '{entity_name}'")
```

```
import pandas as pd
from scipy.spatial.distance import cosine

# Read the CSV file into a DataFrame
```

```

# Define the pairs of entities you want to compare
entity_pairs = [
    ('margherita', 'pizza'),
    ('margherita',
'http://www.city.ac.uk/ds/inm713/hadi_ghasemi/pizza_type/margherita'),
    ('margherita', 'pizza margherita'),
    ('margherita', 'bianca'),
    ('margherita', 'US')
]

# Iterate over the pairs of entities
for pair in entity_pairs:
    entity1, entity2 = pair

    # Extract the vectors for the entities
    vector1 = data[data['embedding'] == entity1].values[:, 1:].flatten()
    vector2 = data[data['embedding'] == entity2].values[:, 1:].flatten()

    # Calculate the cosine similarity between the vectors
    similarity_score = 1 - cosine(vector1, vector2)

    # Print the similarity score
    print(f"Similarity between {entity1} and {entity2}:
{similarity_score}")

```

first pair: (margherita, pizza)

```

Embedding found for entity 'margherita': [-0.90330523 0.7394789 0.47412163 0.18838903 -0.21315785 1.1000587
-0.48671755 0.7067824 -0.043227013 0.30160412 0.036673274 0.35353392
-0.29591385 -0.21899311 0.1859529 -0.38629165 -0.41257676 -0.9830062
0.39107108 0.5008457 -0.7510905 -0.06974218 -0.07735316 1.1725054
0.022743857 0.44085312 -0.14182943 0.7527927 0.587256 -0.62579453
-1.2131273 -0.15273029 -0.15025403 0.73014224 0.8381575 -1.5932864
0.20112906 -0.329726 -0.13647959 0.0157611 -0.058285628 0.97453094
0.35404804 -0.8270246 0.57804364 1.0531812 -0.46110415 1.2081242
0.8465455 -0.18759419 0.654666 -0.3338666 -1.7238941 0.033202678
0.7806755 0.047036797 0.16589133 0.2651749 -0.6556322 0.48066208
0.27154422 0.36355433 -0.385655 0.51798004 0.2425503 -0.023876999
-0.328722 0.38064495 0.36833328 -0.36140448 0.93799907 -0.28579277
-0.12850584 -0.21987931 -0.19258536 -0.22911851 -0.23532604 -0.9262261
-0.2037161 -0.07810496 0.30178446 -0.21756153 -0.098129354 -0.28979632
-0.51457703 0.5147229 -1.0270996 0.07224296 0.8688955 0.24956469
-0.18557581 0.04401348 1.0955815 0.044483595 0.56499535 -0.79124415
-0.31261757 0.19058938 -0.49876145 0.58133084]

```

Embedding found for entity 'pizza': [-0.010809262 1.1767548 0.2798657 -0.73813385 0.16267647 0.45487127
-1.0108865 0.15563673 -0.41070148 0.04842101 0.027632307 0.29205298
-0.33195052 0.23424244 0.17669937 -0.32366353 -0.24088012 -0.49174863
-0.03845937 0.048141498 -0.3779919 -0.37594017 -0.7446236 -0.27265146
0.9113576 -0.24292663 0.15465054 0.47121078 0.09281813 -1.2615762
-0.21030615 -0.22750564 -0.19920312 0.5317482 0.53527445 -0.68338364
0.54152584 -0.66247445 0.17173521 0.8110917 0.46959886 0.964048
0.06848939 -0.106631085 0.16819969 0.81150013 -0.23818539 0.10872622
1.0703415 0.15653253 0.6996008 -0.31092474 -0.2564935 -0.5461322
0.2541159 -0.34348425 0.26852936 -0.38139614 -0.9437395 0.03075634
0.61910784 0.37897593 0.0686741 0.5241072 0.30109674 0.18253705
-0.045871243 -0.0974827 0.6224751 0.35791326 0.331098 -0.5077695
-0.20289794 0.22260036 -0.19682439 0.4318151 0.57854813 -1.6176314
-0.7189334 -0.31307337 0.061425988 -0.13579975 0.23452547 0.22388542
-0.5977958 0.55163896 -1.1650362 -0.11926291 -0.20538685 0.23294513
0.37598872 -0.26021737 0.7805055 -0.64835674 1.1064181 -0.59966993
0.014685542 0.35025194 0.15217692 -0.07618414]

Similarity between **margherita** and **pizza**: **0.5790514051961504**

Given that "pizza" and "margherita" are both concepts within the ontology of pizza 2, it would be expected that they have a high degree of similarity. The obtained result confirms this expectation.

Second pair is : (maegherita, http://www.city.ac.uk/ds/inm713/hadi_ghasemi/pizza_type/margherita)

Embedding found for entity 'http://www.city.ac.uk/ds/inm713/hadi_ghasemi/pizza_type/margherita'
-0.99322754 0.35173795 0.24342537 0.21575028 -0.10712048 0.52698314
-0.113644436 0.594556 -0.47903514 0.0024799537 -0.41484413 -0.75180286
-0.34945825 0.42536336 -0.26746014 0.11175962 0.24339087 1.1125425
-0.039544113 -0.00028989522 -0.12847298 0.44738427 0.46079332 0.39386198
0.050278552 0.16589282 -0.2728623 0.7344415 0.78032416 -0.9580098
0.724796 -0.6959314 -1.007071 -0.003151602 0.11975696 0.80089396
0.7002867 -0.60968125 0.873632 -0.3886309 0.63623494 0.10535488
0.30221415 -0.18667845 -0.48342642 -1.1877892 -0.31630474 -0.44608605
0.19589162 0.049703233 -0.46780756 0.6180284 -0.40065965 0.03480072
-0.3002782 0.355755 -0.19326288 0.7651957 -0.4765681 0.0356085 0.44141346
0.45478377 -0.029896796 -0.9131536 0.35819072 0.4325591 -0.58239543
-0.30963314 0.36384422 0.7705927 0.8107347 0.48466682 -0.22825673
0.557312 0.45606315 0.503075 -0.13184063 -0.8262978 -0.8271187 0.10037292
-0.88124233 -0.4588036 -0.75791866 0.18643743 -0.77520543 0.23260868
-0.17663579 -0.5630636 1.2000818 0.6883577 0.14980972 0.9651493
-1.0901057 -0.07387962]

Similarity between **margherita** and
http://www.city.ac.uk/ds/inm713/hadi_ghasemi/pizza_type/margherita:
0.3179901843147217

In this part, the similarity between the word "margherita" and its URI address in the ontology was computed. It was expected to have a high similarity score, but despite being relatively similar, the obtained similarity score might not be as high as expected. One possible reason for this could be that the word embedding representations of "margherita" and its vector representation have significant differences, leading to a lower similarity score.

Third pair: (margherita, pizza margherita) 0.7940730842662989

4th pair: (margherita, bianca) 0.6125173372546684

5th pair: (margherita, US) 0.38876724626378323

6th pair: (US, <http://dbpedia.org/resource/US>): 0.43053579612827253

7th pair: (margherita , <http://dbpedia.org/resource/US>) 0.28827882378274783

By comparing the obtained similarity scores, we can observe that the term "margherita" has the highest similarity with the phrase "pizza margherita." This is logical since "margherita" within the pizza ontology is expected to have a higher similarity to the specific concept of "pizza margherita" than the general concept of "pizza." Additionally, as the contents of "pizza bianca" can be similar to "pizza margherita," we would expect its similarity score to be higher than the general concept of "pizza" but lower than "pizza margherita," which is indeed the case.

Another interesting point is the similarity score between "margherita" and "us" that is relatively low. However, what is noteworthy here is that the similarity score between "margherita" and its corresponding URI is higher than the similarity between "margherita" and "us." Similarly, the similarity between "us" and its URI is also higher. However, the lowest similarity score is between "margherita" and the URI corresponding to "us."

