Word2Vec Development

April 6, 2022

1 Experimentation with one-fifth of our dataset

1.1 (155k entries)

1.2 SETUP

[2]:
```
import gensim
import pandas as pd
```

[3]:
```
!dir
```

```
 Volume in drive C is Windows
 Volume Serial Number is 5029-78BD

 Directory of C:\Users\aKost\Desktop\2021-2022\SPRING 2022\capstone

04/06/2022  03:39 PM    <DIR>          .
04/06/2022  03:39 PM    <DIR>          ..
04/06/2022  03:39 PM    <DIR>          .ipynb_checkpoints
04/05/2022  05:18 AM        33,808,934 comments_dataframe_2.csv
04/04/2022  09:40 AM        34,204,493 comments_dataframe_4.csv
04/04/2022  05:38 PM        36,042,101 comments_dataframe_5.csv
04/02/2022  07:56 PM        60,328,596 june_to_march_postings.csv
04/01/2022  02:15 PM            33,442 Submission Gathering.ipynb
04/05/2022  05:19 AM            21,871 Subreddit Comment Gathering.ipynb
04/01/2022  04:09 PM            35,392 Subreddit Comment Gathering.pdf
04/06/2022  03:38 PM                72 Word2Vec Development.ipynb
               8 File(s)    164,474,901 bytes
               3 Dir(s)  17,820,962,816 bytes free
```

[6]:
```
c_df_1 = pd.read_csv('comments_dataframe_2.csv', index_col=[0])
```

[7]:
```
c_df_1
```

[7]:

	submission_id	comment_id
0	othlae	h6vbymj
1	othk19	h6vc0jp
2	othjhx	h6vy7kb

```
3             othjhx    h6vc1f9
4             othizk    h6vbhii
…              …        …
162468        pu4z4h    he0nyd8
162469        pu4z4h    he0jcmv
162470        pu4z4h    he0ka5j
162471        pu4z4h    he0qwgx
162472        pu4z4h    he0ruk4

                                                comment_text
0       This submission has been removed because [text…
1       This submission has been removed because [text…
2       You realize theyre making an MMO right? That r…
3       I think the problem with this event is that it…
4       Your post has been removed automatically becau…
…                                                     …
162468  Hit escape and then exit the menu and it shoul…
162469                       Had this happen  to me today
162470                                        Yess me too
162471  Yup happened to me like 5-6 times this evening…
162472  My friend also had this issue when we were pla…

[162473 rows x 3 columns]
```

[8]:
```
c_df_1.shape
```

[8]:
```
(162473, 3)
```

1.3 Cleaning

[14]:
```
df = c_df_1[c_df_1["comment_text"].str.contains("This submission has been␣
 ↪removed because")==False]
```

[16]:
```
remove = "Your post has been removed automatically"
df = df[df["comment_text"].str.contains(remove)==False]
```

[17]:
```
df
```

[17]:
```
       submission_id comment_id  \
2             othjhx    h6vy7kb
3             othjhx    h6vc1f9
5             othidx    h6vburh
6             othidx    h6vbuy6
7             othidx    h6vct3w
…              …        …
162468        pu4z4h    he0nyd8
162469        pu4z4h    he0jcmv
```

```
162470        pu4z4h    he0ka5j
162471        pu4z4h    he0qwgx
162472        pu4z4h    he0ruk4
```

	comment_text
2	You realize theyre making an MMO right? That r…
3	I think the problem with this event is that it…
5	This is simultaneously surprising and not surp…
6	Voyboy called him out on twitter actually :o \…
7	Awkward.
…	…
162468	Hit escape and then exit the menu and it shoul…
162469	Had this happen to me today
162470	Yess me too
162471	Yup happened to me like 5-6 times this evening…
162472	My friend also had this issue when we were pla…

[156550 rows x 3 columns]

[24]:
```
df = df.reset_index()
```

[25]:
```
first_comment = df.comment_text[0]
```

1.4 Preprocessing

[26]:
```
gensim.utils.simple_preprocess(first_comment)
```

[26]:
```
['you',
 'realize',
 'theyre',
 'making',
 'an',
 'mmo',
 'right',
 'that',
 'requires',
 'lore',
 'so',
 'does',
 'arcane',
 'we',
 'dont',
 'want',
 'them',
 'to',
 'stop',
 'doing',
```

```
 'lore',
 'we',
 'want',
 'them',
 'to',
 'do',
 'it',
 'right',
 'and',
 'we',
 'know',
 'they',
 'can',
 'because',
 'most',
 'stuff',
 'on',
 'universe',
 'is',
 'at',
 'the',
 'least',
 'decent']
```

[27]:
```
text_1 = df.comment_text.apply(gensim.utils.simple_preprocess)
```

[29]:
```
text_1
```

[29]:
```
0         [you, realize, theyre, making, an, mmo, right,…
1         [think, the, problem, with, this, event, is, t…
2         [this, is, simultaneously, surprising, and, no…
3         [voyboy, called, him, out, on, twitter, actual…
4                                                 [awkward]
                                …
156545    [hit, escape, and, then, exit, the, menu, and,…
156546                   [had, this, happen, to, me, today]
156547                                       [yess, me, too]
156548    [yup, happened, to, me, like, times, this, eve…
156549    [my, friend, also, had, this, issue, when, we,…
Name: comment_text, Length: 156550, dtype: object
```

1.5 Building the Word2Vec Model

We start by building a Word2Vec model that featurizes words by a window of 10.

[31]:
```
model = gensim.models.Word2Vec(
    window=10,
```

```
    min_count=2,
    workers=4
)
```

[32]:
```
model.build_vocab(text_1, progress_per=1000)
```

[33]:
```
model.epochs
```

[33]:
```
5
```

[34]:
```
model.corpus_count
```

[34]:
```
156550
```

[35]:
```
model.train(text_1, total_examples=model.corpus_count, epochs=model.epochs)
```

[35]:
```
(17861963, 22431455)
```

[36]:
```
model.save("./word2vec-minibatch-1.model")
```

Some examples of how the model featurized

[38]:
```
model.wv.most_similar("good")
```

[38]:
```
[('bad', 0.766593337059021),
 ('decent', 0.7135423421859741),
 ('great', 0.7107148766517639),
 ('solid', 0.7048998475074768),
 ('strong', 0.6690939664840698),
 ('terrible', 0.6552281975746155),
 ('hard', 0.5731886029243469),
 ('difficult', 0.5670546293258667),
 ('weak', 0.5510604977607727),
 ('tough', 0.5500995516777039)]
```

[39]:
```
model.wv.similarity(w1="ahri", w2="good")
```

[39]:
```
0.12185803
```

[40]:
```
model.wv.similarity(w1="great", w2="good")
```

[40]:
```
0.71071494
```

[41]:
```
model.wv.similarity(w1="terrible", w2="awesome")
```

[41]:
```
0.2879062
```

[43]:
```
model.wv.similarity(w1="badass", w2="jinx")
```

[43]:
```
0.4222573
```

[44]:
```
model.wv.similarity(w1="badass", w2="teemo")
```

[44]:
```
0.13311929
```

Casting the data to two dimensions for visualization purposes using t-SNE.

[52]:
```
vocab = list(model.wv.key_to_index)
X = model.wv[vocab]
```

[57]:
```
from sklearn.manifold import TSNE
```

[58]:
```
tsne = TSNE(n_components=2)
X_tsne = tsne.fit_transform(X)
```

[59]:
```
df = pd.DataFrame(X_tsne, index=vocab, columns=['x','y'])
```

[68]:
```
df.loc['the']
```

[68]:
```
x     9.080900
y   -27.367327
Name: the, dtype: float32
```

[97]:
```
df.iloc[0]
```

[97]:
```
x     9.080900
y   -27.367327
Name: the, dtype: float32
```

[98]:
```
df
```

[98]:

	x	y
the	9.080900	-27.367327
to	8.408158	-30.016670
and	8.595320	-30.189817
you	-13.109694	-42.185535
is	1.397774	-24.298342
…	…	…
ragequitters	7.224241	19.435146
rows	21.278358	9.363431
shnow	27.252644	39.208580
xmas	16.569359	13.075291
rafiitz	28.081831	34.363239

```
[27228 rows x 2 columns]
```

[71]:
```
import matplotlib.pyplot as plt
```

[96]:
```
for i in model.wv.most_similar("good"):
    print(i[0], "\t", [df.loc[i[0]]["x"], df.loc[i[0]]["y"]])
```

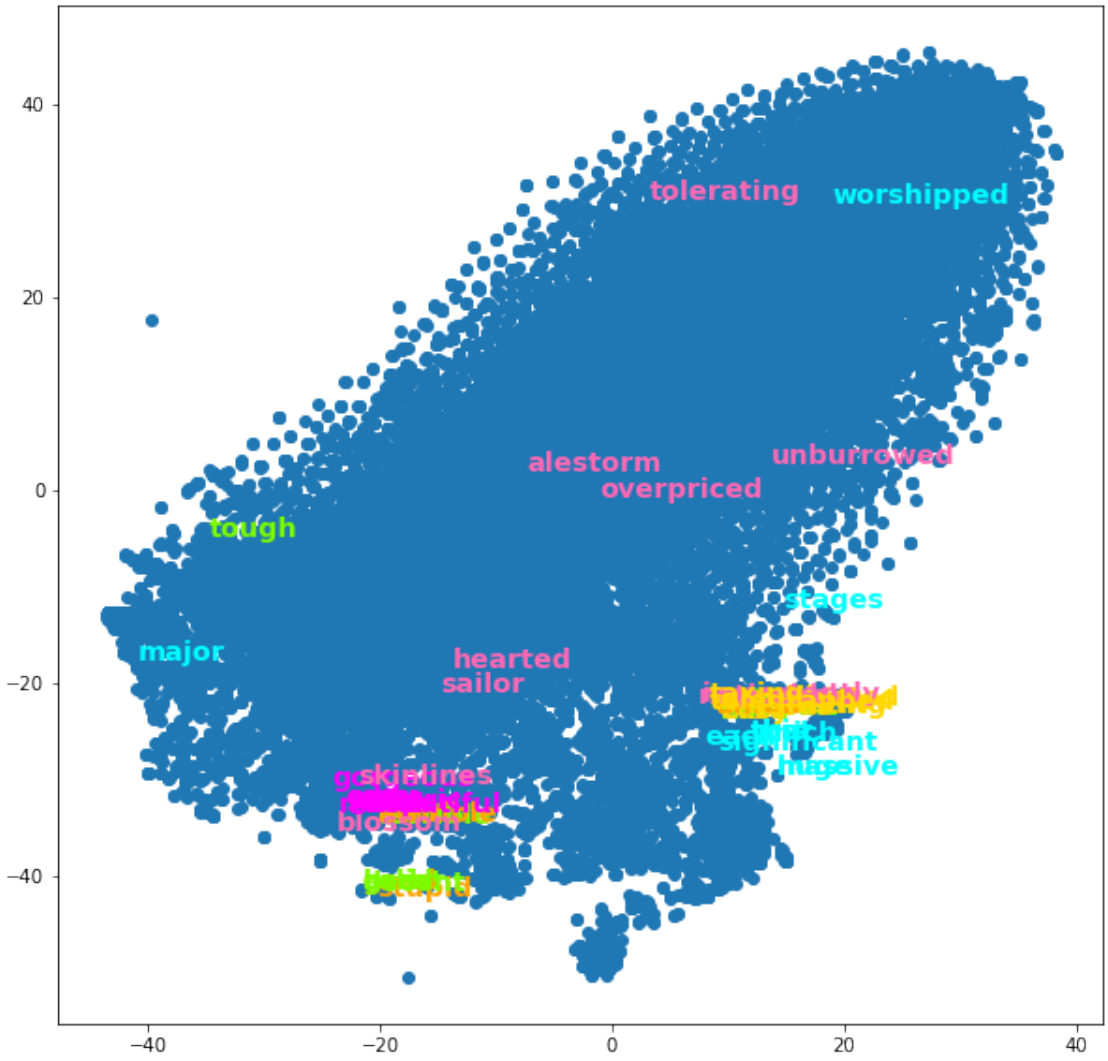
```
bad      [-21.445189, -41.40423]
decent   [-21.552238, -41.517162]
great    [-21.586025, -41.511234]
solid    [-21.572563, -41.497044]
strong   [10.614889, -22.611483]
terrible         [-20.181316, -34.638165]
hard     [9.263787, -23.758017]
difficult        [9.210935, -23.275816]
weak     [10.58605, -22.543108]
tough    [-34.813103, -4.9837813]
```

1.6 Plotting

[117]:
```
# BAD
words_similar_to_bad = model.wv.most_similar("bad")

bad_list = list()

for i in words_similar_to_bad:
    bad_list.append(i[0])

# GOOD
words_similar_to_good = model.wv.most_similar("good")

good_list = list()

for i in words_similar_to_good:
    good_list.append(i[0])

# BADASS
words_similar_to_badass = model.wv.most_similar("badass")

badass_list = list()

for i in words_similar_to_badass:
    badass_list.append(i[0])

# DIFFICULT
words_similar_to_difficult = model.wv.most_similar("difficult")
```

```
difficult_list = list()

for i in words_similar_to_difficult:
    difficult_list.append(i[0])

# THE
words_similar_to_the = model.wv.most_similar("the")

the_list = list()

for i in words_similar_to_the:
    the_list.append(i[0])

# WORSHIPPED
words_similar_to_worshipped = model.wv.most_similar("worshipped")

worshipped_list = list()

for i in words_similar_to_worshipped:
    worshipped_list.append(i[0])
```

[118]:
```
fig, ax = plt.subplots(figsize=(10,10))
#ax = fig.add_subplot(1,1,1)

ax.scatter(df['x'], df['y'])

def plot_annotate(w,p,c):
    ax.annotate(w, p, color = c, fontweight='bold', fontsize='x-large')

for word, pos in df.iterrows():
    if word in bad_list:
        plot_annotate(word,pos,'orange')
    if word in good_list:
        plot_annotate(word,pos,'lawngreen')
    if word in badass_list:
        plot_annotate(word,pos,'magenta')
    if word in difficult_list:
        plot_annotate(word,pos,'gold')
    if word in the_list:
        plot_annotate(word, pos, 'cyan')
    if word in worshipped_list:
        plot_annotate(word, pos, 'hotpink')
```



[119]:
```
plt.savefig('./window_10.png')
```

```
<Figure size 432x288 with 0 Axes>
```

2 Creating a new model

Still using the same technique as earlier, I'm experimenting how certain words ('the', 'good', 'bad', 'badass', 'worshipped') map when we change the size of the window used to build the model.

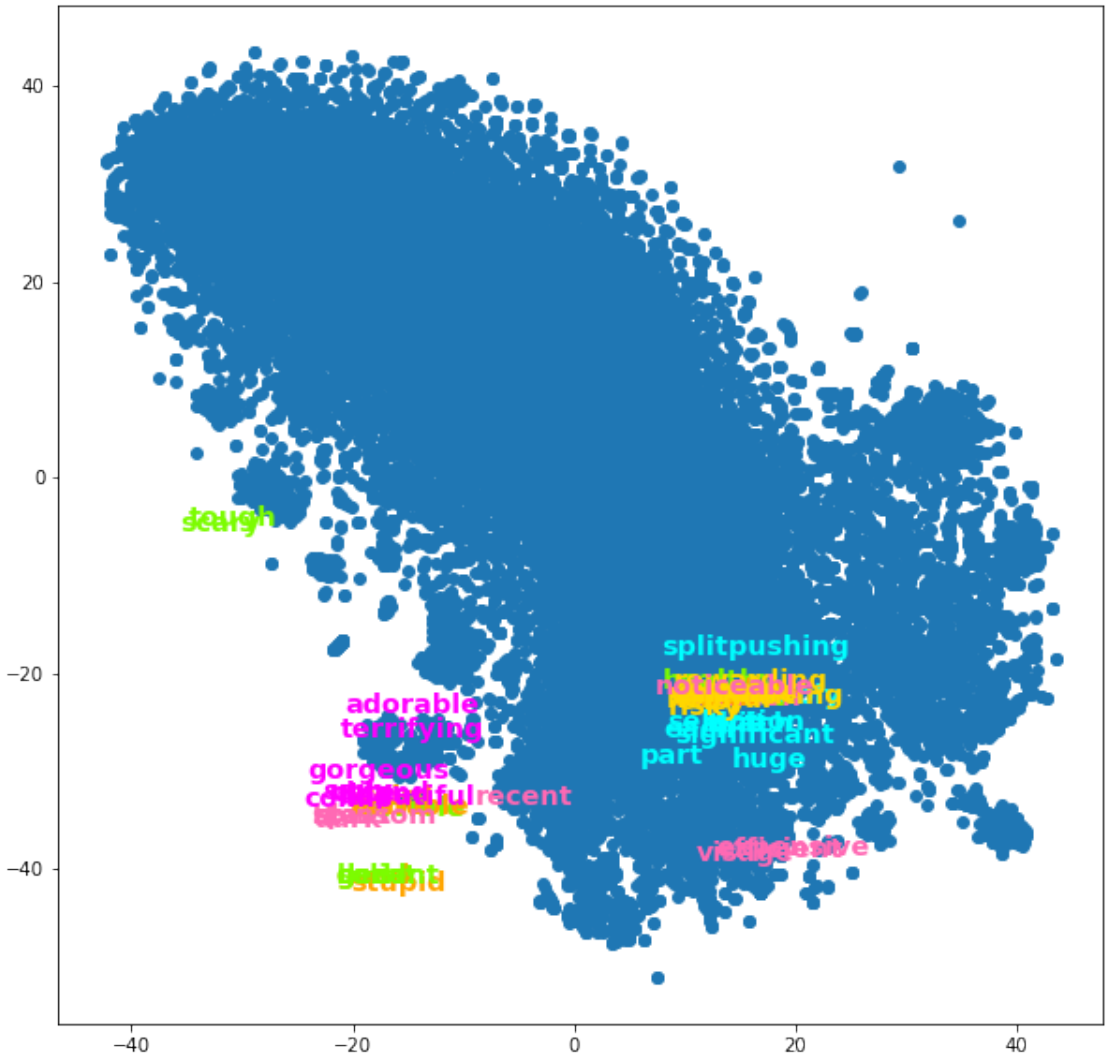
[120]:
```
model_2 = gensim.models.Word2Vec(
    window=5,
    min_count=2,
    workers=4
)
```

```
model_2.build_vocab(text_1, progress_per=1000)

model_2.train(text_1, total_examples=model_2.corpus_count, epochs=model_2.
 ↪epochs)

model_2.save("./word2vec-minibatch-1-model-2.model")

vocab_2 = list(model_2.wv.key_to_index)
X_2 = model_2.wv[vocab_2]

tsne_2 = TSNE(n_components=2)
X_tsne_2 = tsne_2.fit_transform(X_2)

df_2 = pd.DataFrame(X_tsne_2, index=vocab_2, columns=['x','y'])
```

[121]:
```
# BAD
words_similar_to_bad = model_2.wv.most_similar("bad")

bad_list = list()

for i in words_similar_to_bad:
    bad_list.append(i[0])

# GOOD
words_similar_to_good = model_2.wv.most_similar("good")

good_list = list()

for i in words_similar_to_good:
    good_list.append(i[0])

# BADASS
words_similar_to_badass = model_2.wv.most_similar("badass")

badass_list = list()

for i in words_similar_to_badass:
    badass_list.append(i[0])

# DIFFICULT
words_similar_to_difficult = model_2.wv.most_similar("difficult")

difficult_list = list()

for i in words_similar_to_difficult:
    difficult_list.append(i[0])
```

```
# THE
words_similar_to_the = model_2.wv.most_similar("the")

the_list = list()

for i in words_similar_to_the:
    the_list.append(i[0])

# WORSHIPPED
words_similar_to_worshipped = model_2.wv.most_similar("worshipped")

worshipped_list = list()

for i in words_similar_to_worshipped:
    worshipped_list.append(i[0])
```

[122]:
```
fig, ax = plt.subplots(figsize=(10,10))
#ax = fig.add_subplot(1,1,1)

ax.scatter(df_2['x'], df_2['y'])

def plot_annotate(w,p,c):
    ax.annotate(w, p, color = c, fontweight='bold', fontsize='x-large')

for word, pos in df.iterrows():
    if word in bad_list:
        plot_annotate(word,pos,'orange')
    if word in good_list:
        plot_annotate(word,pos,'lawngreen')
    if word in badass_list:
        plot_annotate(word,pos,'magenta')
    if word in difficult_list:
        plot_annotate(word,pos,'gold')
    if word in the_list:
        plot_annotate(word, pos, 'cyan')
    if word in worshipped_list:
        plot_annotate(word, pos, 'hotpink')
```



[123]:
```
plt.savefig('./window_5.png')
```

```
<Figure size 432x288 with 0 Axes>
```

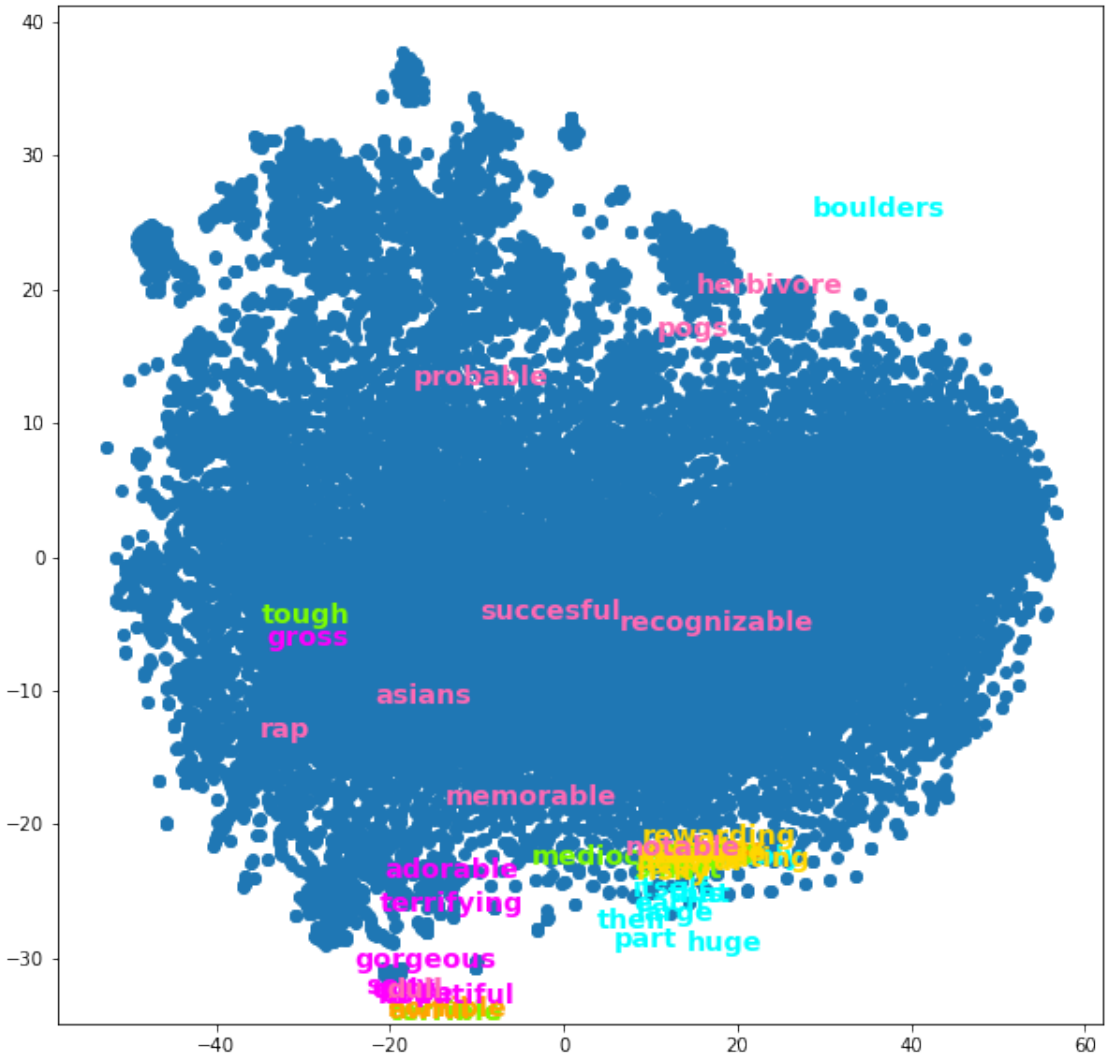
[124]:
```
model_3 = gensim.models.Word2Vec(
    window=4,
    min_count=2,
    workers=4
)

model_3.build_vocab(text_1, progress_per=1000)

model_3.train(text_1,
              total_examples=model_3.corpus_count,
```

```
                epochs=model_3.epochs)

model_3.save("./word2vec-minibatch-1-model-3.model")

vocab_3 = list(model_3.wv.key_to_index)
X_3 = model_3.wv[vocab_3]

tsne_3 = TSNE(n_components=2)
X_tsne_3 = tsne_3.fit_transform(X_3)

df_3 = pd.DataFrame(X_tsne_3, index=vocab_3, columns=['x','y'])
```

[125]:
```
# BAD
words_similar_to_bad = model_3.wv.most_similar("bad")

bad_list = list()

for i in words_similar_to_bad:
    bad_list.append(i[0])

# GOOD
words_similar_to_good = model_3.wv.most_similar("good")

good_list = list()

for i in words_similar_to_good:
    good_list.append(i[0])

# BADASS
words_similar_to_badass = model_3.wv.most_similar("badass")

badass_list = list()

for i in words_similar_to_badass:
    badass_list.append(i[0])

# DIFFICULT
words_similar_to_difficult = model_2.wv.most_similar("difficult")

difficult_list = list()

for i in words_similar_to_difficult:
    difficult_list.append(i[0])

# THE
words_similar_to_the = model_3.wv.most_similar("the")
```

```
the_list = list()

for i in words_similar_to_the:
    the_list.append(i[0])

# WORSHIPPED
words_similar_to_worshipped = model_3.wv.most_similar("worshipped")

worshipped_list = list()

for i in words_similar_to_worshipped:
    worshipped_list.append(i[0])
```

[126]:
```
fig, ax = plt.subplots(figsize=(10,10))
#ax = fig.add_subplot(1,1,1)

ax.scatter(df_3['x'], df_3['y'])

def plot_annotate(w,p,c):
    ax.annotate(w, p, color = c, fontweight='bold', fontsize='x-large')

for word, pos in df.iterrows():
    if word in bad_list:
        plot_annotate(word,pos,'orange')
    if word in good_list:
        plot_annotate(word,pos,'lawngreen')
    if word in badass_list:
        plot_annotate(word,pos,'magenta')
    if word in difficult_list:
        plot_annotate(word,pos,'gold')
    if word in the_list:
        plot_annotate(word, pos, 'cyan')
    if word in worshipped_list:
        plot_annotate(word, pos, 'hotpink')
```



[127]:
```
plt.savefig('./window_4.png')
```

```
<Figure size 432x288 with 0 Axes>
```

3 Final Comment

As of today, April 6, 2022, using just one-fifth of the data that will be available to me by tomorrow, it appears that different window sizes produce different similarity 'definitions', and by that I mean words that are similar to 'good' seem to spread more with smaller window sizes while words that are similar to 'bad' consistently group together.

My intentions as of right are to continue to use the Word2Vec model and to use other modes of visualization and clustering such as kNN clustering in combination with principal component analysis.