Photo credit: Unsplash

When Topic Modeling is Part of the Text Pre-processing

How to effectively and creatively pre-process text data

Susan Li Follow
Oct 5, 2019 · 5 min read

A few months ago, we built a content based recommender system using a relative clean text data set. Because I collected the hotel descriptions my self, I made sure that the

descriptions were useful for the goals we were going to accomplish. However, the real-world text data is never clean and there are different pre-processing ways and steps for different goals.

Topic modeling in NLP is rarely my final goal in an analysis, I use it often to either explore data or as a tool to make my final model more accurate. Let me show you what I meant.

The Data

We are still using the Seattle Hotel description data set I collected earlier, and I made it a bit more messier this time. We are going to skip all the EDA processes and I want to make recommendations as quickly as possible.

If you have read my previous post, I am sure you understand the following code script. Yes, we are looking for top 5 most similar hotels with "Hilton Garden Inn Seattle Downtown" (except itself), according to hotel description texts.

Make Recommendations

```
from nltk.corpus import stopwords
from sklearn.metrics.pairwise import linear_kernel
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import LatentDirichletAllocation
import random
import re, nltk, spacy, gensim
import pyLDAvis
import pyLDAvis.sklearn
import matplotlib.pyplot as plt
%matplotlib inline
pd.set_option('display.max_columns', 50)

df = pd.read_csv('data/Seattle_Hotels_dirty.csv', encoding="latin-1")
df.set_index('name', inplace = True)
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(df['desc'])
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)

indices = pd.Series(df.index)
```

```python
def recommendations(name, cosine_similarities = cosine_similarities):

    recommended_hotels = []

    # gettin the index of the hotel that matches the name
    idx = indices[indices == name].index[0]

    # creating a Series with the similarity scores in descending order
    score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending = False)

    # getting the indexes of the 10 most similar hotels except itself
    top_10_indexes = list(score_series.iloc[1:6].index)

    # populating the list with the names of the top 10 matching hotels
    for i in top_10_indexes:
        recommended_hotels.append(list(df.index)[i])

    return recommended_hotels

recommendations('Hilton Garden Inn Seattle Downtown')
```

dirty_hotel_rec.py hosted with ♡ by GitHub view raw

dirty_hotel_rec.py

```
['Hilton Seattle',
 "Mildred's Bed and Breakfast",
 'Seattle Airport Marriott',
 'Days Inn by Wyndham Seattle North of Downtown',
 'Holiday Inn Express & Suites North Seattle - Shoreline']
```

Figure 1

Our model returns the above 5 hotels and thinks they are top 5 most similar hotels to “Hilton Garden Inn Seattle Downtown”. I am sure you don’t agree, neither do I. Let’s say why the model thinks they are similar by looking at these descriptions.

```
df.loc['Hilton Garden Inn Seattle Downtown'].desc
```

```
'Located on the southern tip of Lake Union, the Hilton Garden Inn Seattle Downtown hotel is perfectly located for business and leisure. Non-Smoking\nHotel is 100% non-smoking, including e-cigarettes, in all guest rooms and public areas. A fee of up to $250 USD will be assessed for smoking in a non-smoking room. Please ask the Front Desk for locations of designated outdoor smoking areas. Check-in: 4:00 pm. Check-out: 12:00 pm. Cancellation policies may vary depending on the rate or dates of your reservation. Please refer to your reservation confirmation to verify your cancellation policy.\n'
```

```
df.loc["Mildred's Bed and Breakfast"].desc
```

```
'A rare find in the heart of Seattle. 100% non-smoking. Check-in: 4:00 pm. Check-out: 12:00 pm. Cancellation policies may vary depending on the rate or dates of your reservation.'
```

```
df.loc["Seattle Airport Marriott"].desc
```

```
'We streamline your travel routine. Explore the local area, brimming with alluring attractions. Reserve one of our 14 versatile event spaces for your next business meeting or wedding reception. We also feature an outdoor atrium, which provides a gorgeous backdrop for intimate gatherings. Check-in: 4:00 PM, Check-out: 12:00 PM. We are committed to providing our guests and associates with a smoke-free environment, and are proud to boast one of the most comprehensive smoke-free hotel policies in the industry. Although smoking is not permitted within hotel buildings themselves, guests who smoke are permitted to do so outside in designated areas.'
```

Found anything interesting? Yes, there are indeed somethings in common in these three hotel descriptions, they all have the same check in and check out time, and they all have the similar smoking policies. But are they important? Can we declare two hotels are similar just because they are all “non-smoking”? Of course not, these are not important characteristics and we shouldn’t measure similarity in vector space of these texts.

We need to find a way to safely remove these texts programmatically, while not removing any other useful characteristics.

Topic modeling comes to our rescue. But before that, we need to wrangle the data to make it in the right format.

- Split each description into sentences. So, for example, Hilton Garden Inn Seattle Downtown’s entire description will be split into 7 sentences.

```
df.reset_index(inplace=True)
df = pd.concat([pd.Series(str(row['name']), str(row['desc']).split('. '))
                    for _, row in df.iterrows()]).reset_index()
df.columns = ['sentence', 'name']
df['sentence'] = df['sentence'].map(lambda x: re.sub(r'\W+', ' ', x))
df.loc[df['name'] == 'Hilton Garden Inn Seattle Downtown']
```

split_desc.py hosted with ♡ by GitHub view raw

split_desc.py

	sentence	name
0	Located on the southern tip of Lake Union the ...	Hilton Garden Inn Seattle Downtown
1	Non Smoking Hotel is 100 non smoking including...	Hilton Garden Inn Seattle Downtown
2	A fee of up to 250 USD will be assessed for sm...	Hilton Garden Inn Seattle Downtown
3	Please ask the Front Desk for locations of des...	Hilton Garden Inn Seattle Downtown
4	Check in 4 00 pm	Hilton Garden Inn Seattle Downtown
5	Check out 12 00 pm	Hilton Garden Inn Seattle Downtown
6	Cancellation policies may vary depending on th...	Hilton Garden Inn Seattle Downtown
7	Please refer to your reservation confirmation ...	Hilton Garden Inn Seattle Downtown

Table 1

Topic Modeling

- We are going to build topic model for all the sentences together. I decided to have 40 topics after several experiments.

```
vectorizer = CountVectorizer(analyzer='word',
                             min_df=3,                        # minimum reqd occurences of a wor
                             stop_words='english',             # remove stop words
                             lowercase=True,                   # convert all words to lowercase
                             token_pattern='[a-zA-Z0-9]{3,}',  # num chars > 3
                             max_features=3000,             # max number of uniq words
                            )

data_vectorized = vectorizer.fit_transform(df['sentence'])

lda_model = LatentDirichletAllocation(n_components=40, # Number of topics
                                      learning_method='online',
                                      random_state=0,
                                      n_jobs = -1  # Use all available CPUs
                                     )
lda_output = lda_model.fit_transform(data_vectorized)

pyLDAvis.enable_notebook()
pyLDAvis.sklearn.prepare(lda model, data vectorized, vectorizer, mds='tsne')
```

sent_topic_model.py hosted with ♡ by GitHub view raw

sent_topic_model.py

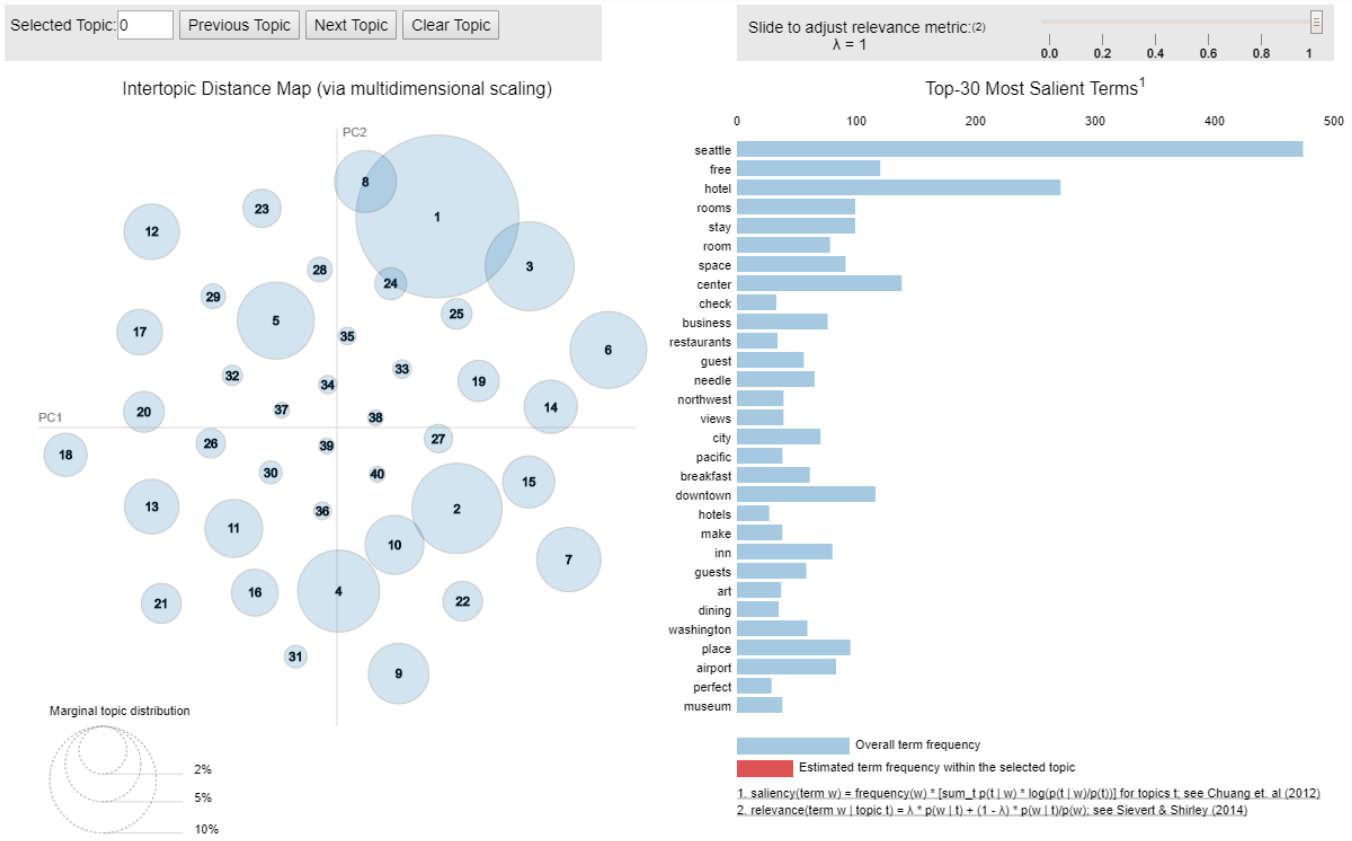

Figure 2

Not too bad, there were not too much overlapping.

The following code scripts on how to show the top keywords in each topic, and how to find the dominant topic in each document were borrowed from this excellent tutorial.

- To understand better, you may want to investigate top 20 words in each topic.

```
# Show top 20 keywords for each topic
def show_topics(vectorizer=vectorizer, lda_model=lda_model, n_words=20):
    keywords = np.array(vectorizer.get_feature_names())
    topic_keywords = []
    for topic_weights in lda_model.components_:
        top_keyword_locs = (-topic_weights).argsort()[:n_words]
        topic_keywords.append(keywords.take(top_keyword_locs))
    return topic_keywords
```

```
topic_keywords = show_topics(vectorizer=vectorizer, lda_model=lda_model, n_words=20)

# Topic - Keywords Dataframe
df_topic_keywords = pd.DataFrame(topic_keywords)
df_topic_keywords.columns = ['Word '+str(i) for i in range(df_topic_keywords.shape[1])]
df_topic_keywords.index = ['Topic '+str(i) for i in range(df_topic_keywords.shape[0])]
df_topic_keywords
```

top_20_words.py hosted with ♡ by GitHub view raw

top_20_words.py

We shall have 40 topics, and each topic shows 20 keywords. Its very hard to print out the entire table, I will only show a small part of it.

	Word 0	Word 1	Word 2	Word 3	Word 4	Word 5	Word 6	Word 7	Word 8	Word 9	Word 10	Word 11	Word 12
Topic 0	seattle	hotel	inn	lake	union	stay	experience	offers	south	guests	suites	downtown	offer
Topic 1	stay	make	need	sure	comfortable	time	road	extra	convenient	night	extended	enjoyable	offering
Topic 2	free	wifi	guests	smoke	property	parking	proud	amenities	furnished	designed	motel	airport	stylish
Topic 3	event	meeting	space	feet	square	events	meetings	000	special	planning	people	team	catering
Topic 4	smoking	non	100	areas	accessible	hotel	suite	reservation	tell	upgrade	public	club	attentive
Topic 5	business	available	free	high	access	internet	speed	center	complimentary	parking	guest	hotel	stay
Topic 6	named	gateway	grand	sheraton	provides	diverse	core	vibrant	city	pacific	located	northwest	seattle
Topic 7	views	comforts	skyline	stunning	look	scenic	home2	diamond	technology	destinations	landmarks	enjoy	like
Topic 8	accommodation	parking	beer	vary	mile	regency	coffee	studios	quiet	history	avenue	enjoy	2018
Topic 9	queen	hill	anne	come	capitol	suite	neighborhood	door	know	just	volunteer	fireplaces	distinctive
Topic 10	pool	day	outdoor	indoor	relax	heated	fitness	long	swimming	seasonal	amenities	spa	include
Topic 11	located	union	lake	downtown	seattle	leisure	reservation	perfectly	cancellation	inn	garden	hilton	hotel
Topic 12	check	12pm	years	guest	rooms	smoking	floor	apartment	mason	use	500	major	time

Table 2

By staring at the table, we can guess that at least topic 12 should be one of the topics we would like to dismiss, because it contains several words that meaningless for our purpose.

In the following code scripts, we:

- Create document-topic matrix.
- Create a data frame where each document is a row, and each column is a topic.

- The weight of each topic is assigned to each document.
- The last column is the dominant topic for that document, in which it carries the most weight.
- When we merge this data frame to the previous sentence data frame. We are able to find the the weight of each topic in every sentence, and the dominant topic for each sentence.

```python
# Create Document - Topic Matrix
lda_output = lda_model.transform(data_vectorized)

# column names
topicnames = ["Topic" + str(i) for i in range(40)]

# index names
docnames = ["Doc" + str(i) for i in range(len(data))]

# Make the pandas dataframe
df_document_topic = pd.DataFrame(np.round(lda_output, 2), columns=topicnames, index=docnames)

# Get dominant topic for each document
dominant_topic = np.argmax(df_document_topic.values, axis=1)
df_document_topic['dominant_topic'] = dominant_topic
df_document_topic.reset_index(inplace=True)
df_sent_topic= pd.merge(df, df_document_topic, left_index=True, right_index=True)
df_sent_topic.drop('index', axis=1, inplace=True)
```

sent_topic.py hosted with ♡ by GitHub view raw

sent_topic.py

- Now we can visually examine dominant topics assignment of each sentence for “Hilton Garden Inn Seattle Downtown”.

```
df_sent_topic.loc[df_sent_topic['name'] == 'Hilton Garden Inn Seattle
Downtown'][['sentence', 'dominant_topic']]
```

	sentence	dominant_topic
0	Located on the southern tip of Lake Union, the	0

0	Located on the southern tip of Lake Union the ...	0
1	Non Smoking Hotel is 100 non smoking including...	4
2	A fee of up to 250 USD will be assessed for sm...	4
3	Please ask the Front Desk for locations of des...	4
4	Check in 4 00 pm	12
5	Check out 12 00 pm	12
6	Cancellation policies may vary depending on th...	4
7	Please refer to your reservation confirmation ...	4

Table 3

- By staring at the above table, my assumption is that if a sentence's dominant topic is topic 4 or topic 12, that sentence is likely to be useless.
- Let's see a few more example sentences that have topic 4 or topic 12 as their dominant topic.

```
df_sent_topic.loc[df_sent_topic['dominant_topic'] == 4][['sentence',
'dominant_topic']].sample(20)
```

	sentence	dominant_topic
1053	The Spa at the WAC will spoil you with a compl...	4
697	Our hotel is completely non smoking	4
70	Please ask the Front Desk for locations of des...	4
7	Please refer to your reservation confirmation ...	4
23	We do not allow smoking in our rooms public ar...	4
1144	100 non smoking	4
605	Non Smoking Hotel	4
69	A fee of up to 250 USD will be assessed for sm...	4

102	The characters Attentive staff members who alw...	4
431	100 non smoking and accessible accommodations ...	4
6	Cancellation policies may vary depending on th...	4
68	Non Smoking Hotel is 100 non smoking including...	4
737	The accommodations are romantic and timeless a...	4
3	Please ask the Front Desk for locations of des...	4
1147	Cancellation policies may vary depending on th...	4
2	A fee of up to 250 USD will be assessed for sm...	4

Table 4

```
df_sent_topic.loc[df_sent_topic['dominant_topic'] == 12][['sentence',
'dominant_topic']].sample(10)
```

	sentence	dominant_topic
604	Check in 15 00 check out 11 00	12
24	Check in is at 3pm and check out is at 12pm	12
608	Check in 15 00 check out 11 00	12
78	Check in begins at 4 00pm	12
546	Check emails in the Work Zone	12
82	Late check out is subject to availability	12
1146	Check out 12 00 pm	12
4	Check in 4 00 pm	12
5	Check out 12 00 pm	12
71	Check in 4 00 pm Check out 12 00 pm	12

Table 5

- After reviewing the above two tables, I decided to remove all the sentences that have topic 4 or topic 12 as their dominant topic.

```
print('There are',
len(df_sent_topic.loc[df_sent_topic['dominant_topic'] == 4]),
'sentences that belong to topic 4 and we will remove')
print('There are',
len(df_sent_topic.loc[df_sent_topic['dominant_topic'] == 12]),
'sentences that belong to topic 12 and we will remove')
```

```
There are 20 sentences that belong to topic 4 and we will remove
There are 19 sentences that belong to topic 12 and we will remove
```

```
df_sent_topic_clean =
df_sent_topic.drop(df_sent_topic[(df_sent_topic.dominant_topic == 4)
| (df_sent_topic.dominant_topic == 12)].index)
```

- Next, we will join the clean sentence together in to a descriptions. That is, making it back to one description per hotel.

```
df_description = df_sent_topic_clean[['sentence','name']]
df_description = df_description.groupby('name')
['sentence'].agg(lambda col: ' '.join(col)).reset_index()
```

- Let's see what left for our "Hilton Garden Inn Seattle Downtown"

```
df_description['sentence'][45]
```

```
'Located on the southern tip of Lake Union the Hilton Garden Inn Seattle Downtown hotel is perfectly located for business and leisure'
```

There is only one sentence left and it is about the location of the hotel and this is what I had expected.

Make Recommendations

Using the same cosine similarity measurement, we are going to find the top 5 most similar hotels with “Hilton Garden Inn Seattle Downtown” (except itself), according to the cleaned hotel description texts.

```python
df_description.set_index('name', inplace = True)
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0, stop_words='english')
tfidf_matrix = tf.fit_transform(df_description['sentence'])
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)

indices = pd.Series(df_description.index)
def recommendations(name, cosine_similarities = cosine_similarities):

    recommended_hotels = []

    # gettin the index of the hotel that matches the name
    idx = indices[indices == name].index[0]

    # creating a Series with the similarity scores in descending order
    score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending = False)

    # getting the indexes of the 5 most similar hotels except itself
    top_10_indexes = list(score_series.iloc[1:6].index)

    # populating the list with the names of the top 5 matching hotels
    for i in top_10_indexes:
        recommended_hotels.append(list(df_description.index)[i])

    return recommended_hotels

recommendations('Hilton Garden Inn Seattle Downtown')
```

clean_desc_rec.py hosted with ♡ by GitHub view raw

clean_desc_rec.py

```
['Silver Cloud Inn - Seattle Lake Union',
 'Residence Inn by Marriott Seattle Downtown/Lake Union',
 'Staybridge Suites Seattle Downtown - Lake Union',
 'Homewood Suites by Hilton Seattle Downtown',
 'Days Inn by Wyndham Seattle North of Downtown']
```

Figure 3

Nice! Our method worked!

Jupyter notebook can be found on Github. Have a great weekend!

Data Science NLP Naturallanguageprocessing Data Preprocessing Nlp Tutorial