NLP on the Abstracts

> https://colab.research.google.com/drive/1c9H1W-vJYkXLlfaMV8kJKjB1w8B29Xm0

In [0]:

```
import pandas as pd
import matplotlib.pyplot as plt
import csv
```

In [0]:

```
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
warnings.filterwarnings("ignore", category=FutureWarning)
```

Getting the data

We import the dataset that we cleaned in the scraping notebook, and just check that it looks right.

In [0]:

```
!wget https://www.dropbox.com/s/6wcx4z3akdmqy4f/wos_df_agile.csv?dl=0
```

```
--2019-10-24 18:06:52--  https://www.dropbox.com/s/6wcx4z3akdmqy4f/wos_df_agile.csv?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.82.1, 2620:100:6032:1::a27d:5201
Connecting to www.dropbox.com (www.dropbox.com)|162.125.82.1|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/6wcx4z3akdmqy4f/wos_df_agile.csv [following]
--2019-10-24 18:06:53--  https://www.dropbox.com/s/raw/6wcx4z3akdmqy4f/wos_df_agile.csv
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://ucaf333ea7c7760407b65e3dbc9a.dl.dropboxusercontent.com/cd/0/inline/ArBZ
1Wc7dnZrWEAF7tZQCq9ArURKpe3H7G-p9r5nxawwF9CUUq43fW83YqVSOHk_muYiexLsGyOWqZoOeNcqV-SkBP57-
jx9iQMm0WQZgBXhQgt5jrx8WplqPkmvfMx5fIA/file# [following]
--2019-10-24 18:06:53--  https://ucaf333ea7c7760407b65e3dbc9a.dl.dropboxusercontent.com/c
d/0/inline/ArBZ1Wc7dnZrWEAF7tZQCq9ArURKpe3H7G-p9r5nxawwF9CUUq43fW83YqVSOHk_muYiexLsGyOWqZ
oOeNcqV-SkBP57-jx9iQMm0WQZgBXhQgt5jrx8WplqPkmvfMx5fIA/file
Resolving ucaf333ea7c7760407b65e3dbc9a.dl.dropboxusercontent.com (ucaf333ea7c7760407b65e3
dbc9a.dl.dropboxusercontent.com)... 162.125.82.6, 2620:100:6032:6::a27d:5206
Connecting to ucaf333ea7c7760407b65e3dbc9a.dl.dropboxusercontent.com (ucaf333ea7c7760407b
65e3dbc9a.dl.dropboxusercontent.com)|162.125.82.6|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5229225 (5.0M) [text/plain]
Saving to: 'wos_df_agile.csv?dl=0'

wos_df_agile.csv?dl 100%[===================>]   4.99M  11.8MB/s    in 0.4s

2019-10-24 18:06:54 (11.8 MB/s) - 'wos_df_agile.csv?dl=0' saved [5229225/5229225]
```

In [0]:

```
df = pd.read_csv('wos_df_agile.csv?dl=0',encoding='utf-8',sep='\t',index_col=0)
df.reset_index(drop=True,inplace=True)
```

In [0]:

```
print(df.shape)
df.head(2)
```

```
(901, 11)
```

Out[0]:

	Author	Title	Journal	Sub-field	Keywords	Abstract	References	City of pub	Year
0	Ettlie, JE	R&D and global manufacturing performance	MANAGEMENT SCIENCE	R&D; agility; process innovation; product inno...	LARGE MULTIPRODUCT FIRMS; D INTENSITY; DIVERSI...	R&D intensity and manufacturing performance we...	Allison P., 1990, SOCIOL METHODOL, V20, P93, D...	LINTHICUM HTS	1998.0
1	Ottaway, TA; Burns, JR	Adaptive, agile approaches to organizational a...	DECISION SCIENCES	NaN	FLEXIBLE MANUFACTURING SYSTEMS	Intelligent agent-based approaches to software...	Anthony R.N., 1965, PLANNING CONTROL SYS; BUZA...	ATLANTA	1997.0

In [0]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 901 entries, 0 to 900
Data columns (total 11 columns):
Author          901 non-null object
Title           901 non-null object
Journal         901 non-null object
Sub-field       834 non-null object
Keywords        778 non-null object
Abstract        901 non-null object
References      901 non-null object
City of pub     901 non-null object
Year            895 non-null float64
Field           901 non-null object
DOI             901 non-null object
dtypes: float64(1), object(10)
memory usage: 77.6+ KB
```

Preprocessing - tokinization

In [0]:

```
import spacy
nlp = spacy.load("en")


import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

First we clean the abstracts, by tokenizing them, lowercasing the words, removing numbers and then joining the tokens to one string again.

In [0]:

```
df['tokens'] = df['Abstract'].map(lambda t: word_tokenize(t))
df['clean_tokens'] = df['tokens'].map(lambda t: [token.lower() for token in t if not tok
en.isdigit()]) #.isalpha???
df['clean_Abstract'] = df['clean_tokens'].map(lambda t: ' '.join(t)) #join each list usi
ng space inbetween the tokens
```

In [0]:

```
print(df.Abstract[10])
```

```
print(df.clean_Abstract[10])
```

```
Virtual corporations, enterprise re-engineering, and adaptive/agile manufacturing are all
new concepts based on the accomplishments of integrated manufacturing of the past decade.
The new manufacturing enterprises are characterized by an ability to effect flexible reco
nfiguration of resources, shorter cycle times and quick response to customer demands. Inf
ormation is a key factor in transcending physical barriers and imparting the enterprise-o
riented agility and adaptiveness to organizations. To this end, a theory-based reference
model for information integration is needed in manufacturing enterprises. Employs the par
adigm of parallel formulation as the reference model and demonstrates how it is used to g
uide the planning for information integration. The model provides both a detailed data an
d task analysis of manufacturing functions and their interactions, and guidelines for reg
rouping tasks into parallel processes and thereby achieving a high level of global integr
ation. Describes a case study of the model, conducted on the existing CIM model at Rensse
laer to evaluate and reformulate the previous processes. The results show a better design
featuring concurrent execution of functions which in turn support agility and adaptivenes
s.
virtual corporations , enterprise re-engineering , and adaptive/agile manufacturing are a
ll new concepts based on the accomplishments of integrated manufacturing of the past deca
de . the new manufacturing enterprises are characterized by an ability to effect flexible
reconfiguration of resources , shorter cycle times and quick response to customer demands
. information is a key factor in transcending physical barriers and imparting the enterpr
ise-oriented agility and adaptiveness to organizations . to this end , a theory-based ref
erence model for information integration is needed in manufacturing enterprises . employs
the paradigm of parallel formulation as the reference model and demonstrates how it is us
ed to guide the planning for information integration . the model provides both a detailed
data and task analysis of manufacturing functions and their interactions , and guidelines
for regrouping tasks into parallel processes and thereby achieving a high level of global
integration . describes a case study of the model , conducted on the existing cim model a
t rensselaer to evaluate and reformulate the previous processes . the results show a bett
er design featuring concurrent execution of functions which in turn support agility and a
daptiveness .
```

Then we use spacy to retokenize and lemmatize the words, remove stopwords and remove words based on their POS (and puncuation).

In [0]:

```
preprocessed_toks = []

for Ab in nlp.pipe(df['clean_Abstract']):
  Ab_tok = [token.lemma_ for token in Ab if token.pos_ in ['NOUN', 'PROPN', 'ADJ', 'ADV'
, 'VERB'] and not token.is_stop]
  preprocessed_toks.append(Ab_tok)
```

We now save these filtered tokens as a new collumn in the dataframe.

In [0]:

```
df['filtered_tokens_final'] = preprocessed_toks
print(df.Abstract[10])
print(df.filtered_tokens_final[10])
```

```
Virtual corporations, enterprise re-engineering, and adaptive/agile manufacturing are all
new concepts based on the accomplishments of integrated manufacturing of the past decade.
The new manufacturing enterprises are characterized by an ability to effect flexible reco
nfiguration of resources, shorter cycle times and quick response to customer demands. Inf
ormation is a key factor in transcending physical barriers and imparting the enterprise-o
riented agility and adaptiveness to organizations. To this end, a theory-based reference
model for information integration is needed in manufacturing enterprises. Employs the par
adigm of parallel formulation as the reference model and demonstrates how it is used to g
uide the planning for information integration. The model provides both a detailed data an
d task analysis of manufacturing functions and their interactions, and guidelines for reg
rouping tasks into parallel processes and thereby achieving a high level of global integr
ation. Describes a case study of the model, conducted on the existing CIM model at Rensse
laer to evaluate and reformulate the previous processes. The results show a better design
featuring concurrent execution of functions which in turn support agility and adaptivenes
s.
['virtual', 'corporation', 'enterprise', '-', 'engineering', 'adaptive', 'agile', 'manufa
cturing', 'new', 'concept', 'base', 'accomplishment', 'integrate', 'manufacturing', 'past
', 'decade', 'new', 'manufacturing', 'enterprise', 'characterize', 'ability', 'effect', '
```

```
, 'decade', 'new', 'manufacturing', 'enterprise', 'characterize', 'ability', 'effect',
flexible', 'reconfiguration', 'resource', 'short', 'cycle', 'time', 'quick', 'response',
'customer', 'demand', 'information', 'key', 'factor', 'transcend', 'physical', 'barrier',
'impart', 'enterprise', 'orient', 'agility', 'adaptiveness', 'organization', 'end', 'theo
ry', 'base', 'reference', 'model', 'information', 'integration', 'need', 'manufacturing',
'enterprise', 'employ', 'paradigm', 'parallel', 'formulation', 'reference', 'model', 'dem
onstrate', 'guide', 'planning', 'information', 'integration', 'model', 'provide', 'detail
ed', 'datum', 'task', 'analysis', 'manufacturing', 'function', 'interaction', 'guideline'
, 'regroup', 'task', 'parallel', 'process', 'achieve', 'high', 'level', 'global', 'integr
ation', 'describe', 'case', 'study', 'model', 'conduct', 'exist', 'cim', 'model', 'rensse
laer', 'evaluate', 'reformulate', 'previous', 'process', 'result', 'well', 'design', 'fea
ture', 'concurrent', 'execution', 'function', 'turn', 'support', 'agility', 'adaptiveness
']
```

Bag of Word and TF-IDF

Now we use Gensim to build a dictionary from the filtered tokens, then make a Bag of Words (corpus), and then use TF-IDF to normalize the corpus.

In [0]:

```
from gensim.corpora.dictionary import Dictionary
from gensim.models.tfidfmodel import TfidfModel
```

In [0]:

```
dictionary = Dictionary(df['filtered_tokens_final'])
dictionary.filter_extremes(no_below=5, no_above=0.5, keep_n=1000) # Here wefilter out low
-frequency / high-frequency tokens, also limit the vocabulary to max 1000 words
corpus = [dictionary.doc2bow(doc) for doc in df['filtered_tokens_final']]

corpus[0][:5] # check the top of the BOW corpus for the first first abstract
```

Out[0]:

```
[(0, 2), (1, 3), (2, 1), (3, 1), (4, 2)]
```

In [0]:

```
dictionary.get(1) #And we can see that the dictionary can still be used for indexing
```

Out[0]:

```
'agility'
```

In [0]:

```
tfidf = TfidfModel(corpus) # Fitting TfidfModel
tfidf_corpus = tfidf[corpus] # Transforming the corpus

tfidf_corpus[0][:5] # check the top of the TFIDF corpus for the first first abstract
```

Out[0]:

```
[(0, 0.26339842054487783),
 (1, 0.0751627574961822),
 (2, 0.10458264636401556),
 (3, 0.08674838487902369),
 (4, 0.15982080928478185)]
```

Dimensionality reduction

Here we use Gensims LSI to do dimensionality reduction (and maybe find topics)

In [0]:

```
from gensim.models.lsimodel import LsiModel
```

In [0]:

```
# We fit the LSI model on the tfidf_corpus, giving the dictionary as reference and choosi
ng the number of topics.
lsi = LsiModel(tfidf_corpus, id2word=dictionary, num_topics=100) # In more serious settin
gs one would pick between 300-400
lsi.show_topics(num_topics=10)
```

Out[0]:

```
[(0,
  '0.239*"supply" + 0.233*"chain" + 0.142*"project" + 0.135*"agility" + 0.127*"firm" + 0.
121*"performance" + 0.110*"business" + 0.107*"capability" + 0.101*"strategy" + 0.099*"mod
el"'),
 (1,
  '-0.480*"project" + 0.469*"supply" + 0.462*"chain" + -0.208*"team" + -0.187*"software"
+ -0.133*"development" + -0.102*"agile" + -0.093*"innovation" + 0.080*"performance" + -0.
074*"system"'),
 (2,
  '-0.574*"project" + -0.351*"supply" + -0.345*"chain" + -0.172*"software" + -0.168*"team
" + 0.159*"capability" + 0.159*"business" + 0.150*"organizational" + 0.138*"service" + 0.
111*"strategic"'),
 (3,
  '-0.294*"lean" + 0.269*"project" + 0.250*"capability" + -0.218*"manufacturing" + 0.210*
"firm" + 0.183*"organizational" + -0.177*"system" + 0.151*"agility" + -0.144*"service" +
0.139*"team"'),
 (4,
  '-0.395*"service" + 0.326*"sc" + 0.288*"lean" + 0.269*"manufacturing" + 0.184*"performa
nce" + -0.157*"humanitarian" + -0.154*"supply" + -0.152*"chain" + 0.139*"project" + -0.12
7*"team"'),
 (5,
  '-0.594*"team" + 0.435*"service" + 0.374*"project" + -0.199*"software" + -0.160*"leader
ship" + -0.140*"development" + 0.105*"e" + 0.098*"customer" + 0.095*"innovation" + -0.082
*"work"'),
 (6,
  '0.517*"service" + 0.427*"team" + -0.217*"humanitarian" + 0.165*"sc" + 0.152*"performan
ce" + 0.123*"software" + -0.121*"strategic" + 0.121*"customer" + 0.115*"lean" + 0.109*"e"
'),
 (7,
  '-0.544*"sc" + -0.282*"humanitarian" + 0.193*"product" + -0.180*"service" + 0.177*"firm
" + 0.167*"innovation" + 0.140*"manufacturing" + -0.133*"logistic" + 0.113*"flexibility"
+ 0.099*"market"'),
 (8,
  '-0.521*"innovation" + -0.403*"sc" + -0.302*"product" + 0.156*"manufacturing" + -0.149*
"strategy" + 0.126*"system" + 0.106*"agility" + -0.106*"new" + 0.105*"software" + 0.101*"
service"'),
 (9,
  '0.418*"sc" + -0.361*"lean" + -0.347*"innovation" + -0.221*"humanitarian" + 0.206*"deci
sion" + 0.171*"flexibility" + -0.141*"sme" + -0.119*"manufacturing" + 0.108*"product" + -
0.106*"capability"')]
```

We see some differences in the topics. For example, the first topic (0) seems to revolve around supply chain management, while the third topic (3) seems to revolve more around manufactoruring capabilities. Yet another topic (8) seems to be primarily about innovation.

In [0]:

```
lsi_corpus = lsi[tfidf_corpus] # We use the trained model to transform the corpus
```

Lastly we use Gensims MatrixSimilarity to create a document-topic matrix

In [0]:

```
from gensim.similarities import MatrixSimilarity
```

In [0]:

```
document_topic_matrix = MatrixSimilarity(lsi_corpus)
document_topic_matrix_ix = document_topic_matrix.index #The document_topic_matrix_ix is t
he LSA feature matrix lsi_corpus is a sparse version
```

In [0]:

```
# We make it in to a dataframe just to make it easier to display
df_doc_topic=pd.DataFrame(data=document_topic_matrix_ix,index=['Doc '+str(i) for i in df
.index.values],columns=['Topic '+str(i) for i in range(document_topic_matrix_ix.shape[1]
)])
df_doc_topic.head(3)
```

Out[0]:

	Topic 0	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10	Topic 11
Doc 0	0.274239	-0.028396	0.106275	0.029585	0.202943	-0.003040	0.068265	0.216463	0.036163	-0.040635	0.107368	-0.049177
Doc 1	0.211348	-0.121450	-0.006315	-0.064819	-0.103745	-0.120349	0.014671	0.021305	0.086019	0.154983	0.031784	0.147042
Doc 2	0.357682	-0.203855	0.160394	-0.023037	-0.089853	-0.026624	0.011730	0.156173	-0.295065	-0.165111	-0.070601	0.133722

3 rows × 100 columns

Word Embedding

Here we use Word2Vec form Gensim to make a word embedding

In [0]:

```
from gensim.models import Word2Vec
import gensim
```

In [0]:

```
W2V = Word2Vec(df['clean_tokens'], size=100, window=8, min_count=3, workers=4, iter=10)
# Training the language model
```

In [0]:

```
W2V.most_similar('culture') # Did the model training  work?
```

Out[0]:

```
[('enforced', 0.9607012867927551),
 ('vision', 0.9600250124931335),
 ('facilitates', 0.9582845568656921),
 ('roles', 0.9570927023887634),
 ('activation', 0.9542210102081299),
 ('relational', 0.9513115882873535),
 ('collectivism', 0.9490099549293518),
 ('diffusion', 0.9470982551574707),
 ('complementarities', 0.9404903650283813),
 ('process-oriented', 0.9393550157546997)]
```

Here we see that the model it good at finding similar words!!

EDA - Count based

So here we want to see if we can find differences in the words that are used in the abstracts in articles from different fields.

In [0]:

```
from collections import Counter
import itertools
```

First we choose some Fields we would like to compare. And we check how many tokens we have for each Field.

In [0]:

```
field_0 = list(itertools.chain(*df[df['Field'] == 'Business; Management']['filtered_tokens_final']))
field_1 = list(itertools.chain(*df[df['Field'] == 'Business; Psychology, Applied; Management']['filtered_tokens_final']))
field_2 = list(itertools.chain(*df[df['Field'] == 'Management; Social Sciences, Mathematical Methods']['filtered_tokens_final']))

counter_0 = Counter(field_0)
counter_1 = Counter(field_1)
counter_2 = Counter(field_2)

print(len(field_0))
print(len(field_1))
print(len(field_2))
```

```
17392
279
60
```

Then we print the 20 most used words in each Field

In [0]:

```
print('Top words Field 0')
print(counter_0.most_common(20))
print('Top words Field 1')
print(counter_1.most_common(20))
print('Top words Field 2')
print(counter_2.most_common(20))
```

```
Top words Field 0
[('supply', 317), ('chain', 281), ('research', 193), ('agility', 192), ('study', 187), ('paper', 164), ('approach', 155), ('firm', 154), ('model', 152), ('management', 149), ('business', 145), ('performance', 137), ('strategy', 124), ('purpose', 123), ('value', 121), ('agile', 112), ('design', 106), ('finding', 106), ('relationship', 101), ('implication', 101)]
Top words Field 1
[('team', 14), ('change', 9), ('global', 9), ('organization', 8), ('new', 6), ('work', 6), ('management', 4), ('environment', 4), ('agile', 4), ('study', 4), ('address', 4), ('provide', 4), ('meta', 4), ('complex', 3), ('model', 3), ('process', 3), ('demonstrate', 3), ('mne', 3), ('challenge', 2), ('demand', 2)]
Top words Field 2
[('research', 3), ('system', 3), ('discuss', 2), ('paper', 2), ('award', 2), ('field', 2), ('work', 2), ('dynamic', 2), ('genesis', 1), ('forrester', 1), ('give', 1), ('summarize', 1), ('finding', 1), ('context', 1), ('stream', 1), ('embed', 1), ('describe', 1), ('effect', 1), ('subsequent', 1), ('briefly', 1)]
```

As we can see from the results, the focus of the three study fields related to our theme 'business agility' is different for each of them.

Field 0: indicates agility from the perspective of the value chain (the total stakeholders that contribute to the production of a finished good/service and add value along the way). We can see terms such as 'supply','chain','design' and'relationship'.

Field 1: presents agility from the persepective of the economic actor and, since organizational agility is a relatively new concept, we can see terms such as 'change' and 'management', indicating at the changes a company needs to undertake in order to become agile.

Field 2: is perhaps the most unclear. The main keywords point out an academic discussion of a macro/industry focus. The 'systemic' view of the innovation system describes relationships between economic actors in an industry, whose collective actions determine the industry to move furward.

Unsupervised Learning - K-means clustering and HDBSCAN

K-means

Though looking at unsupervised methods such as LSI that show groupings of abstract topics, on a word-based representation, it could now be beneficial to get a visual representation of clusterings. Therefore, will the final step be to unsupervised clustering methods

From getting the document-topic matrix earlier on from LSI, we can now perform K-means clustering - the most common way of clustering, using k-nearest neighbors. The purpose is to see whether distinguishable clusters are formed of semantically similar topics within the article abstracts.

In [0]:

```
#Let's figure out how many clusters really add explaninability..

from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
km_scores= []
km_silhouette = []

#We'll create a for loop to both calculate and predict scores.
#Because, we suspect that a high number of clusters are needed, we'll explore up to '30'
clusters.
for i in range(2,30):
    km = KMeans(n_clusters=i, random_state=0).fit(document_topic_matrix_ix)
    preds = km.predict(document_topic_matrix_ix)

    km_scores.append(-km.score(document_topic_matrix_ix))

    silhouette = silhouette_score(document_topic_matrix_ix,preds)
    km_silhouette.append(silhouette)
```

Now that we have the silhouette scores, we can plot them using the 'KElbowVisualizer'.

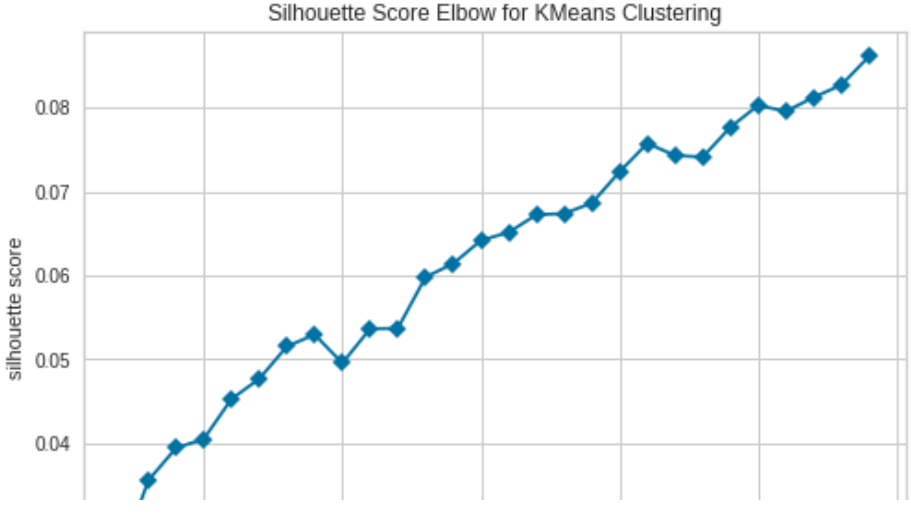
In [0]:

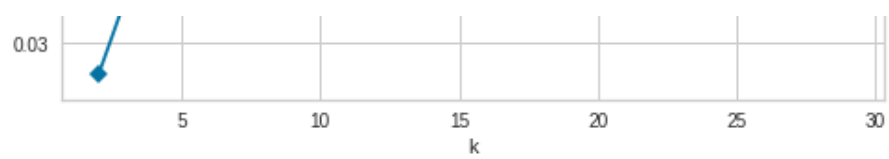
```
# Import the KElbowVisualizer method
from yellowbrick.cluster import KElbowVisualizer

# Instantiate a scikit-learn K-Means model
model = KMeans(random_state=0)

# Instantiate the KElbowVisualizer with the number of clusters and the metric
visualizer = KElbowVisualizer(model, k=(2,30), metric='silhouette', timings=False)

# Fit the data and visualize
visualizer.fit(document_topic_matrix_ix)
visualizer.poof()
```





We see that unfortunately, our abstract words are not explained by clustering well, unless an unreasonable amount of clusters are used. However, we do see point around k=15, where circa 5 more clusters do not add value. So, the best 'elbow point' seems to be 15.

With this information in mind, let's try to compute K-means clustering with 15 as the number of clusters.

In [0]:

```
from sklearn.cluster import KMeans
clusterer = KMeans(n_clusters = 15)
clusterer.fit(document_topic_matrix_ix)
```

Out[0]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=15, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In order to subsequently plot the clusters, we need to do dimensionalility reduction of our document-topic matrix. 'UMAP' is appropriate for this.

In [0]:

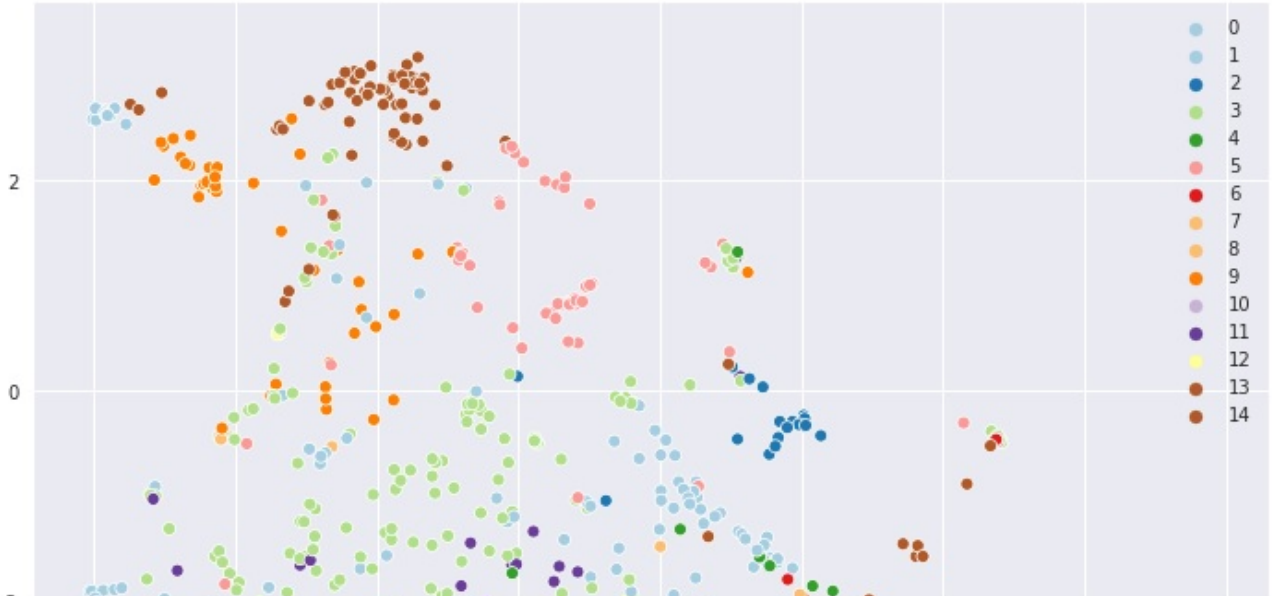
```
#Dimensionalility reduction for plotting
import umap
embeddings = umap.UMAP(n_neighbors=15, metric='cosine').fit_transform(document_topic_matr
ix_ix)
```

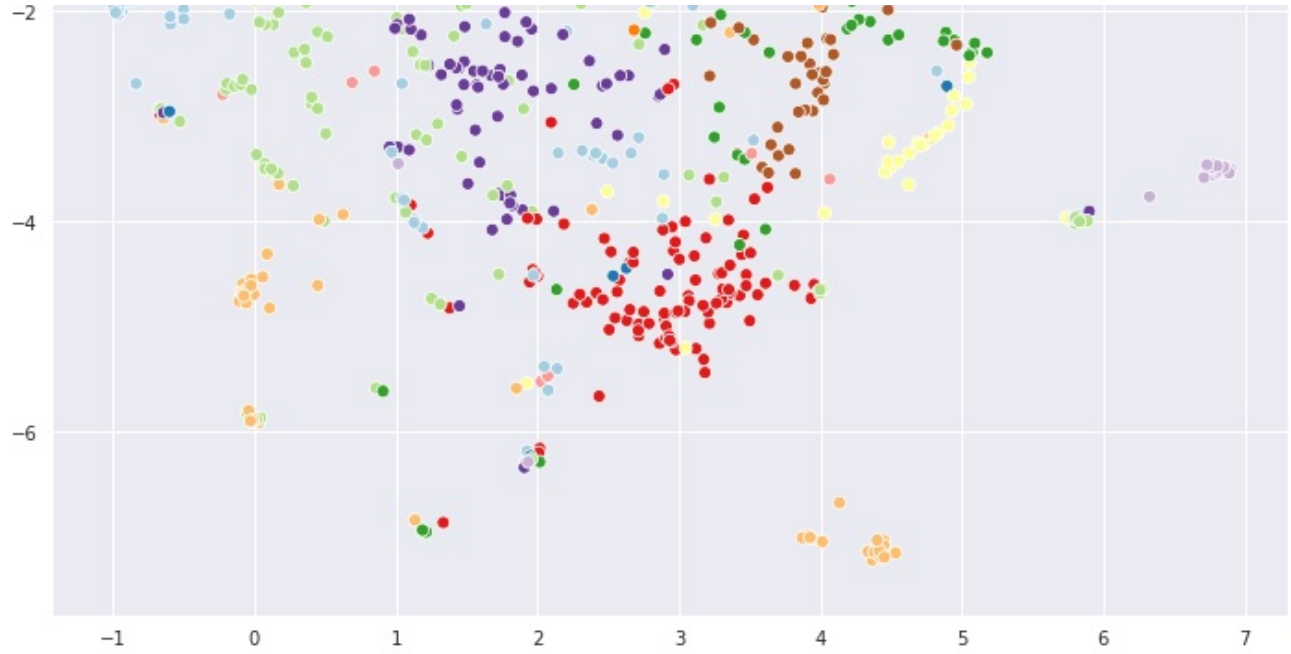
Finally, it's time to plot the clusterings.

In [0]:

```
#Seaborn is used for this
import seaborn as sns
sns.set_style("darkgrid")

plt.rcParams.update({'font.size': 12})
plt.figure(figsize=(12,12))
g = sns.scatterplot(*embeddings.T,
                    #reduced[:,0],reduced[:,1],
                    hue=clusterer.labels_,
                     palette="Paired",
                    legend='full')
```





From the K-means clustering we can see that there, while overlapping, are quite a few abstract topics that tend to form somewhat clear clusters

In [0]:

```
# Let's explore the clusters ... that should actually correlate with topics found by LDA
df['cluster'] = clusterer.labels_
```

In [0]:

```
df[df['cluster'] == 0]['filtered_tokens_final'].head(10)
```

Out[0]:

```
0     [d, intensity, manufacturing, performance, eva...
3     [industrial, management, 1980, mark, end, twen...
6     [corporation, compete, global, marketplace, ti...
8     [exploratory, empirical, study, modernize, dur...
9     [deal, agile, manufacturing, new, paradigm, or...
10    [virtual, corporation, enterprise, -, engineer...
11    [base, review, current, manufacturing, trend, ...
39    [industrial, practice, experience, highlight, ...
42    [paper, present, indicator, base, multi, -, ob...
46    [today, complex, dynamic, supply, chain, marke...
Name: filtered_tokens_final, dtype: object
```

In [0]:

```
lsi.show_topics(5)[4]
```

Out[0]:

```
(4,
 '-0.395*"service" + 0.326*"sc" + 0.288*"lean" + 0.269*"manufacturing" + 0.184*"performan
ce" + -0.157*"humanitarian" + -0.154*"supply" + -0.152*"chain" + 0.139*"project" + -0.127
*"team"')
```

The words in topic 4 from lsi seems somewhat similar to the words in cluster 0 made by k-means. (manufacturing,performance) But looking through them it is diffifult to see any clear correspondane.

HDBSCAN

K-means clustering has however proved not to be the best for data with many clusters small in size. We subsequently want to see if we can produce more distinguishable clusters using HDBSCAN, to filter out some of the noise that we saw in the K-means plot.

In [0]:

```
pip install hdbscan #install the hdbscan
```

```
Collecting hdbscan
  Downloading https://files.pythonhosted.org/packages/18/35/a117860dd7f38eb1a77e4e30ac69a
63c45c5a6356f503874fbe80fc464b7/hdbscan-0.8.23.tar.gz (4.9MB)
     |████████████████████████████████| 4.9MB 2.8MB/s
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
    Preparing wheel metadata ... done
Requirement already satisfied: cython>=0.27 in /usr/local/lib/python3.6/dist-packages (fr
om hdbscan) (0.29.13)
Requirement already satisfied: joblib in /usr/local/lib/python3.6/dist-packages (from hdb
scan) (0.14.0)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.6/dist-packages (f
rom hdbscan) (1.17.3)
Requirement already satisfied: scipy>=0.9 in /usr/local/lib/python3.6/dist-packages (from
hdbscan) (1.3.1)
Requirement already satisfied: scikit-learn>=0.17 in /usr/local/lib/python3.6/dist-packag
es (from hdbscan) (0.21.3)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from hdbsca
n) (1.12.0)
Building wheels for collected packages: hdbscan
  Building wheel for hdbscan (PEP 517) ... done
  Created wheel for hdbscan: filename=hdbscan-0.8.23-cp36-cp36m-linux_x86_64.whl size=233
3650 sha256=922b1fc7b696242f9dc7e11b8546715f808f06ed6c751d97846250b79f0a8f5a
  Stored in directory: /root/.cache/pip/wheels/1e/2a/69/0cdd54146243126667l8f7526b0ab39ca
d65cea1c1b3892768
Successfully built hdbscan
Installing collected packages: hdbscan
Successfully installed hdbscan-0.8.23
```

In [0]:

```
import hdbscan
```

Instead of directly feeding our document-topic matrix, we can convert it into a distance matrix, which the HDBSCAN has a better time removing noise with.

In [0]:

```
from sklearn.metrics.pairwise import pairwise_distances
distance_matrix = pairwise_distances(document_topic_matrix_ix)

#Here, we define our HDBSCAN clustering algorithm.
clusterer = hdbscan.HDBSCAN(min_cluster_size=9, #We set the minimum size for clusters to
be 8.
                            cluster_selection_method='leaf',
                            leaf_size=40,
                            prediction_data=True)

clusterer.fit(distance_matrix)
```

Out[0]:

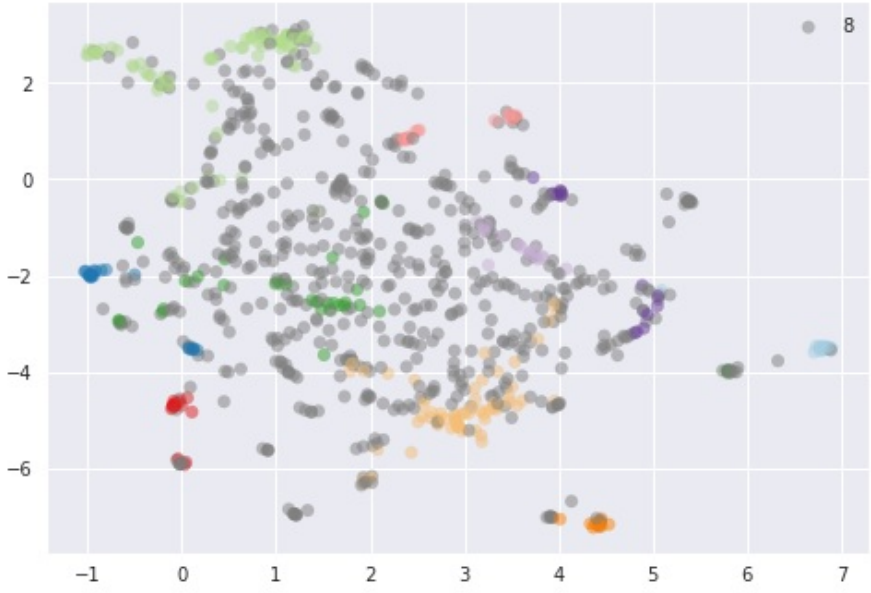
```
HDBSCAN(algorithm='best', allow_single_cluster=False, alpha=1.0,
        approx_min_span_tree=True, cluster_selection_method='leaf',
        core_dist_n_jobs=4, gen_min_span_tree=False, leaf_size=40,
        match_reference_implementation=False, memory=Memory(location=None),
        metric='euclidean', min_cluster_size=9, min_samples=None, p=None,
        prediction_data=True)
```

In [0]:

```
color_palette = sns.color_palette('Paired', 12)
cluster_colors = [color_palette[x] if x >= 0
                  else (0.5, 0.5, 0.5)
                  for x in clusterer.labels_]
cluster_member_colors = [sns.desaturate(x, p) for x, p in
```

```
                          zip(cluster_colors, clusterer.probabilities_)]
plt.scatter(*embeddings.T, s=50, linewidth=0, c=cluster_member_colors, alpha=0.5)
plt.legend(model.labels_)

plt.show()
```



We already see that the filtering out of noise through clusters with few words provided more clear clusters in the plot, now lest look at the actual words:

In [0]:

```
#We can inspect the most common words within each cluster - here we do it just for 3 clus
ters
import itertools
counter0 = Counter(list(itertools.chain(*df[df.cluster == 0]['filtered_tokens_final'])))
counter1 = Counter(list(itertools.chain(*df[df.cluster == 1]['filtered_tokens_final'])))
counter2 = Counter(list(itertools.chain(*df[df.cluster == 2]['filtered_tokens_final'])))
```

In [0]:

```
print(counter0.most_common(10))
print(counter1.most_common(10))
print(counter2.most_common(10))
```

```
[('manufacturing', 183), ('system', 135), ('study', 115), ('agile', 99), ('paper', 84), (
'business', 83), ('company', 83), ('model', 80), ('agility', 77), ('network', 77)]
[('team', 137), ('leadership', 90), ('leader', 45), ('study', 30), ('agility', 29), ('mod
el', 27), ('organization', 27), ('research', 26), ('work', 25), ('approach', 24)]
[('decision', 154), ('make', 66), ('approach', 44), ('portfolio', 39), ('system', 38), ('
study', 37), ('paper', 36), ('process', 36), ('management', 31), ('model', 30)]
```

Here we see some really nice and meaningfull clusters, especialy cluster 2 with these words: Team, leadership, leader, organization. But also cluster 3: Decision, approch, process.