Extraction-Based Summarization Demo

DATA 5420/6420

Name: Dallin Moore

In this demonstration I will cover usage of the TextRank algorithm for performing automatic extraction-based document summarization. This is a method that is useful for shortening very long documents to a set number of "top" sentences that serve to summarize a document.

For this example, we will first work with the transcript of my last talk, which was a hours worth of talking, and summarize it down to a few sentences. Then, we'll examine another source that's a bit more coherent and contains less filler -- a Wikipedia explanation of automated summarization.

As always, let's start by bringing in our dependencies:

```
!pip install contractions
```

```
Collecting contractions
  Downloading contractions-0.1.73-py2.py3-none-any.whl (8.7 kB)
Collecting textsearch>=0.0.21 (from contractions)
  Downloading textsearch-0.0.24-py2.py3-none-any.whl (7.6 kB)
Collecting anyascii (from textsearch>=0.0.21->contractions)
  Downloading anyascii-0.3.2-py3-none-any.whl (289 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 289.9/289.9 kB 3.4 MB/s eta 0:00:00
Collecting pyahocorasick (from textsearch>=0.0.21->contractions)
  Downloading pyahocorasick-2.1.0-cp310-cp310-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (110
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 110.7/110.7 kB 4.2 MB/s eta 0:00:00
Installing collected packages: pyahocorasick, anyascii, textsearch, contractions
Successfully installed anyascii-0.3.2 contractions-0.1.73 pyahocorasick-2.1.0 textsearch-0.0.24
```

```
import re
import nltk
import networkx as nx

from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.corpus import stopwords
from nltk.cluster.util import cosine_distance

import contractions


nltk.download('punkt')
nltk.download('stopwords')

import matplotlib.pyplot as plt
import pandas as pd

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

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

Load in the transcript file

```
# Load transcript from file
transcript_file = '/content/transcript.txt'
with open(transcript_file, 'r') as file:
    transcript_text = file.read()


transcript_text
```

'\ufeff0:4:2.830 --> 0:4:5.260\nCarly Fox\nAlright then I guess I'll go ahead and get started.\n0:4:7.350 --> 0:4:14.890\nCarly Fox\nAnd today, we're going to be talking about topic modeling, which is a pretty cool unsupervised method.\n0:4:14.930 --> 0:4:16.340\nCarly Fox\nDefinitely use it pretty frequently.\n0:4:18.720 --> 0:4:29.460\nCarly Fox\nSo as always, we'll start off with a brief lecture component on topic modeling, and then we'll do a demonstration in Python afterwards.\n0:4:29.470 --> 0:4:32.450\nCarly Fox\nSo go ahead and.\n0:4:39.170 --> 0:4:39.640\nCarly Fox\nInstead.\n0:4:42.410 --> 0:4:42.690\nCarly Fox\nOK.\n0:4:47.20 --> 0:4:47.320\nCarly Fox\nAll right.\n0:4:47.330 --> 0:4:47.710\nCarly Fox\nThere we go.\n0:4:47.720 --> 0:4:48.490\nCarly Fox\nSorry about that guys.\n0:4:48.500 --> 0:4:49.260\nCarly Fox\nOK, alright.\n0:4:49.270 --> 0:4:49.950\nCarly Fox\nSo topic modeling.\n0:4:51.850 --> 0:4:59.630\nCarly Fox\nOK, so going to introduce the concepts of tech summarization and from gosh I can't talk this morning.\n0:4:59.640 --> 0:5:12.880\nCarly Fox\nSorry information extraction or then going to look at the processes that underlie topic modeling and we'll examine probably the most common type of topic modeling algorithm, latent Dirichlet allocation.\n0:5:13.330 --> 0:5:21.900\nCarly Fox\nWe'll look at the concept of topic coherence, and then we're going to apply actually a different a little bit more modern of a topic modeling algorithm.\n0:5:22.130 --> 0:5:34.170\nCarly Fox\nDo an example corpus so text summarization and information extraction is a huge part of natural language processing and text analysis.\n0:5:34.360 --> 0:5:43.840\nCarly Fox\nSo generally speaking, when you guys have a question about something you're trying to figure out some piece of information, where do you go to find an answer?\n0:5:49.690 --> 0:5:49.950\nSpeaker 1\nGoogle.\n0:5:49.290 --> 0:5:51.840\nTodd Hesson\nGoogle Chat P.\n0:5:52.490 --> 0:5:53.400\nMichael Truex\nGet all search engines.\n0:5:50.660 --> 0:5:55.530\nCarly Fox\nGoogle, right, Google or Chat TBT, that's those are sort of like our options, right?\n0:5:55.540 --> 0:6:9.210\nCarly Fox\nWe have Google and more recently we have chat TPT and when you type in a query to Google or to chat GPT we then go about finding the answer for talking about Google.\n0:6:9.220 --> 0:6:26.450\nCarly Fox\nAt least oftentimes we'll take a look at the snippet up at the top right it instead of just listing out all of these different sources that probably contain the answer, it provides an extracted summary.\n0:6:26.720 --> 0:6:30.410\nCarly Fox\nOr if we're talking about chat, GPT, we don't even have to look for this.\n0:6:30.420 --> 0:6:45.330\nCarly Fox\nIt's gonna give us an abstracted summary that's specific to our query, so this is that idea of information extraction before Google existed right before the Internet was indexed.\n0:6:45.580 --> 0:6:48.720\nCarly Fox\nFinding information online was really difficult, right?\n0:6:48.730 --> 0:6:55.790\nCarly Fox\nBecause you would have to go through web page, the web page or web page typing them in in order to find that information.\n0:6:56.80 --> 0:7:4.610\nCarly Fox\nGoogle made that easier by indexing websites that you could then query and more easily find the answer.\n0:7:5.100 --> 0:7:17.720\nCarly Fox\nMore recently, they started doing those snippets where it could identify the top most relevant website and actually extract out sentences that were relevant to your query.\n0:7:17.790 --> 0:7:32.80\nCarly Fox\nAnd then chat, GPT took that even a step further and didn't just give you a list of websites, but actually created a specific summary to your personal query to give you more specified search results.\n0:7:32.310 --> 0:7:47.810\nCarly Fox\nSo this is sort of been the evolution of information extraction and it's because if we couldn't find a way to distill all of the information that exists out in the world down, we would be faced with information overload, right.\n0:7:47.820 --> 0:7:50.930\nCarly Fox\nThere is so much information that exists out in the world.\n0:7:50.940 --> 0:8:6.150\nCarly Fox\nWe need a way to distill it down and so there are a number of different methods in tech summarization that allow us to do that, and this is even something that we do ourselves right when we're trying to like maybe explain a movie plot to someone.\n0:8:6.280 --> 0:8:13.910\nCarly Fox\nIf you've ever talked to a little kid and they're trying to explain a movie plot to you, they don't do a good job of this, right?\n0:8:13.920 --> 0:8:16.850\nCarly Fox\nThey'll tell you like every single detail that they remember.\n0:8:16.860 --> 0:8:18.420\nCarly Fox\nThey're like and this happened.\n0:8:18.430 --> 0:8:19.400\nCarly Fox\nAnd then this happened.\n0:8:19.410 --> 0:8:19.950\nCarly Fox\nAnd this happened.\n0:8:19.960 --> 0:8:23.340\nCarly Fox\nYou're like, Oh my gosh, information overload, right?\n0:8:23.430 --> 0:8:32.260\nCarly Fox\nSo the ability to distill down information into more like bite sized chunks that are really easy to digest and understand is an important thing we do as humans.\n0:8:32.270 --> 0:8:38.810\nCarly Fox\nAnd again, we seek to use natural language processing and unsupervised text mining to do this for us as well.\n0:8:39.300 --> 0:8:46.380\nCarly Fox\nSo the first method that I'm going to be talking about this week that we can use to do this is topic modeling.\n0:8:46.630 --> 0:8:52.870\nCarly Fox\nSo topic modeling does fall under this sort of class of unsupervised text mining.\n0:8:52.880 --> 0:8:56.780\nCarly Fox\nMethods of summarization and information extraction.\n0:8:57.330 --> 0:9:4.480\nCarly Fox\nThe goal was topic modeling is to use some probabilistic based statistical modeling techniques.\n0:9:5.80 --> 0:9:5.620\nCarly Fox\nUmm.\n0:9:5.900 --> 0:9:41.400\nCarly Fox\nFor example, singular value decomposition, which is a form of dimensionality reduction, or latent Dirichlet allocation, which we're gonna be talking about here next, in order to take basically a much larger group or corpus of documents and distill it down to a set of sort of underlying or latent topics, themes or concepts, so that we can take a group of unknown documents and begin to understand the topics that are discussed amongst them as a means of summarization.\n0:9:41.580 --> 0:9:58.930\nCarly Fox\nSo we can sort of think about it again as a dimensionality reduction technique to take a large number of word vectors that we would get when we feature engineer our corpus and then distill it down into a much smaller set of topic.\n0:9:58.940 --> 0:10:6.170\nCarly Fox\nSo we could take like a 16,000 dimensional set of word vectors and distill it down to like 20 topics.\n0:10:6.180 --> 0:10:10.230\nCarly Fox\nSo we can understand again high level what are these documents talking about.\n0:10:10.400 --> 0:10:14.310\nCarly Fox\nSo we can see a basic example of this here.\n0:10:14.700 --> 0:10:21.350\nCarly Fox\nSo say we're dealing with the following words that we would of course vectorize, but these would be the original words.\n0:10:22.150 --> 0:10:26.300\nCarly Fox\nHow many topics would you guys say exist amongst these set of words?\n0:10:29.830 --> 0:10:32.340\nTodd Hesson\nI'd say probably 3 maybe.\n0:10:32.110 --> 0:10:32.480\nCarly Fox\nOK.\n0:10:32.970 --> 0:10:33.430\nCarly Fox\nYeah.\n0:10:33.470 --> 0:10:35.580\nCarly Fox\nOK, three, two or three?\n0:10:35.620 --> 0:10:36.110\nCarly Fox\nYeah.\n0:10:36.200 --> 0:10:37.620\nCarly Fox\nAnd what would you say those topics are?\n0:10:40.900 --> 0:10:41.800\nTyler Stout\nEnviron hint.\n0:10:42.150 --> 0:10:42.600\nCarly Fox\nEnvironment.\n0:10:44.800 --> 0:10:45.420\nTodd Hesson\nTechnology.\n0:10:46.160 --> 0:10:46.700\nCarly Fox\nTechnology.\n0:10:53.460 --> 0:10:53.760\nCarly Fox\nOK.\n0:10:53.720 --> 0:10:55.100\nTodd Hesson\nYeah, yeah, yeah.\n0:10:54.430 --> 0:10:55.930\nMichael Truex\nIt covers most of it really, yeah.\n0:10:53.770 --> 0:10:58.990\nCarly Fox\nSo OK, so let me we have two topics, OK, two, two or three topics.\n0:10:59.380 --> 0:11:0.800\nCarly Fox\nMaybe there's some subtopics in there.\n0:11:0.810 --> 0:11:1.470\nCarly Fox\nWe'll stick with the two.\n0:11:1.480 --> 0:11:3.250\nCarly Fox\nWe'll say environment and technology.\n0:11:3.420 --> 0:11:8.310\nCarly Fox\nSo that's basically the process that we're gonna be doing with topic modeling.\n0:11:8.420 --> 0:11:11.50\nCarly Fox\nWe're going to take our corpus.\n0:11:11.180 --> 0:11:16.340\nCarly Fox\nWe're gonna convert it into a term document matrix, as we've done in the past.\n0:11:16.680 --> 0:11:19.110\nCarly Fox\nWe're then going to feature engineer it.\n0:11:19.320 --> 0:11:21.430\nCarly Fox\nSorry if you guys can hear my dog barking in the background.\n0:11:21.440 --> 0:11:23.670\nCarly Fox\nThis probably feeling like delivering a package or something.\n0:11:24.260 --> 0:11:31.520\nCarly Fox\nUh, we're going to feature engineer it to create our set of document vectors or I'm sorry.\n0:11:32.180 --> 0:11:34.210\nCarly Fox\nDocument and word vectors.\n0:11:34.420 --> 0:11:40.620\nCarly Fox\nWe're then going to apply a topic model for kind of A2 pronged purpose.\n0:11:40.800 --> 0:11:50.390\nCarly Fox\nWe wanna understand the topic composition of our documents to understand what are the topics that are being discussed in a given document.\n0:11:50.500 --> 0:11:56.830\nCarly Fox\nAnd we also wanted to understand what those topics are and we do that by examining the topic word distribution.\n0:11:56.840 --> 0:12:8.0\nCarly Fox\nSo basically understanding what are the most common and most important words that exist within a topic and that will give us insight into what the topic is and that leads us to our third step.\n0:12:8.10 --> 0:12:14.330\nCarly Fox\nAs I said, to evaluate the top features of that topic to produce some sort of meaningful thematic label.\n0:12:14.550 --> 0:12:19.580\nCarly Fox\nSo step one, of course, is our vectorization.\n0:12:19.670 --> 0:12:20.720\nCarly Fox\nWe've done this before.\n0:12:20.730 --> 0:12:25.20\nCarly Fox\nWe've covered this where we are taking our larger corpus of documents.\n0:12:25.250 --> 0:12:42.960\nCarly Fox\nWe are creating a term document matrix like we see here and then we can use a bag of words model technique like count vectorization or TFIDF vectorization in order to feature engineer our documents to convert them into a structured data set.\n0:12:43.990 --> 0:12:50.720\nCarly Fox\nOnce we have that set up, we can then choose from a host of different topic modeling algorithms.\n0:12:51.150 --> 0:13:18.450\nCarly Fox\nI would say that latent Dirichlet allocation or LDA is probably the most commonly implemented, so that's the one I'm gonna focus on here just for the sake of time and the idea with latent Dirichlet allocation is that there's this main assumption that documents are mixtures of topics and topics are characterized by a set distribution of words.\n0:13:18.680 --> 0:13:33.70\nCarly Fox\nSo basically what LDA is doing is that it's working backwards from the document level to determine which topics would generate that document, and which words would generate that topic.\n0:13:33.200 --> 0:13:36.780\nCarly Fox\nSo we're sort of doing like a top down approach

here.\n0:13:38.340 --> 0:13:50.610\nCarly Fox\nSo generally speaking, the way that LDA works is that sort of like clustering analysis u
pfront, you're going to select a value for the number of topics you wanted to find.\n0:13:50.840 --> 0:13:57.930\nCarly Fox\nSo that is
user defined which I guess is a downside of this, but I don't know.\n0:13:57.940 --> 0:14:3.670\nCarly Fox\nI I think with something li
ke topic modeling, it's not as big of a deal.\n0:14:3.680 --> 0:14:11.250\nCarly Fox\nI think to like be off in terms of the number of
topics that you're producing, because really it's just a decision for the user.\n0:14:11.260 --> 0:14:30.950\nCarly Fox\nI think to det
ermine sort of how granular you wanna get, like if you want really high level topics that are sort of broad then you can do a small num
ber and if you want to get really specific in terms of the information you're or I'm sorry, really specific in terms of like the topic
s, then do a larger number of topics.\n0:14:30.960 --> 0:14:32.230\nCarly Fox\nSo it's not as big of a deal.\n0:14:33.180 --> 0:14:36.9
50\nCarly Fox\nUmm to have to set it yourself, I would say as it is in clustering analysis.\n0:14:37.60 --> 0:14:50.550\nCarly Fox\nSo
you select your size 4, the topic number of topics you want to find, and then every word that exists within your corpus is going to at
first be randomly assigned to one of those topics.\n0:14:50.780 --> 0:14:55.570\nCarly Fox\nAnd So what it's gonna do in each iteration
is update two different matrices.\n0:14:55.660 --> 0:15:8.760\nCarly Fox\nThere's going to be a topic word distribution matrix which is
going to look at the probability that a given word could appear in each topic, right?\n0:15:8.910 --> 0:15:16.850\nCarly Fox\nAnd so ag
ain, this is really has to do with the distribution of words across all of your documents, right?\n0:15:16.860 --> 0:15:39.0\nCarly Fox
\nSo basically looking at the Co occurrence of words within a given document across all documents, right to say that like if the words
I don't know tree and flower are Co occur or Co occurring a lot across you know a set number of documents then they probably belong to
the same topic.\n0:15:39.10 --> 0:15:42.90\nCarly Fox\nWhereas if you have the word like tree and.\n0:15:43.300 --> 0:15:51.300\nCarly
Fox\nRoadway that are not commonly Co occurring across documents, then they probably belong to different topics.\n0:15:51.400 --> 0:15:
57.480\nCarly Fox\nSo it's looking at the that the joint probability of words across all of your documents.\n0:15:57.670 --> 0:15:57.91
0\nCarly Fox\nOK.\n0:15:57.920 --> 0:16:0.40\nCarly Fox\nThat's our topic word distribution matrix.\n0:16:0.350 --> 0:16:12.160\nCarly
Fox\nThe other thing that it's updating is a document topic distribution matrix which has to do with the mixture of topics present with
in a document.\n0:16:12.170 --> 0:16:20.740\nCarly Fox\nSo basically, are there topics that tend to Co occur within documents, right?\n
0:16:20.750 --> 0:16:31.950\nCarly Fox\nSo if we have in one document a topic that is about nature, that's maybe gonna be more likely t
o Co occur in that same document with a topic about.\n0:16:33.320 --> 0:16:38.450\nCarly Fox\nI don't know whether versus one about tec
hnology like we saw before, OK.\n0:16:38.460 --> 0:16:42.430\nCarly Fox\nSo it's looking at the mixture of topics within documents as w
ell.\n0:16:42.740 --> 0:16:45.910\nCarly Fox\nOK, so what's going to go through this process iteratively?\n0:16:45.920 --> 0:17:7.780\n
Carly Fox\nSort of reassigning words to topics to find sort of the best and most probable combination of words within a topic so that w
e sort of get the most coherent topics possible, and then it's going to again also iteratively decide which topics seem to make up each
document.\n0:17:7.790 --> 0:17:34.570\nCarly Fox\nWhat is the highest probability topics that are coming up in each document so it does
this iteratively until it's sort of stopped reassigning things sort of like in K means clustering analysis that would be the point of c
onversions, and so at the end what we get is our output is the probability that a given word belongs to a topic, and then the probabili
ty of each topic within a document.\n0:17:36.270 --> 0:17:44.710\nCarly Fox\nAnd So what that gives us in terms of our sort of ability
to interpret this information is sort of threefold.\n0:17:45.200 --> 0:17:45.690\nCarly Fox\nOK.\n0:17:45.760 --> 0:17:57.180\nCarly Fo
x\nSo given that sort of probabilities that we get as our output, we can determine which topics are most dominant within a given.\n0:1
7:58.220 --> 0:18:1.260\nCarly Fox\nUmm, within within specific documents, right?\n0:18:1.270 --> 0:18:13.850\nCarly Fox\nSo I can know
that document A is primarily about maybe topic one and topic two, which are nature and weather and then I can see this other documen
t.\n0:18:13.860 --> 0:18:23.890\nCarly Fox\nDocument B is primarily about topic 3, which is technology and so on, and that is going to
give me a high level understanding of what a given document is about.\n0:18:24.40 --> 0:18:24.310\nCarly Fox\nOK.\n0:18:25.360 --> 0:1
8:36.390\nCarly Fox\nI didn't also get a sense of which documents are sort of most prototypical or the best example of a particular top
ic.\n0:18:36.490 --> 0:18:55.550\nCarly Fox\nSo if I wanna get insight beyond sort of just what the top words are within a topic to see
like what is an example of a document that sort of really fits the mold of topic one, I can pull out that document, maybe give it a qui
ck skim and get some more context for what that topic is about.\n0:18:55.600 --> 0:18:57.370\nCarly Fox\nSo that's sort of a nice thing
you can do.\n0:18:57.800 --> 0:19:3.890\nCarly Fox\nAnd then finally, we can look at the words that have the greatest weight within a t
opic.\n0:19:3.900 --> 0:19:12.820\nCarly Fox\nAnd really, these are gonna be the words that have the highest probability of belonging t
o that topic, and that's going to give us an idea of what that topic is.\n0:19:12.990 --> 0:19:24.510\nCarly Fox\nSo in LDA, because w
e're dealing with basically probabilities, word probabilities are values are always gonna be positive, and they're going to range betwe
en zero and one.\n0:19:24.580 --> 0:19:26.990\nCarly Fox\nSo you can really just think of them as probabilities.\n0:19:27.80 --> 0:19:4
1.760\nCarly Fox\nSo if we had a topic here and we had the following words that fell within this topic, we would see that crime is sort
of our highest weighted our highest probable word within that topic.\n0:19:41.770 --> 0:19:54.990\nCarly Fox\nWe then would have killer
with similar probabilities slightly lower, followed by real life, and then we have cat with a much lower probability and ice cream with
an even lower probability.\n0:19:55.190 --> 0:20:0.640\nCarly Fox\nSo given these words and these weights, what would you guys say?\n0:
20:0.650 --> 0:20:1.740\nCarly Fox\nThis topic is about.\n0:20:2.860 --> 0:20:3.700\nChase Jensen\nTrue crack.\n0:20:3.930 --> 0:20:6.7
0\nCarly Fox\nTrue cry, right?\n0:20:6.80 --> 0:20:13.730\nCarly Fox\nSo we do see some non relevant words sort of leaking into this to
pic because but they are very low probability.\n0:20:13.740 --> 0:20:23.460\nCarly Fox\nSo we can examine again our top probability wor
ds and that's gonna give us insight into what the topic of or what this topic really pertains to.\n0:20:26.150 --> 0:20:36.640\nCarly F
ox\nSo I did say before that is not as big of a deal to sort of determine the exact value or the best value of K in topic modeling.\n0:
20:36.910 --> 0:20:46.320\nCarly Fox\nBut there is still a way to sort of empirically decide that if it's something that is of interes
t, and that is through a method called topic coherence.\n0:20:46.710 --> 0:21:4.920\nCarly Fox\nSo as we know topic modeling, unsupervi
sed text mining technique, there isn't really a way to evaluate the accuracy of the topics that you've created or the topic assignment
within a document, because there isn't a ground truth that we can compare it against, right?\n0:21:4.930 --> 0:21:27.250\nCarly Fox\nWe
don't have labels going into this, so topic coherence is sort of a way that we can sort of semi evaluate the quality of our topics and
that can be done by examining the level of similarity between the highest scoring or the highest or the most probable words within a to
pic.\n0:21:27.660 --> 0:21:33.50\nCarly Fox\nSo the steps to doing this, it's basically again just all comes down to probabilities.\n0:
21:33.200 --> 0:21:41.570\nCarly Fox\nSo basically, for a given topic, we can calculate the joint probabilities of the top words in tha
t topic.\n0:21:41.580 --> 0:21:51.850\nCarly Fox\nSo given that or you know what is the probability that tree and again flower cooccur
within our corpus, right?\n0:21:51.860 --> 0:21:55.790\nCarly Fox\nHow often or what is the probability of those two things occurring w
ith one another?\n0:21:55.850 --> 0:22:16.330\nCarly Fox\nAnd we can look at that probability then for each of those pairs, we can then
calculate the conditional probability, which is basically not just how likely are they to Co occur, but how much more likely are they t
o Co occur together versus how often we would expect them to Co occur just by chance alone.\n0:22:16.420 --> 0:22:16.610\nCarly Fox\nO
K.\n0:22:16.620 --> 0:22:33.280\nCarly Fox\nThat would give us our conditional probability, and then once we have those conditional pro
babilities for each set of our top words within a topic, we can aggregate together the probabilities and then that'll give us a coheren
t score.\n0:22:34.150 --> 0:22:45.40\nCarly Fox\nThe higher the coherence score, the better quality your topic is, because we're seeing
that the words that exist within that topic are highly related to one another.\n0:22:45.210 --> 0:22:55.640\nCarly Fox\nIf you have a l
ow coherence score, we're saying that these are words that got put together, but they don't actually tend to Co occur that often and th
ey don't really seem to be related.\n0:22:55.650 --> 0:23:7.510\nCarly Fox\nSo if you have a low coherence score, you have sort of poor
quality topics, so similar to the elbow rule that you can use in K means clustering analysis.\n0:23:7.520 --> 0:23:19.300\nCarly Fox\nW
e can similarly plot the change in coherence score against the number of topics that we have in order to sort of determine what an opti
mal number of topics is.\n0:23:19.310 --> 0:23:28.810\nCarly Fox\nSo here we see as we increase our topics that we're getting sort of o
ur highest level of coherence at like 8 or 9 topics.\n0:23:28.940 --> 0:23:38.490\nCarly Fox\nSo that is a way that you can sort of mor
e empirically determine topics in a topic model, but isn't something that's always necessary?\n0:23:39.190 --> 0:23:51.530\nCarly Fox\n
Umm so it's it is appropriate to also again do it in a more user defined way just depending on how specific or broad you want your topi
cs to be so.\n0:23:53.670 --> 0:24:1.80\nCarly Fox\nOK, so last thing to discuss before we jump into our example is just where is topic
modeling useful.\n0:24:1.630 --> 0:24:15.880\nCarly Fox\nSo topic modeling, I would say tends to be more useful when you're dealing wit
h at least slightly longer documents in that you want to use it with documents you would expect to have more than one.\n0:24:16.690 --
0:24:17.960\nCarly Fox\nTopic right?\n0:24:17.970 --> 0:24:27.90\nCarly Fox\nBecause the method like LDA is assuming that you have mult

iple topics that make up a document, right?\n0:24:27.100 --> 0:24:35.710\nCarly Fox\nSo topic modeling isn't gonna be as you as useful on like really short documents that are maybe only like a sentence or two long.\n0:24:35.720 --> 0:24:44.330\nCarly Fox\nIt's gonna be more youthful on things like, say, you've got a group of quarterly earning reports and you wanna know, what were they talking about?\n0:24:44.340 --> 0:24:44.850\nCarly Fox\nHigh level.\n0:24:44.860 --> 0:24:46.730\nCarly Fox\nWhat were some of the themes that were coming up?\n0:24:47.180 --> 0:24:53.760\nCarly Fox\nTopic modeling would be a great thing to apply here to give you that high level summary of what was being discussed.\n0:24:54.270 --> 0:24:57.470\nCarly Fox\nAlso useful in things like call transcripts.\n0:24:58.270 --> 0:25:14.530\nCarly Fox\nHave you ever collection of news articles that you want to understand, or which example I'm gonna show you guys if you are interested in keeping up in a particular field like we're gonna look at the new RIPS conference, right?\n0:25:14.540 --> 0:25:24.190\nCarly Fox\nYou're trying to stay up to date on like what is cutting edge in your field, but you don't have time to just sit around and read a bunch of conference papers.\n0:25:25.100 --> 0:25:28.970\nCarly Fox\nI don't know who would maybe wanna do that anyways, but you just wanna know.\n0:25:28.980 --> 0:25:34.130\nCarly Fox\nOK, what are some of the themes that are being talked about at these conferences?\n0:25:34.440 --> 0:25:38.150\nCarly Fox\nThen topic modeling again is gonna be a really useful thing to just sort of highlight.\n0:25:38.160 --> 0:25:39.940\nCarly Fox\nThese are the topics that are coming up.\n0:25:40.90 --> 0:25:46.290\nCarly Fox\nThis is what is considered cutting edge and then you could then from there decide you know I want it now.\n0:25:46.300 --> 0:25:50.550\nCarly Fox\nLike read this one specific paper I want to like look up a YouTube video or whatever.\n0:25:50.560 --> 0:25:52.150\nCarly Fox\nIf I don't want to read a paper, yes, Todd.\n0:25:54.610 --> 0:25:59.120\nTodd Hesson\nSo as he calls scripts, I've been some something similar but with chat transcripts.\n0:25:59.130 --> 0:26:1.100\nTodd Hesson\nI'm assuming you can do the kind of same function.\n0:26:1.830 --> 0:26:2.80\nCarly Fox\nMm-hmm.\n0:26:7.30 --> 0:26:7.310\nCarly Fox\nUmm.\n0:26:2.570 --> 0:26:12.660\nTodd Hesson\nWould that transcript need to be all in like 1 document or does it need to be separated out into individual lines like user and agent are the best structure?\n0:26:11.880 --> 0:26:16.930\nCarly Fox\nSo you could separate them out, but I, but I would still like.\n0:26:17.400 --> 0:26:27.840\nCarly Fox\nI would still want to like if you were to separate them out by speaker, you would still wanna have like 1 document that is all of the lines of the speaker and one document.\n0:26:27.850 --> 0:26:52.210\nCarly Fox\nThat's all of the lines of the agent, because you're not like you really needed to have some sort of length on it to have multiple topics come up, because if it's just like each conversational turn as a different document, it's going to, I mean, at best have one topic come up, but it might not even it could just be like, yes, no.\n0:26:52.220 --> 0:26:53.90\nCarly Fox\nOK, whatever.\n0:26:53.100 --> 0:26:54.860\nCarly Fox\nIn which case there's not even gonna be a topic.\n0:26:55.750 --> 0:26:56.180\nCarly Fox\nUmm.\n0:26:56.390 --> 0:26:59.20\nCarly Fox\nOr you'll get a topic that's just like, yes, no.\n0:26:59.30 --> 0:26:59.640\nCarly Fox\nOK.\n0:26:59.650 --> 0:27:0.180\nCarly Fox\nThank you.\n0:27:0.190 --> 0:27:2.220\nCarly Fox\nThat kind of thing, which is like not super helpful.\n0:27:2.290 --> 0:27:13.440\nCarly Fox\nSo I would say either you could try leaving it together or create a one document for all of the agent and all of the client of responses.\n0:27:13.530 --> 0:27:13.990\nCarly Fox\nGo question.\n0:27:14.970 --> 0:27:15.370\nTodd Hesson\nI'm sure.\n0:27:15.380 --> 0:27:15.710\nTodd Hesson\nThank you.\n0:27:16.160 --> 0:27:18.180\nCarly Fox\nGot awesome.\n0:27:18.930 --> 0:27:19.290\nCarly Fox\nOK.\n0:27:19.300 --> 0:27:32.520\nCarly Fox\nSo we'll go ahead and transition over to our Python notebook and again, we're gonna be looking at a bunch of conference papers from merits, which, if you're not familiar, is a big conference.\n0:27:32.530 --> 0:27:33.780\nCarly Fox\nThey do every year.\n0:27:33.890 --> 0:27:37.880\nCarly Fox\nThat has to do with neural networks, machine learning and other neural processing.\n0:27:37.890 --> 0:27:54.240\nCarly Fox\nSo umm, think pretty, pretty relevant and is a good source again for a field that is changing as rapidly as like data analytics, machine learning that kind of thing is a good conference to keep up with.\n0:27:54.250 --> 0:27:58.260\nCarly Fox\nSo you know what is cutting edge, but this sort of thing would also apply.\n0:27:58.270 --> 0:28:10.350\nCarly Fox\nI know a lot of people were watching, like the Microsoft Conference or Google IO also had a conference where recently so this is something you could apply to like conference transcripts as well to sort of be like, OK, what?\n0:28:10.390 --> 0:28:11.860\nCarly Fox\nI don't wanna watch the whole thing.\n0:28:11.910 --> 0:28:13.50\nCarly Fox\nWhat were they talking about?\n0:28:13.60 --> 0:28:13.880\nCarly Fox\nWhat were the highlights?\n0:28:15.20 --> 0:28:31.200\nCarly Fox\nSo all right, so here we have our template right here and I talked about LDA and the lecture because LDA is again probably the most well known method of topic modeling.\n0:28:31.490 --> 0:28:45.430\nCarly Fox\nThe one I'm going to show you is very similar in terms of its process, but it's a little bit more modern of it technique and it is just runs faster, which is why I'm going to demonstrate it here instead of LDA.\n0:28:45.550 --> 0:28:53.10\nCarly Fox\nBut if anybody is interested in getting additional code to do LDA, just shoot me a message and I can send you over some sample code as well.\n0:28:54.100 --> 0:28:54.550\nCarly Fox\nOK.\n0:28:54.590 --> 0:28:57.940\nCarly Fox\nSo we'll go ahead and load in our dependencies.\n0:29:0.440 --> 0:29:8.950\nCarly Fox\nWhen the method we're going to be talking about, I didn't even mention is called non negative matrix factorization or an MF for short. OK.\n0:29:10.610 --> 0:29:12.720\nCarly Fox\nAlright, so we've got our dependencies loaded in.\n0:29:14.100 --> 0:29:24.270\nCarly Fox\nWe're gonna set up our text preprocessing function first, because we're going to be basically extracting from the neuroph website.\n0:29:24.280 --> 0:29:26.810\nCarly Fox\nWe're going to be downloading the.\n0:29:28.80 --> 0:29:30.990\nCarly Fox\nCompressed folder of conference papers.\n0:29:31.140 --> 0:29:44.560\nCarly Fox\nAnd then we're going to need to extract out those papers from specific folders, read in the files, and we're just going to add in the text preprocessing into that loop function just to save ourselves some time.\n0:29:44.570 --> 0:29:48.510\nCarly Fox\nMake it a little bit more efficient, so we'll start with our text preprocessing function.\n0:29:48.520 --> 0:29:54.870\nCarly Fox\nHere we're going to be doing all of our sort of typical methods here in terms of cleaning up the text.\n0:29:54.880 --> 0:30:2.400\nCarly Fox\nWe're gonna get rid of anything that is not a a letter.\n0:30:2.460 --> 0:30:5.200\nCarly Fox\nAlright, so we'll get rid of numbers.\n0:30:5.210 --> 0:30:8.910\nCarly Fox\nSpecial characters, anything like that, we'll perform case folding.\n0:30:8.920 --> 0:30:23.310\nCarly Fox\nWe're gonna go ahead and tokenize our text so that we can remove all stop words, and we're also gonna bring in a lemmatized or in order to perform lemmatization to drop down all of our words to their root forms.\n0:30:23.420 --> 0:30:27.60\nCarly Fox\nSo we'll go ahead and bring in our Wordnet limit tizer.\n0:30:33.610 --> 0:30:38.720\nCarly Fox\nWe're then going to go ahead and apply the limit tizer to our tokens.\n0:30:49.790 --> 0:30:57.30\nCarly Fox\nAnd then we're just going to rejoin back together our Lima so that we have continuous text again text streams.\n0:31:5.580 --> 0:31:5.750\nCarly Fox\nOK.\n0:31:7.750 --> 0:31:12.50\nCarly Fox\nAll right, not going to spend a ton of time talking about what we're doing here.\n0:31:12.60 --> 0:31:15.50\nCarly Fox\nBasically, we're just going to go ahead and access the.\n0:31:16.70 --> 0:31:25.730\nCarly Fox\nRepository of conference papers from the nerves website download, umm, a folder from one of the conference years.\n0:31:25.740 --> 0:31:29.410\nCarly Fox\nGo ahead and unzip it.\n0:31:29.520 --> 0:31:51.280\nCarly Fox\nThen we're going to go ahead and just ice create a directory of the different of the folder path so that we can then down here create a list of the papers, extract them out from the folders, and then go ahead and for each of the papers that exist within the four folders, we're going to be pulling papers from, we're gonna go ahead and clean them.\n0:31:51.320 --> 0:31:53.830\nCarly Fox\nAnd then we're going to append them into a list.\n0:31:54.0 --> 0:31:55.940\nCarly Fox\nSo and just go ahead.\n0:31:58.840 --> 0:32:3.390\nCarly Fox\nAnd download our papers here and we can see this is our directory.\n0:32:3.400 --> 0:32:8.470\nCarly Fox\nSo we have a number of different folders that have these conference papers.\n0:32:8.760 --> 0:32:14.40\nCarly Fox\nWe're just gonna go ahead and extract out the conference papers from these folders right here.\n0:32:15.770 --> 0:32:21.580\nCarly Fox\nZero through 4, just for the sake of not having a huge amount of conference papers.\n0:32:21.650 --> 0:32:33.70\nCarly Fox\nSo go ahead and again we want to specify that we want folders 0 through 4, so we'll set our range here to 0 through 5.\n0:32:33.180 --> 0:32:40.510\nCarly Fox\nWe'll create an empty papers list in order to store each of the papers that get pulled in and cleaned.\n0:32:42.200 --> 0:32:45.570\nCarly Fox\nThen we're going to go ahead and create the.\n0:32:47.840 --> 0:32:50.0\nCarly Fox\nSet up the folder path in the file names.\n0:32:50.220 --> 0:33:10.510\nCarly Fox\nPython knows where to look for each of these papers to pull them in, and then once they have been read in and stored as our data object, we can go ahead and apply our preprocessed text function and store that in a new object called preprocessed data.\n0:33:17.870 --> 0:33:24.130\nCarly Fox\nAnd then each time it gets cleaned, we want to take that paper and append it into our papers list.\n0:33:29.690 --> 0:33:34.0\nCarly Fox\nPet and this is gonna take probably around like half a minute to run.\n0:33:34.10 --> 0:33:40.200\nCarly Fox\nSo we're just gonna move on while that's loading and just start typing in some of our next steps here.\n0:33:40.290 --> 0:33:47.140\nCarly Fox\nSo we're going to perform some feature engineering because we need to create our term document matrix.\n0:33:47.270 --> 0:33:48.780\nCarly Fox\nWe'll just use TF IDF.\n0:33:51.270 --> 0:34:8.730\nCarly Fox\nJust with basic parameters, we're not going to do anything, anything fancy here because we're not dealing with a huge corpus, so I don't feel the need to sort of like do like the Max DF or the Mindy F to limit the number of features that we're including, we're going to be working with around, I think, 16,000 features.\n0:34:8.740 --> 0:34:9.750\nCarly Fox\n

So, not too bad.\n0:34:14.40 --> 0:34:14.370\nCarly Fox\nOK.\n0:34:14.380 --> 0:34:43.650\nCarly Fox\nSo just basic TFIDF vectorizer wh
ere you're going to store that in a document term matrix DTM where we take our vectorizer and we bit it to our data which is papers the
n going to create vocabulary set because we're going to want to be able to pull out each of our feature names at a later point.\n0:34:4
3.660 --> 0:34:48.830\nCarly Fox\nSo we can see sort of what words make up each topic.\n0:34:48.840 --> 0:34:51.90\nCarly Fox\nSo we'll
need to create that vocabulary list.\n0:34:51.180 --> 0:34:55.190\nCarly Fox\nWe're gonna create an array from our features.\n0:35:5.94
0 --> 0:35:14.210\nCarly Fox\nAnd finally, we'll just go ahead and print out our shape so that we know how many documents we have in ou
r corpus and the number of features we're dealing with.\n0:35:14.280 --> 0:35:20.910\nCarly Fox\nSo just give that a couple more second
s to load being a little slow on me.\n0:35:25.440 --> 0:35:25.680\nCarly Fox\nComma.\n0:35:37.730 --> 0:35:37.950\nCarly Fox\nCame.\n0:
35:37.960 --> 0:35:39.840\nCarly Fox\nMaybe I'll start typing in the step blog.\n0:35:40.870 --> 0:35:42.560\nCarly Fox\nMaybe it's loa
ded for other people.\n0:35:42.910 --> 0:35:45.580\nCarly Fox\nOK, Todd, just move on down here.\n0:35:45.590 --> 0:35:46.870\nCarly Fo
x\nWell, that's still loading.\n0:35:47.290 --> 0:35:53.380\nCarly Fox\nWe can set up our negative matrix factorization, so I'm going t
o set the number of topics equal to 20.\n0:35:53.450 --> 0:35:55.540\nCarly Fox\nAgain, arbitrary choice.\n0:35:55.710 --> 0:35:58.140
\nCarly Fox\nBut for me, 20 is a good way.\n0:35:58.150 --> 0:36:4.200\nCarly Fox\nThis is a conference that covers a lot of different
topics because it's pretty broad, it's machine learning.\n0:36:4.210 --> 0:36:6.620\nCarly Fox\nYou'll networks and also like neural pr
ocessing.\n0:36:6.630 --> 0:36:11.360\nCarly Fox\nSo we would expect there to be sort of a large number of topics that papers could fal
l into.\n0:36:11.370 --> 0:36:19.430\nCarly Fox\nSo we'll set it equal to 20 in terms of there we go, finish running so we can go ahead
and go ahead and apply that.\n0:36:20.480 --> 0:36:27.220\nCarly Fox\nSo what we're dealing with here, we have 144 documents in our cor
pus and we have around 16,000 features.\n0:36:29.150 --> 0:36:33.780\nCarly Fox\nSo we'll go ahead and set up our negative matrix facto
rization model.\n0:36:34.70 --> 0:36:38.290\nCarly Fox\nWe're going to set our number of components equal to our number of topics, whic
h is 20.\n0:36:39.940 --> 0:36:42.940\nCarly Fox\nWe'll set our random state equal to 42.\n0:36:43.920 --> 0:36:44.840\nCarly Fox\nWill
do.\n0:36:46.380 --> 0:36:58.270\nCarly Fox\nCD is our solver which is coordinate descent, and we'll set our Max iterations as 1000, so
that's the number of iterations.\n0:36:58.280 --> 0:37:3.80\nCarly Fox\nAgain, that this is going to run through in terms of updating t
hose two different matrices.\n0:37:4.650 --> 0:37:8.120\nCarly Fox\nBefore it will stop running if it hasn't already converged.\n0:37:
8.320 --> 0:37:8.890\nCarly Fox\nOK.\n0:37:9.290 --> 0:37:10.300\nCarly Fox\nThen we'll go ahead.\n0:37:10.390 --> 0:37:16.50\nCarly Fo
x\nAnd once we have this specified, we're good to go ahead and fit it to our document term matrix.\n0:37:27.800 --> 0:37:29.830\nCarly
Fox\n8 And it runs pretty quickly.\n0:37:29.890 --> 0:37:31.490\nCarly Fox\nTalk about like 5 seconds to run.\n0:37:33.60 --> 0:37:37.4
80\nCarly Fox\nLDA is going to take a bit longer.\n0:37:37.490 --> 0:37:38.990\nCarly Fox\nI mean, this isn't a huge corpus.\n0:37:39.0
--> 0:38:1.200\nCarly Fox\nI probably wouldn't have taken like, you know, multiple minutes to run, but once you start to sort of increa
se the number of documents you're working with, increase the number of features increase maybe the number of topics that you wanna fin
d, there's a pretty significant difference in terms of the amount of time it takes to run this method, the negative matrix factorizatio
n versus LDA, LDA is quite a bit slower.\n0:38:1.350 --> 0:38:7.510\nCarly Fox\nSo, OK, alright, so now we have created our topic mode
l.\n0:38:7.590 --> 0:38:12.330\nCarly Fox\nWe want to now get out those three different pieces of information that we talked about, rig
ht.\n0:38:12.480 --> 0:38:37.100\nCarly Fox\nSo we can examine what are the top terms in our topic so that we can begin to understand,
OK, what are these topics we can also examine the top topic per document to say, OK, for a given document, what are the topics that see
m to compose it right to make up that document.\n0:38:37.210 --> 0:38:49.220\nCarly Fox\nSo we can understand what that document is abo
ut and then we can finally examine the the most critical prototypical document for a given topic.\n0:38:49.310 --> 0:38:49.580\nCarly F
ox\nOK.\n0:38:49.590 --> 0:38:52.420\nCarly Fox\nSo those are sort of our three things we get out of our topic model.\n0:38:52.460 -->
0:39:0.510\nCarly Fox\nSo we'll start with examining the top turns per topic by creating a term topic matrix duck.\n0:39:1.290 --> 0:3
9:8.400\nCarly Fox\nFirst, we're going to extract out what our terms are for our for each topic and we can get that from our.\n0:39:10.
800 --> 0:39:13.280\nCarly Fox\nComponents attribute from our model.\n0:39:15.670 --> 0:39:15.960\nCarly Fox\nOK.\n0:39:17.20 --> 0:39:
17.530\nCarly Fox\nAll right.\n0:39:17.820 --> 0:39:26.490\nCarly Fox\nAnd I'm just going to add in some comments to discuss what each
of these steps are doing since they thought it would maybe not be super fun to type all these steps out.\n0:39:26.680 --> 0:39:37.970\n
Carly Fox\nBut basically what we're doing here is we're gonna save the indices of our top key terms by topic, or then going to match th
e key.\n0:39:40.540 --> 0:39:48.740\nCarly Fox\nLike, gosh, the key term indices with that vocab list that we created a couple of steps
back.\n0:39:48.750 --> 0:39:54.800\nCarly Fox\nSo that's going to be again, like the actual words themselves versus just their word vec
tor, OK.\n0:39:58.580 --> 0:40:5.100\nCarly Fox\nAnd we're going to go ahead and we will join the key terms by topic so that they're gr
ouped together.\n0:40:6.800 --> 0:40:14.870\nCarly Fox\nIt's all just allow us to display it better and then we're going to go ahead an
d create a data frame to view this just so it's a little bit easier to understand.\n0:40:15.0 --> 0:40:18.750\nCarly Fox\nSo our data w
e're filling in is our topics.\n0:40:18.760 --> 0:40:26.920\nCarly Fox\nWe're going to be looking at each of the terms per topic or are
20 different topics.\n0:40:30.180 --> 0:40:30.320\nCarly Fox\nNo.\n0:40:32.860 --> 0:40:33.280\nCarly Fox\nThere we go.\n0:40:33.510 --
> 0:40:34.800\nCarly Fox\nOK, perfect.\n0:40:35.130 --> 0:40:47.590\nCarly Fox\nOK, so here we see in my data frame I have an observati
on or a row for each of my 20 topics and I can see the top words that make up each of these topics.\n0:40:47.740 --> 0:40:53.110\nCarly
Fox\nSo I can begin to start to understand what some of these topics pertain to.\n0:40:53.600 --> 0:40:58.70\nCarly Fox\nSo let's see,
there's one.\n0:40:58.120 --> 0:41:6.830\nCarly Fox\nSo like for example here topic two, we have a lot of words that are related to neu
rons, right?\n0:41:6.840 --> 0:41:16.390\nCarly Fox\nSo I would assume without again looking too much at it, that these are gonna be pa
pers that have to do with actual like neural processes.\n0:41:16.540 --> 0:41:30.990\nCarly Fox\nWe have things like dendritic channel
s, so dendrites happening to do with neurons and haptic and synapses also have to do when neurons membrane cell conductance activity, v
oltage impacts and abs tree and so on.\n0:41:31.0 --> 0:41:31.210\nCarly Fox\nRight.\n0:41:31.220 --> 0:41:35.880\nCarly Fox\nSo these
are going to be papers that have to do with actual neural activity, right?\n0:41:35.890 --> 0:41:43.500\nCarly Fox\nAnd that's somethin
g that I can pretty quickly ascertain from looking at these top words for ones where it's less clear.\n0:41:43.770 --> 0:41:46.980\nCar
ly Fox\nWe will look at an example again of how we can leverage chat.\n0:41:46.990 --> 0:41:52.360\nCarly Fox\nGPT to help us get some
topic labels that pertain to these.\n0:41:52.650 --> 0:42:5.970\nCarly Fox\nSo we'll get back to that well next look at our umm, the to
p topics for each of our documents, OK, by creating a document topic matrix.\n0:42:6.340 --> 0:42:10.440\nCarly Fox\nSo this one will b
e pretty simple.\n0:42:10.650 --> 0:42:17.390\nCarly Fox\nWe're just gonna be bringing in our document topics object and displaying tha
t as a data frame.\n0:42:19.110 --> 0:42:20.170\nCarly Fox\nOops, sorry guys.\n0:42:20.680 --> 0:42:20.960\nCarly Fox\nGo away.\n0:42:2
3.160 --> 0:42:23.480\nCarly Fox\nThere we go.\n0:42:23.780 --> 0:42:25.380\nCarly Fox\nOK, awesome.\n0:42:25.550 --> 0:42:30.100\nCarl
y Fox\nSo here we see our document topic matrix.\n0:42:30.630 --> 0:42:35.480\nCarly Fox\nWhat we have here is our rows, our all of our
documents and our is our columns.\n0:42:35.490 --> 0:42:48.360\nCarly Fox\nWe have our topics and what we have is our values are sort o
f the probability of that topic existing within a given document.\n0:42:48.500 --> 0:42:51.270\nCarly Fox\nSo based off of this, what w
ould we say?\n0:42:51.780 --> 0:42:56.830\nCarly Fox\nFor our first document, what is the topic that is most dominant?\n0:43:3.960 -->
0:43:4.200\nTim Williams\nYou.\n0:43:6.710 --> 0:43:7.110\nCarly Fox\nWhich one?\n0:43:7.980 --> 0:43:9.960\nTim Williams\nTwo, the.\n
0:43:11.40 --> 0:43:13.290\nCarly Fox\nSo for so I'm talking about.\n0:43:13.300 --> 0:43:15.290\nCarly Fox\nSorry for the first one.\n
0:43:15.500 --> 0:43:15.850\nCarly Fox\nYeah.\n0:43:14.670 --> 0:43:16.300\nAva Whittington\nNo 14.\n0:43:13.890 --> 0:43:17.590\nTim W
illiams\nOhh document sorry ohh 14.\n0:43:15.860 --> 0:43:18.710\nCarly Fox\nYeah, sorry, 14.\n0:43:18.120 --> 0:43:18.910\nChase Jense
n\nTea, yeah.\n0:43:18.720 --> 0:43:19.330\nCarly Fox\nYeah, yeah, yeah.\n0:43:19.340 --> 0:43:20.290\nCarly Fox\nSorry if that was con
fusing.\n0:43:20.300 --> 0:43:20.490\nCarly Fox\nYeah.\n0:43:20.500 --> 0:43:29.400\nCarly Fox\nSo we're looking to, this is our docume
nt here and we can see that pretty much only topic 14 comes up in this document.\n0:43:29.410 --> 0:43:44.330\nCarly Fox\nThere is a li
ttle bit slight probability of topic 6 as well, but this one is almost entirely made up of what his topic 14, whereas we can see if the
re are some other documents that are split up a bit more.\n0:43:44.340 --> 0:44:0.710\nCarly Fox\nSo if we look at document two, we see
that we have a little bit from topic one and topic two and then we have a higher probability of topic 6 as well has the highest probabi
lity of topic 17, right?\n0:44:0.720 --> 0:44:19.60\nCarly Fox\nSo we can see again and idea of how these documents are composed, which
ones are discussing multiple topics, which ones are only sort of mainly discussing one topic and so on to see what is this document ove
rall about right.\n0:44:21.440 --> 0:44:25.130\nCarly Fox\nAnd we will examine a paper in just a second.\n0:44:25.140 --> 0:44:36.810\n
Carly Fox\nWe'll look at that first document to see if topic 14 appears to sort of match up with what is being talked about.\n0:44:37.3
20 --> 0:44:51.800\nCarly Fox\nAnd in document one, OK, so first let's get a better understanding of maybe what topic 14 is OK and we'l

20 --> 0:44:51.860\nCarly Fox\nAnd in document one, OK, so first let's get a better understanding of maybe what topic 14 is OK and we'
l do that now by examining the most prototypical document per topic.\n0:44:51.870 --> 0:44:56.330\nCarly Fox\nOK, that's gonna give us
some additional context about what our topics are about.\n0:44:56.530 --> 0:45:4.400\nCarly Fox\nSo I'm just gonna go ahead again and a
dd in comments to discuss what each of these steps are doing.\n0:45:4.410 --> 0:45:20.570\nCarly Fox\nSo here we're going to just be ex
tracting the rows with the highest with the Max scores for each of our topics, one through 20.\n0:45:21.460 --> 0:45:25.840\nCarly Fox
\nWell, then go ahead and save the index number for those rows.\n0:45:28.760 --> 0:45:33.850\nCarly Fox\nWe're then going to go ahead a
nd save their score, sort of their probability value.\n0:45:37.230 --> 0:45:41.590\nCarly Fox\nAnd well then locate the document.\n0:4
5:41.600 --> 0:45:50.640\nCarly Fox\nThat's actually associated with that index number for each topic, and then we're going to just go
ahead and save those as a list.\n0:45:52.490 --> 0:46:9.770\nCarly Fox\nOK, so that way you can then again create a data frame that's g
onna show us what the topic is, the score, the paper number, and then we're going to actually be able to view the top terms for that gi
ven topic.\n0:46:9.780 --> 0:46:13.820\nCarly Fox\nAnd then the the text from the paper that is the top paper here.\n0:46:15.280 --> 0:
46:17.260\nCarly Fox\nCat. Right?\n0:46:18.250 --> 0:46:31.800\nCarly Fox\nSo here we can see for topic one our topic is of course topi
c one, the most dominant paper for topic one is paper #33 and I can see the actual paper that matches with that topic.\n0:46:32.50 -->
0:46:34.920\nCarly Fox\nAnd of course, paired next to it, we also have four topic.\n0:46:34.930 --> 0:46:38.420\nCarly Fox\nOne these a
re the top features that describe that topic.\n0:46:38.770 --> 0:46:49.210\nCarly Fox\nSo if I wanted to now go down to topic 14, which
we was our first document, our first document was labeled as primarily topic 14.\n0:46:50.10 --> 0:47:1.990\nCarly Fox\nI can Scroll do
wn here and again I could probably figure this out myself if with enough looking, but this is a place again where I can leverage chat G
PT to take out some of the.\n0:47:3.460 --> 0:47:3.780\nCarly Fox\nSort of.\n0:47:3.790 --> 0:47:25.940\nCarly Fox\nAnalysis thematic a
nalysis that I would do manually, I can go ahead and just copy my top features and I can copy this sort of sample of the text and you c
ould do this with multiple samples of documents that fell under this topic, but I found, at least in this case that it's it worked pret
ty well with even just one example.\n0:47:27.80 --> 0:47:29.420\nCarly Fox\nI can go to my chat TBT and say.\n0:47:30.530 --> 0:47:34.7
0\nCarly Fox\nUmm, I'm working with conference papers.\n0:47:38.860 --> 0:47:40.140\nCarly Fox\nFrom your apps.\n0:47:41.650 --> 0:47:5
2.480\nCarly Fox\nUh, here are the top he features and some example text from a document.\n0:47:53.870 --> 0:47:59.230\nCarly Fox\nUmm
that was dominant for this topic.\n0:48:1.440 --> 0:48:4.100\nCarly Fox\nCan you provide a topic label?\n0:48:8.670 --> 0:48:8.850\nCar
ly Fox\nThis.\n0:48:15.270 --> 0:48:19.540\nCarly Fox\nOK so here it gives me neural network modeling of sleep and dreaming.\n0:48:19.6
10 --> 0:48:21.500\nCarly Fox\nNeural psychology of cognitive theory.\n0:48:21.910 --> 0:48:28.50\nCarly Fox\nIf you don't want to take
it at its word, you can always be like, provide an explanation so you can understand its logic.\n0:48:31.850 --> 0:48:32.870\nCarly Fox
\nOK, so it'll give you.\n0:48:33.830 --> 0:48:42.160\nCarly Fox\nBased off of the provided features text mentioned, several key featur
es related to sleep and dreaming such as sleep R.E.M.\n0:48:42.230 --> 0:48:42.750\nCarly Fox\nSequencing.\n0:48:43.940 --> 0:48:45.790
\nCarly Fox\nI don't know all of these terms.\n0:48:45.800 --> 0:48:57.770\nCarly Fox\nAwake model, non rapid eye movement and so on an
d it will give you this nice long explanation to convince you that this is a good label, which again still worth manually checking.\n0:
48:57.780 --> 0:49:8.970\nCarly Fox\nMaybe a couple examples to make sure that this is lining up, but we can go ahead and do that now t
o see if we think that that sort of matches up with the text that's in that first paper.\n0:49:9.80 --> 0:49:13.60\nCarly Fox\nSo we'll
say this is our topic label.\n0:49:15.90 --> 0:49:17.670\nCarly Fox\nJust add that in down here so we remember.\n0:49:24.930 --> 0:49:2
9.570\nCarly Fox\nOK, neural network modeling of sleep and dreaming, neuropsychology and cognitive theory.\n0:49:30.850 --> 0:49:34.30
\nCarly Fox\nAnd now I can go back up here.\n0:49:34.570 --> 0:49:38.190\nCarly Fox\nAnd that was my first paper that was primarily top
ic 14.\n0:49:40.220 --> 0:49:44.830\nCarly Fox\nSo we'll do that and I'm not gonna pull up the full paper because it's pretty long.\n0:
49:44.840 --> 0:49:49.470\nCarly Fox\nI'll just do the 1st 500 characters and we can see if that seems to match up.\n0:49:51.30 --> 0:4
9:56.920\nCarly Fox\nSo here we have sleep, dreaming neuropsychology.\n0:49:59.550 --> 0:50:0.610\nCarly Fox\nWaking sleep, brain.\n0:5
0:3.760 --> 0:50:4.630\nCarly Fox\nYou're on processes.\n0:50:9.830 --> 0:50:11.490\nCarly Fox\nNeural network model.\n0:50:11.550 -->
0:50:14.30\nCarly Fox\nOhh it's all right.\n0:50:14.80 --> 0:50:23.350\nCarly Fox\nSo does seem to be a pretty good fit for our topic d
own here, which was neural network modeling of sleep and dreaming.\n0:50:23.420 --> 0:50:32.610\nCarly Fox\nThose were terms that all c
ame up in that paper, right mentioned of neural network, mentioning of sleep and dreaming, mentioning of neuropsychology.\n0:50:32.780
--> 0:50:33.270\nCarly Fox\nRight.\n0:50:33.310 --> 0:50:38.430\nCarly Fox\nAnd so this wasn't even the top paper that represents this
topic.\n0:50:38.900 --> 0:50:45.700\nCarly Fox\nIt's just a paper that has that topic come up, but it does seem that that topic does a
good job of representing representing that paper.\n0:50:46.580 --> 0:50:55.820\nCarly Fox\nSo great way again if you're dealing with lo
ng papers right where we're expecting there to be.\n0:50:55.860 --> 0:50:59.930\nCarly Fox\nAgain, multiple themes that are coming up,
this is a great way to begin.\n0:50:59.940 --> 0:51:14.810\nCarly Fox\nTo summarize that information and to understand what are some of
the key themes and topics that are coming up either across an entire corpus or given a specific document of interest so.\n0:51:17.700 -
-> 0:51:23.530\nCarly Fox\nWe do still have a little bit of time, wasn't sure how long this demo was going to take to run through.\n0:5
1:23.540 --> 0:51:28.280\nCarly Fox\nI feel like I've gotten a little overambitious with some of the demos I've done before.\n0:51:28.4
40 --> 0:51:57.890\nCarly Fox\nWe'll probably get it done with this one a little bit longer, but since we have the time, if people want
to stick around to have a little bit of a discussion or to talk about any questions that you have about this method, one of the things
I thought would be maybe youthful to talk about is just sort of understanding where you guys think that you might be able to apply this
sort of method or if you have any questions about where like you and your own personal work could apply topic modeling.\n0:51:58.100 --
> 0:51:59.400\nCarly Fox\nSo I'll open the floor to that.\n0:52:5.620 --> 0:52:8.30\nTodd Hesson\nHe mentioned that LA takes a little b
it longer.\n0:52:8.280 --> 0:52:8.480\nCarly Fox\nUh-huh.\n0:52:8.100 --> 0:52:15.940\nTodd Hesson\nIs it just because it's an older mo
del or does it do things differently like so what's the difference between to?\n0:52:15.460 --> 0:52:16.310\nCarly Fox\nYeah.\n0:52:16.
560 --> 0:52:17.50\nCarly Fox\nYeah.\n0:52:17.60 --> 0:52:23.590\nCarly Fox\nSo the end product is is very similar between LDA and nega
tive matrix factorization.\n0:52:23.720 --> 0:52:28.970\nCarly Fox\nThe sort of main difference is the sort of the calculations that ar
e happening on the back end.\n0:52:29.280 --> 0:52:48.820\nCarly Fox\nSo LDA is a probability based method, so it has to again sort of
calculate all of these probabilities between words Co occurring with across all of our different documents and sort of do this iterativ
ely of doing again all of these different probability calculations.\n0:52:49.700 --> 0:52:56.650\nCarly Fox\nNegative matrix factorizat
ion relies on methods from linear algebra, and it does matrix.\n0:52:56.740 --> 0:53:0.550\nCarly Fox\nDecomposition, which is just i
t's just faster.\n0:53:0.620 --> 0:53:14.960\nCarly Fox\nSo it's again sort of the end products are the same, but the math that underli
es the methods are different and matrix decomposition is just faster than this sort of calculation of all these like probability distri
butions, if that makes sense.\n0:53:16.880 --> 0:53:17.750\nTodd Hesson\nThat makes sense.\n0:53:23.210 --> 0:53:24.20\nCarly Fox\nI do
n't think so.\n0:53:17.860 --> 0:53:24.460\nTodd Hesson\nSo is there any scenario where we'd want to use one over the other, or just i
t's always?\n0:53:24.30 --> 0:53:29.460\nCarly Fox\nI mean, yeah, I I think as always it it sort of depends.\n0:53:29.470 --> 0:53:38.4
00\nCarly Fox\nI mean, I I I would say again the main benefit of this method is that it's fast, at least comparative to LDA.\n0:53:39.1
10 --> 0:54:0.900\nCarly Fox\nSome of the downsides of using negative matrix factorization is that some of those methods I talked about
before, like the topic coherence, you can't do that with negative matrix factorization because the topic coherence population is based
off of probability distributions and you're you're not working with probability distributions and MSFT.\n0:54:0.910 --> 0:54:6.910\nCar
ly Fox\nSo if you wanna be able to calculate topic coherence and you would need to use LDA dough.\n0:54:7.150 --> 0:54:14.630\nCarly Fo
x\nBut if you don't care about that and you just want it to be fast, then I would use negative matrix factorization.\n0:54:16.440 -->
0:54:18.150\nTodd Hesson\nFinished process.\n0:54:17.340 --> 0:54:19.140\nCarly Fox\nYeah, cool.\n0:54:32.490 --> 0:54:40.660\nCarly Fo
x\nAnother place that might be useful, I know that people have brought up before, like open-ended customer feedback surveys.\n0:54:41.6
80 --> 0:54:45.950\nCarly Fox\nSo this would be another place where that would topic modeling would be useful.\n0:54:46.800 --> 0:54:5
3.530\nCarly Fox\nI mean, I guess, unless I I don't, I guess know how long your customer feedback responses tend to be.\n0:54:53.820 --
> 0:55:2.10\nCarly Fox\nBut I feel like normally in things like customer reviews, customer feedback, there will tend to be more than on
e thing that gets brought up.\n0:55:2.200 --> 0:55:15.420\nCarly Fox\nSo this would be another place where again, if you have a large a
mount of open ended responses where you're expecting more than one topic to come up, this is a great way to sort of understand what are
the common themes that are coming up.\n0:55:15.550 --> 0:55:31.160\nCarly Fox\nAnd then to be able to group documents based off of sort
of maybe topics of concern that are coming up or umm topics related to things you're doing well and so on, I'm just sort of summarize t
hat information tell that's another place would be useful.\n0:55:33.80 --> 0:55:34.250\nTodd Hesson\nThat's a good point.\n0:55:34.260

```
--> 0:55:36.830\nTodd Hesson\nHow long was the document typically need to be?\n0:55:36.920 --> 0:55:41.410\nTodd Hesson\nYou know, like
it if it's like two or three sentences, is it worth doing a topic model?\n0:55:41.420 --> 0:55:45.790\nTodd Hesson\nDoes it have to be
a whole 500 page paper or something like that?\n0:55:46.510 --> 0:55:50.550\nCarly Fox\nIt is so it doesn't need to be 500 pages by any
means like it.\n0:55:50.560 --> 0:55:53.120\nCarly Fox\nIt can definitely work with shorter form documents, but.\n0:55:55.470 --> 0:56:
4.600\nCarly Fox\nBasically the the the key thing is that has to be long enough for there to be multiple topics that come up.\n0:56:5.6
00 --> 0:56:12.720\nCarly Fox\nOtherwise, there's not really much of a point in doing topic modeling versus something like document clu
stering, right?\n0:56:12.730 --> 0:56:22.40\nCarly Fox\nBecause if you're only expecting your document to be about like one particular
thing, then you might as well just cluster them and then you can sort of label those clusters.\n0:56:22.980 --> 0:56:37.870\nCarly Fox
\nThe sort of main benefit of topic modeling is that it recognizes that a document can be about more than one thing, where whereas docu
ment clustering doesn't really recognize that, it's really just grouping together similar documents, if that makes sense.\n0:56:38.290
--> 0:56:43.690\nCarly Fox\nSo if it is short, I probably would just default to to document clustering over topic modeling.\n0:56:47.63
0 --> 0:56:47.900\nCarly Fox\nUmm.\n0:56:45.330 --> 0:56:51.600\nTodd Hesson\nSo like your customer feedback form or CSAT scores like t
hat would probably better for clustering than a topic modeling.\n0:56:51.870 --> 0:56:53.100\nTodd Hesson\nIf it's only a couple senten
ces.\n0:56:52.200 --> 0:56:54.840\nCarly Fox\nYet yeah, I would say so.\n0:56:54.850 --> 0:56:56.540\nCarly Fox\nAnd and there's nothin
g to stop you from.\n0:56:56.550 --> 0:57:25.580\nCarly Fox\nLike after you've done document clustering, if you then want to like, mayb
e you could apply topic modeling to those specific cluster groupings to say OK, maybe there are multiple topics that are coming up amon
gst these clusters, but so you could always apply it after the fact that I would maybe start with document clustering if it's if it's s
hort just because you probably aren't going to get super useful topics out of it.\n0:57:25.590 --> 0:57:31.90\nCarly Fox\nIf it's, if i
t's too short, it might just sort of be like really generic words so.\n0:57:32.970 --> 0:57:33.350\nTodd Hesson\nMakes sense?\n0:57:33.
700 --> 0:57:34.310\nCarly Fox\nYeah.\n0:57:34.440 --> 0:57:35.20\nCarly Fox\nGood questions.\n0:57:46.400 --> 0:57:47.380\nBraxton Ree
d\nI have a question.\n0:57:47.850 --> 0:57:47.990\nCarly Fox\nSure.\n0:57:53.790 --> 0:57:54.20\nCarly Fox\nMm-hmm.\n0:57:54.370 -->
0:58:7.290\nBraxton Reed\nHave lots of overlap like they they're conceptually very similar, but you want to show that distinction if th
at makes sense.\n0:58:8.170 --> 0:58:22.290\nCarly Fox\nUmm, so are you saying that they're like similar in that like there's sort of l
ike maybe like words that have, like, multiple meanings that you're trying to maybe?\n0:58:22.300 --> 0:58:23.490\nCarly Fox\nMaybe if
```

Apply some Filtering & Cleaning!

```
def filter_speaker(text, target_speaker):
    filtered_lines = []
    lines = text.split('\n') # split lines list elements
    i = 0
    while i < len(lines):
        speaker = lines[i].strip() # the speaker line comes first
        if speaker.startswith(target_speaker):
            dialogue = lines[i+1].strip()
            if dialogue:
                filtered_lines.append(dialogue)
        i += 2  # Skip to the next speaker line
    return '\n'.join(filtered_lines)


def preprocess_text(text):
    text = contractions.fix(text)

    sentences = sent_tokenize(text)

    cleaned_sentences = []
    stop_words = set(stopwords.words('english'))

    for sentence in sentences:
        words = word_tokenize(sentence.lower()) # tokenize words within each sentence and set to lowercase
        cleaned_words = [re.sub(r'[^\w\s]', '', word) for word in words if word not in stop_words]
        cleaned_sentence = ' '.join(cleaned_words)
        cleaned_sentences.append(cleaned_sentence)

    return cleaned_sentences



target_speaker = 'Carly Fox'
filtered_text = filter_speaker(transcript_text,target_speaker)

# Print the filtered transcript
print(filtered_text)
```

Run through TextRank steps: step 1 -- preprocess text, tokenize at sentence level

```
sentences = preprocess_text(filtered_text)
sentences[0:5]
```

```
['today  going talking topic modeling  pretty cool unsupervised method ',
 'always  start brief lecture component topic modeling  demonstration python afterwards ',
 'instead ',
 'right ',
 'sorry guys ']
```

step 2 -- vectorize to dtm

step 3 -- compute cosine similarity matrix

```
vectorizer = TfidfVectorizer()
sentence_vectors = vectorizer .fit_transform(sentences)

# Create similarity matrix
similarity_matrix = cosine_similarity(sentence_vectors,sentence_vectors)
similarity_df = pd.DataFrame(similarity_matrix, index=range(len(sentences)), columns=range(len(sentences)))
similarity_df
```

	0	1	2	3	4	5	6	7	8	9
0	1.000000	0.089758	0.0	0.0	0.000000	0.331573	0.163238	0.000000	0.055148	0.12196
1	0.089758	1.000000	0.0	0.0	0.000000	0.270704	0.111004	0.000000	0.000000	0.00000
2	0.000000	0.000000	1.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
3	0.000000	0.000000	0.0	1.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
4	0.000000	0.000000	0.0	0.0	1.000000	0.000000	0.152640	0.000000	0.000000	0.00000
...	...	...	...	...	...	...	...	...	...	...
221	0.000000	0.000000	0.0	0.0	0.383379	0.000000	0.000000	0.000000	0.116874	0.00000
222	0.000000	0.000000	0.0	0.0	0.341557	0.000000	0.000000	0.000000	0.000000	0.00000
223	0.097326	0.000000	0.0	0.0	0.000000	0.000000	0.082121	0.189953	0.055633	0.12303
224	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.143590	0.00000
225	0.000000	0.000000	0.0	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000

226 rows × 226 columns

step 4 -- apply pagerank (textrank) algorithm to score sentences

step 5 -- rank sentences

```
scores = nx.pagerank(nx.from_numpy_array(similarity_matrix)) # add argument

ranked_sentences = sorted(((scores[i], sentence) for i, sentence in enumerate(sentences)), reverse=True)
ranked_sentences[0:5] # view top sentences
```

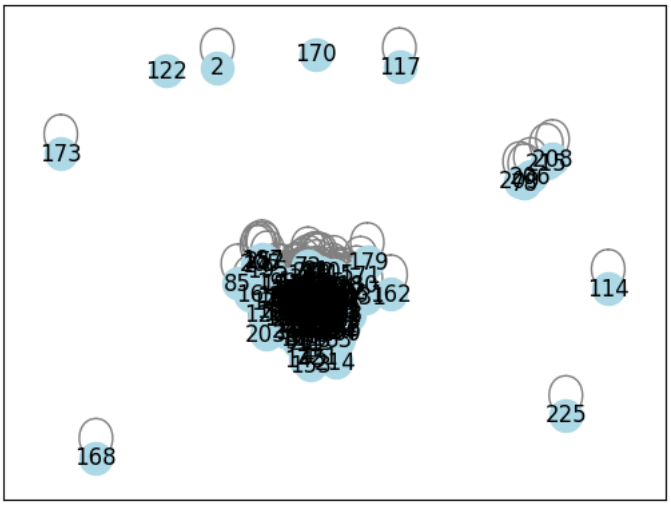
```
[(0.008858034899939402,
  'terms show multiple topics would say like coming multiple topics  going sort differentiating topics going come sort words umm exists
within topic makes sense '),
 (0.008734187475664434,
  'examine top terms topic begin understand  ok  topics also examine top topic per document say  ok  given document  topics seem
compose right make document '),
 (0.008269451707715939,
  'look document two  see little bit topic one topic two higher probability topic 6 well highest probability topic 17  right '),
 (0.0080506827989594,
  'document one  ok  first let us get better understanding maybe topic 14 ok examining prototypical document per topic '),
 (0.007876176467554875,
  'highest probability topics coming document iteratively sort stopped reassigning things sort like k means clustering analysis would
point conversions  end get output probability given word belongs topic  probability topic within document ')]
```

Put it all together into one function, return top (4) sentences in order

```
def textrank_summarize(text, top_n=4):
    original_sentences = sent_tokenize(text)
    sentences = preprocess_text(text)

    vectorizer = TfidfVectorizer()
    sentence_vectors = vectorizer.fit_transform(sentences)

    similarity_matrix = cosine_similarity(sentence_vectors, sentence_vectors)

    scores = nx.pagerank(nx.from_numpy_array(similarity_matrix))

    ranked_sentences = sorted(((scores[i], i) for i in range(len(original_sentences))), reverse=True)

    summary_indices = [index for _, index in ranked_sentences[:top_n]]
    summary = [original_sentences[i] for i in sorted(summary_indices)]

    G = nx.from_numpy_array(similarity_matrix)
    pos = nx.spring_layout(G)
    nx.draw_networkx(G, pos=pos, with_labels=True, node_color='lightblue', edge_color='gray')
    plt.show()

    return ' '.join(summary)
```

Apply to text, also generate visualization of graph

```
# Perform summarization using Textrank
summary = textrank_summarize(filtered_text)

# Print the summary
summary
```




```
'So we can examine what are the top terms in our topic so that we can begin to understa
nd, OK, what are these topics we can also examine the top topic per document to say, O
K, for a given document, what are the topics that seem to compose it right to make up t
hat document. So if we look at document two, we see that we have a little bit from topi
c one and topic two and then we have a higher probability of topic 6 as well has the hi
ghest probability of topic 17, right? And in document one, OK, so first let's get a bet
ter understanding of maybe what topic 14 is OK and we'll do that now by examining the m
ost prototypical document per topic. So you can have terms show up in multiple topics a
```

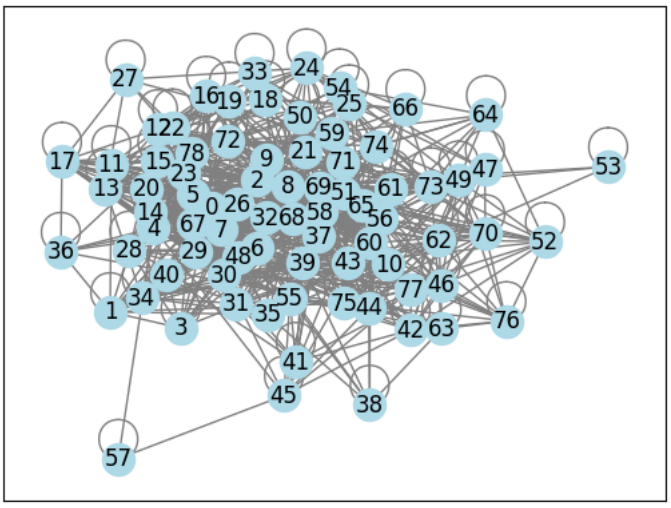
Can compare to ChatGPT results or Augment with ChatGPT

Keep in mind that ChatGPT is currently token limited, so abstraction-based summarization of a document can be out of the question if it's too long...

Try on a Different Set of Text...a bit more Coherent

text2 = """Automatic summarization is the process of shortening a set of data computationally, to create a subset (a summary) that represents the most important or relevant information within the original content. Artificial intelligence algorithms are commonly developed and employed to achieve this, specialized for different types of data.
Text summarization is usually implemented by natural language processing methods, designed to locate the most informative sentences in a given document.[1] On the other hand, visual content can be summarized using computer vision algorithms. Image summarization is the subject of ongoing research; existing approaches typically attempt to display the most representative images from a given image collection, or generate a video that only includes the most important content from the entire collection.[2][3][4] Video summarization algorithms identify and extract from the original video content the most important frames (key-frames), and/or the most important video segments (key-shots), normally in a temporally ordered fashion.[5][6][7][8] Video summaries simply retain a carefully selected subset of the original video frames and, therefore, are not identical to the output of video synopsis algorithms, where new video frames are being synthesized based on the original video content.
Extraction-based summarization
Here, content is extracted from the original data, but the extracted content is not modified in any way. Examples of extracted content include key-phrases that can be used to "tag" or index a text document, or key sentences (including headings) that collectively comprise an abstract, and representative images or video segments, as stated above. For text, extraction is analogous to the process of skimming, where the summary (if available), headings and subheadings, figures, the first and last paragraphs of a section, and optionally the first and last sentences in a paragraph are read before one chooses to read the entire document in detail.[10] Other examples of extraction that include key sequences of text in terms of clinical relevance (including patient/problem, intervention, and outcome).[11]
Applications and systems for summarization
There are broadly two types of extractive summarization tasks depending on what the summarization program focuses on. The first is generic summarization, which focuses on obtaining a generic summary or abstract of the collection (whether documents, or sets of images, or videos, news stories etc.). The second is query relevant summarization, sometimes called query-based summarization, which summarizes objects specific to a query. Summarization systems are able to create both query relevant text summaries and generic machine-generated summaries depending on what the user needs.
An example of a summarization problem is document summarization, which attempts to automatically produce an abstract from a given document. Sometimes one might be interested in generating a summary from a single source document, while others can use multiple source documents (for example, a cluster of articles on the same topic). This problem is called multi-document summarization. A related application is summarizing news articles. Imagine a system, which automatically pulls together news articles on a given topic (from the web), and concisely represents the latest news as a summary. Image collection summarization is another application example of automatic summarization. It consists in selecting a representative set of images from a larger set of images.[13] A summary in this context is useful to show the most representative images of results in an image collection exploration system. Video summarization is a related domain, where the system automatically creates a trailer of a long video. This also has applications in consumer or personal videos, where one might want to skip the boring or repetitive actions. Similarly, in surveillance videos, one would want to extract important and suspicious activity, while ignoring all the boring and redundant frames captured. At a very high level, summarization algorithms try to find subsets of objects (like set of sentences, or a set of images), which cover information of the entire set. This is also called the core-set. These algorithms model notions like diversity, coverage, information and representativeness of the summary. Query based summarization techniques, additionally model for relevance of the summary with the query. Some techniques and algorithms which naturally model summarization problems are TextRank and PageRank, Submodular set function, Determinantal point process, maximal marginal relevance (MMR) etc. Unsupervised approach: TextRank Another keyphrase extraction algorithm is TextRank. While supervised methods have some nice properties, like being able to produce interpretable rules for what features characterize a keyphrase, they also require a large amount of training data. Many documents with known keyphrases are needed. Furthermore, training on a specific domain tends to customize the extraction process to that domain, so the resulting classifier is not necessarily portable, as some of Turney's results demonstrate. Unsupervised keyphrase extraction removes the need for training data. It approaches the problem from a different angle. Instead of trying to learn explicit features that characterize keyphrases, the TextRank algorithm[16] exploits the structure of the text itself to determine keyphrases that appear "central" to the text in the same way that PageRank selects important Web pages. Recall this is based on the notion of "prestige" or "recommendation" from social networks. In this way, TextRank does not rely on any previous training data at all, but rather can be run on any arbitrary piece of text, and it can produce output simply based on the text's intrinsic properties. Thus the algorithm is easily portable to new domains and languages. TextRank is a general purpose graph-based ranking algorithm for NLP. Essentially, it runs PageRank on a graph specially designed for a particular NLP task. For keyphrase extraction, it builds a graph using some set of text units as vertices. Edges are based on some measure of semantic or lexical similarity between the text unit vertices. Unlike PageRank, the edges are typically undirected and can be weighted to reflect a degree of similarity. Once the graph is constructed, it is used to form a stochastic matrix, combined with a damping factor (as in the "random surfer model"), and the ranking over vertices is obtained by finding the eigenvector corresponding to eigenvalue 1 (i.e., the stationary distribution of the random walk on the graph). The vertices should correspond to what we want to rank. Potentially, we could do something similar to the supervised methods and create a vertex for each unigram, bigram, trigram, etc. However, to keep the graph small, the authors decide to rank individual unigrams in a first step, and then include a second step that merges highly ranked adjacent unigrams to form multi-word phrases. This has a nice side effect of allowing us to produce keyphrases of arbitrary length. For example, if we rank unigrams and find that "advanced", "natural", "language", and "processing" all get high ranks, then we would look at the original text and see that these words appear consecutively and create a final keyphrase using all four together. Note that the unigrams placed in the graph can be filtered by part of speech. The authors found that adjectives and nouns were the best to include. Thus, some linguistic knowledge comes into play in this step. Edges are created based on word co-occurrence in this application of TextRank. Two vertices are connected by an edge if the unigrams appear within a window of size N in the original text. N is typically around 2–10. Thus, "natural" and "language" might be linked in a text about NLP. "Natural" and "processing" would also be linked because they would both appear in the same string of N words. These edges build on the notion of "text cohesion" and the idea that words that appear near each other are likely related in a meaningful way and "recommend" each other to the reader. Since this method simply ranks the individual vertices, we need a way to threshold or produce a limited number of keyphrases. The technique chosen is to set a count T to be a user-specified fraction of the total number of vertices in the graph. Then the top T vertices/unigrams are selected based on their stationary probabilities. A post- processing step is then applied to merge adjacent instances of these T unigrams. As a result, potentially more or less than T final keyphrases will be produced, but the number should be roughly proportional to the length of the original text.It is not initially clear why applying PageRank to a co-occurrence graph would produce useful keyphrases. One way to think about it is the following. A word that appears multiple times throughout a text may have many different co-occurring neighbors. For example, in a text about machine learning,

```
the unigram "learning" might co-occur with "machine", "supervised", "un-supervised", and "semi-supervised" in four different
sentences. Thus, the "learning" vertex would be a central "hub" that connects to these other modifying words. Running
PageRank/TextRank on the graph is likely to rank "learning" highly. Similarly, if the text contains the phrase "supervised
classification", then there would be an edge between "supervised" and "classification". If "classification" appears several
other places and thus has many neighbors, its importance would contribute to the importance of "supervised". If it ends
up with a high rank, it will be selected as one of the top T unigrams, along with "learning" and probably "classification".
In the final post-processing step, we would then end up with keyphrases "supervised learning" and "supervised classification".
In short, the co-occurrence graph will contain densely connected regions for terms that appear often and in different contexts.
A random walk on this graph will have a stationary distribution that assigns large probabilities to the terms in the centers
of the clusters. This is similar to densely connected Web pages getting ranked highly by PageRank. This approach has also been
used in document summarization, considered below."""


summary = textrank_summarize(text2)

# Print the summary
summary
```




```
'Automatic summarization is the process of shortening a set of data computationally, to
create a subset (a summary) that\nrepresents the most important or relevant information
within the original content. An example of a summarization problem is document summariz
ation, which attempts to automatically produce an abstract from\na given document. For
keyphrase extraction,\nit builds a graph using some set of text units as vertices. For
example, if we rank unigrams and find that "advanced", "natural" \n"language", and "pro
```