Retrieving Wikipedia articles

In this module, we focused on using nearest neighbors and clustering to retrieve documents that interest users, by analyzing their text. We explored two document representations: word counts and TF-IDF. We also built an iPython notebook for retrieving articles from Wikipedia about famous people.

In this assignment, we are going to dig deeper into this application, explore the retrieval results for various famous people, and familiarize ourselves with the code needed to build a retrieval system. These techniques will be key to building the intelligent application in your capstone project.

Follow the rest of the instructions on this page to complete your program. When you are done, **instead of uploading your code, you will answer a series of quiz questions** (see the quiz after this reading) to document your completion of this assignment. The instructions will indicate what data to collect for answering the quiz.

Learning outcomes

- Execute document retrieval code with the iPython notebook
- Load and transform real, text data
- Compare results with word counts and TF-IDF
- Set the distance function in the retrieval
- Build a document retrieval model using nearest neighbor search

Resources you will need

You will need to install the software tools or use the free Amazon EC2 machine. Instructions for both options are provided in the reading for Module 1.

Download the data and starter code

Before getting started, you will need to download the dataset and the starter iPython notebook that we used in the module.

- Download the wikipedia dataset with articles on famous people here in SFrame format: **people_wiki.gl.zip**
- Download the document retrieval notebook from the module here: **document-retrieval.ipynb**
- Save both of these files in the same directory (where you are calling iPython notebook from) and unzip the data file. **Not sure where to save the files? See this guide.**

Now you are ready to get started!

Note: If you would rather use other ML tools...

You are welcome to use any ML tool for this course, such as scikit-learn. Though, as discussed in the intro module, *we strongly recommend you use IPython Notebook and GraphLab Create. (GraphLab Create is free for academic purposes.)*

If you are choosing to use other packages, we still recommend you use SFrame, which will allow you to scale to much larger datasets than Pandas. (Though, it's possible to use Pandas in this course, if your machine has sufficient memory.) The SFrame package is available in open-source under a permissive BSD license. So, you will always be able to use SFrames for free.

If you are not using SFrame, here is the dataset for this assignment in CSV format, so you can use Pandas or other options out there: people_wiki.csv

Watch the video and explore the iPython notebook on retrieving wikipedia articles

If you haven't done so yet, before you start, we recommend you watch the video where we go over the iPython notebook on retrieving documents from this module. You can then open up the iPython notebook we used and familiarize yourself with the steps we covered in this example.

What you will do

Now you are ready! We are going do three tasks in this assignment. There are several results you need to gather along the way to enter into the quiz after this reading.

1. **Compare top words according to word counts to TF-IDF:** In the notebook we covered in the module, we explored two document representations: word counts and TF-IDF. Now, take a particular famous person, 'Elton John'. What are the 3 words in his articles with highest word counts? What are the 3 words in his articles with highest TF-IDF? These results illustrate why TF-IDF is useful for finding important words. **Save these results to answer the quiz at the end.**

2. **Measuring distance:** Elton John is a famous singer; let's compute the distance between his article and those of two other famous singers. In this assignment, you will use the *cosine distance*, which one measure of similarity between vectors, similar to the one discussed in the lectures. You can compute this distance using the *graphlab*.distances.cosine function. What's the cosine distance between the articles on 'Elton John' and 'Victoria Beckham'? What's the cosine distance between the articles on 'Elton John' and Paul McCartney'? Which one of the two is closest to Elton John? Does this result make sense to you? **Save these results to answer the quiz at the end.**

3. **Building nearest neighbors models with different input features and setting the distance metric:** In the sample notebook, we built a nearest neighbors model for retrieving articles using TF-IDF as features and using the default setting in the construction of the nearest neighbors model. Now, you will build two nearest neighbors models:

- Using word counts as features
- Using TF-IDF as features

In both of these models, we are going to set the distance function to cosine similarity. Here is how: when you call the function

```
graphlab.nearest_neighbors.create
```

add the parameter:

```
distance='cosine'
```

Now we are ready to use our model to retrieve documents. Use these two models to collect the following results:

- What's the most similar article, other than itself, to the one on 'Elton John' using word count features?
- What's the most similar article, other than itself, to the one on 'Elton John' using TF-IDF features?
- What's the most similar article, other than itself, to the one on 'Victoria Beckham' using word count features?
- What's the most similar article, other than itself, to the one on 'Victoria Beckham' using TF-IDF features?

Save these results to answer the quiz at the end.