First off, if you want to extract count features and apply TF-IDF normalization and row-wise euclidean normalization you can do it in one operation with `TfidfVectorizer`:

```
>>> from sklearn.feature_extraction.text import TfidfVectorizer
>>> from sklearn.datasets import fetch_20newsgroups
>>> twenty = fetch_20newsgroups()

>>> tfidf = TfidfVectorizer().fit_transform(twenty.data)
>>> tfidf
<11314x130088 sparse matrix of type '<type 'numpy.float64'>'
    with 1787553 stored elements in Compressed Sparse Row format>
```

Now to find the cosine distances of one document (e.g. the first in the dataset) and all of the others you just need to compute the dot products of the first vector with all of the others as the tfidf vectors are already row-normalized.

As explained by Chris Clark in comments and here Cosine Similarity does not take into account the magnitude of the vectors. Row-normalised have a magnitude of 1 and so the Linear Kernel is sufficient to calculate the similarity values.

The scipy sparse matrix API is a bit weird (not as flexible as dense N-dimensional numpy arrays). To get the first vector you need to slice the matrix row-wise to get a submatrix with a single row:

```
>>> tfidf[0:1]
<1x130088 sparse matrix of type '<type 'numpy.float64'>'
    with 89 stored elements in Compressed Sparse Row format>
```

scikit-learn already provides pairwise metrics (a.k.a. kernels in machine learning parlance) that work for both dense and sparse representations of vector collections. In this case we need a dot product that is also known as the linear kernel:

```
>>> from sklearn.metrics.pairwise import linear_kernel
>>> cosine_similarities = linear_kernel(tfidf[0:1], tfidf).flatten()
>>> cosine_similarities
array([ 1.        ,  0.04405952,  0.11016969, ...,  0.04433602,
    0.04457106,  0.03293218])
```

Hence to find the top 5 related documents, we can use `argsort` and some negative array slicing (most related documents have highest cosine similarity values, hence at the end of the sorted indices array):

```
>>> related_docs_indices = cosine_similarities.argsort()[:-5:-1]
>>> related_docs_indices
array([    0,   958, 10576,  3277])
>>> cosine_similarities[related_docs_indices]
array([ 1.        ,  0.54967926,  0.32902194,  0.2825788 ])
```

```
array([ 1.        ,  0.04967928,  0.05290294,  0.28035785])
```

The first result is a sanity check: we find the query document as the most similar document with a cosine similarity score of 1 which has the following text:

```
>>> print twenty.data[0]
From: lerxst@wam.umd.edu (where's my thing)
Subject: WHAT car is this!?
Nntp-Posting-Host: rac3.wam.umd.edu
Organization: University of Maryland, College Park
Lines: 15

 I was wondering if anyone out there could enlighten me on this car I saw
the other day. It was a 2-door sports car, looked to be from the late 60s/
early 70s. It was called a Bricklin. The doors were really small. In addition,
the front bumper was separate from the rest of the body. This is
all I know. If anyone can tellme a model name, engine specs, years
of production, where this car is made, history, or whatever info you
have on this funky looking car, please e-mail.

Thanks,
- IL
   ---- brought to you by your neighborhood Lerxst ----
```

The second most similar document is a reply that quotes the original message hence has many common words:

```
>>> print twenty.data[958]
From: rseymour@reed.edu (Robert Seymour)
Subject: Re: WHAT car is this!?
Article-I.D.: reed.1993Apr21.032905.29286
Reply-To: rseymour@reed.edu
Organization: Reed College, Portland, OR
Lines: 26

In article <1993Apr20.174246.14375@wam.umd.edu> lerxst@wam.umd.edu (where's my
thing) writes:
>
>  I was wondering if anyone out there could enlighten me on this car I saw
> the other day. It was a 2-door sports car, looked to be from the late 60s/
> early 70s. It was called a Bricklin. The doors were really small. In
addition,
> the front bumper was separate from the rest of the body. This is
> all I know. If anyone can tellme a model name, engine specs, years
> of production, where this car is made, history, or whatever info you
> have on this funky looking car, please e-mail.

Bricklins were manufactured in the 70s with engines from Ford. They are rather
odd looking with the encased front bumper. There aren't a lot of them around,
but Hemmings (Motor News) ususally has ten or so listed. Basically, they are a
performance Ford with new styling slapped on top.

>    ---- brought to you by your neighborhood Lerxst ----

Rush fan?

--
```

```
Robert Seymour              rseymour@reed.edu
Physics and Philosophy, Reed College    (NeXTmail accepted)
Artificial Life Project         Reed College
Reed Solar Energy Project (SolTrain)    Portland, OR
```

Share Edit Follow Flag

edited Oct 29, 2019 at 0:00
kgkmeekg
524 2 8 17

answered Aug 26, 2012 at 8:45
ogrisel
39k 12 115 124

A follow-up question: if i have a very big number of documents, the linear_kernel function in step 2 can be the performance bottleneck, since it's linear to the number of rows. Any thoughts on how to reduce it to sublinear? – Shuo Apr 9, 2014 at 23:20

You can use the "more like this" queries of Elastic Search and Solr that should yield approximate answers with a sub-linear scalability profile. – ogrisel Jun 9, 2015 at 8:02

9 Would this give you the cosine similarity of each document with every other document, instead of just the first one: `cosine_similarities = linear_kernel(tfidf, tfidf)` ? – ionox0 May 12, 2016 at 23:36

2 Yes, this will give you a square matrix of pairwise similarities. – ogrisel May 13, 2016 at 14:42

11 In case others were wondering like I did, in this case linear_kernel is equivalent to cosine_similarity because the TfidfVectorizer produces normalized vectors. See the note in the docs: scikit-learn.org/stable/modules/metrics.html#cosine-similarity – Chris Clark Jun 6, 2016 at 17:07

@ogrisel could you please take a look at stackoverflow.com/questions/39688927/python-tf-idf-predict-a-new-document-similarity/39689190#39689190 ? – Shlomi Schwartz Sep 25, 2016 at 17:05

Which one would be more efficient, `(tfidf*tfidf.T)` or `linear_kernel(tfidf, tfidf)` ? – user14675723 Dec 6, 2020 at 17:53