distance_metrics

July 2, 2018

0.1 New Distance Metrics for Probability Distribution and Bag of Words

A small tutorial to illustrate the new distance functions.

We would need this mostly when comparing how similar two probability distributions are, and in the case of gensim, usually for LSI or LDA topic distributions after we have a LDA model.

Gensim already has functionalities for this, in the sense of getting most similar documents - this, this and this are such examples of documentation and tutorials.

What this tutorial shows is a building block of these larger methods, which are a small suite of distance metrics. We'll start by setting up a small corpus and showing off the methods.

```
In [1]: from gensim.corpora import Dictionary
        from gensim.models import ldamodel
        from gensim.matutils import kullback_leibler, jaccard, hellinger, sparse2full
        import numpy

In [2]: # you can use any corpus, this is just illustratory

        texts = [['bank','river','shore','water'],
                ['river','water','flow','fast','tree'],
                ['bank','water','fall','flow'],
                ['bank','bank','water','rain','river'],
                ['river','water','mud','tree'],
                ['money','transaction','bank','finance'],
                ['bank','borrow','money'],
                ['bank','finance'],
                ['finance','money','sell','bank'],
                ['borrow','sell'],
                ['bank','loan','sell']]

        dictionary = Dictionary(texts)
        corpus = [dictionary.doc2bow(text) for text in texts]

In [3]: numpy.random.seed(1) # setting random seed to get the same results each time.
        model = ldamodel.LdaModel(corpus, id2word=dictionary, num_topics=2)

        model.show_topics()

Out[3]: [(0,
          u'0.164*bank + 0.142*water + 0.108*river + 0.076*flow + 0.067*borrow + 0.063*sell + (
         (1,
          u'0.196*bank + 0.120*finance + 0.100*money + 0.082*sell + 0.067*river + 0.065*water +
```

Let's take a few sample documents and get them ready to test Similarity. Let's call the 1st topic the water topic and the second topic the finance topic.

Note: these are all distance metrics. This means that a value between 0 and 1 is returned, where values closer to 0 indicate a smaller 'distance' and therefore a larger similarity.

```
In [4]: doc_water = ['river', 'water', 'shore']
        doc_finance = ['finance', 'money', 'sell']
        doc_bank = ['finance', 'bank', 'tree', 'water']

        # now let's make these into a bag of words format

        bow_water = model.id2word.doc2bow(doc_water)
        bow_finance = model.id2word.doc2bow(doc_finance)
        bow_bank = model.id2word.doc2bow(doc_bank)

        # we can now get the LDA topic distributions for these
        lda_bow_water = model[bow_water]
        lda_bow_finance = model[bow_finance]
        lda_bow_bank = model[bow_bank]
```

0.2 Hellinger and Kullback–Leibler

We're now ready to apply our distance metrics.

Let's start with the popular Hellinger distance. The Hellinger distance metric gives an output in the range [0,1] for two probability distributions, with values closer to 0 meaning they are more similar.

```
In [5]: hellinger(lda_bow_water, lda_bow_finance)
```

```
Out[5]: 0.51251199778753576
```

```
In [6]: hellinger(lda_bow_finance, lda_bow_bank)
```

```
Out[6]: 0.23407305272210427
```

Makes sense, right? In the first example, Document 1 and Document 2 are hardly similar, so we get a value of roughly 0.5.

In the second case, the documents are a lot more similar, semantically. Trained with the model, they give a much less distance value.

Let's run similar examples down with Kullback Leibler.

```
In [7]: kullback_leibler(lda_bow_water, lda_bow_bank)
```

```
Out[7]: 0.30823547
```

```
In [8]: kullback_leibler(lda_bow_finance, lda_bow_bank)
```

```
Out[8]: 0.19881117
```

NOTE!

KL is not a Distance Metric in the mathematical sense, and hence is not symmetrical. This means that `kullback_leibler(lda_bow_finance, lda_bow_bank)` is not equal to `kullback_leibler(lda_bow_bank, lda_bow_finance)`.

```
In [9]: # As you can see, the values are not equal. We'll get more into the details of this
        later on in the notebook.
        kullback_leibler(lda_bow_bank, lda_bow_finance)
```

```
Out[9]: 0.24780412
```

In our previous examples we saw that there were lower distance values between bank and finance than for bank and water, even if it wasn't by a huge margin. What does this mean?

The `bank` document is a combination of both water and finance related terms - but as bank in this context is likely to belong to the finance topic, the distance values are less between the finance and bank bows.

```
In [10]: # just to confirm our suspicion that the bank bow is more to do with finance:

         model.get_document_topics(bow_bank)
```

```
Out[10]: [(0, 0.44146764073708339), (1, 0.55853235926291656)]
```

It's evident that while it isn't too skewed, it it more towards the finance topic.

Distance metrics (also referred to as similarity metrics), as suggested in the examples above, are mainly for probability distributions, but the methods can accept a bunch of formats for input. You can do some further reading on Kullback Leibler and Hellinger to figure out what suits your needs.

0.3 Jaccard

Let us now look at the Jaccard Distance metric for similarity between bags of words (i.e, documents)

```
In [11]: jaccard(bow_water, bow_bank)
```

```
Out[11]: 0.8571428571428572
```

```
In [12]: jaccard(doc_water, doc_bank)
```

```
Out[12]: 0.8333333333333334
```

```
In [13]: jaccard(['word'], ['word'])
```

```
Out[13]: 0.0
```

The three examples above feature 2 different input methods.

In the first case, we present to jaccard document vectors already in bag of words format. The distance can be defined as 1 minus the size of the intersection upon the size of the union of the vectors.

We can see (on manual inspection as well), that the distance is likely to be high - and it is.

The last two examples illustrate the ability for jaccard to accept even lists (i.e, documents) as inputs. In the last case, because they are the same vectors, the value returned is 0 - this means the distance is 0 and they are very similar.

0.4 Distance Metrics for Topic Distributions

While there are already standard methods to identify similarity of documents, our distance metrics has one more interesting use-case: topic distributions.

Let's say we want to find out how similar our two topics are, water and finance.

```
In [14]: topic_water, topic_finance = model.show_topics()

         # some pre processing to get the topics in a format acceptable to our distance metrics

         def make_topics_bow(topic):
             # takes the string returned by model.show_topics()
             # split on strings to get topics and the probabilities
             topic = topic.split('+')
             # list to store topic bows
             topic_bow = []
             for word in topic:
                 # split probability and word
                 prob, word = word.split('*')
                 # get rid of spaces
                 word = word.replace(" ","")
                 # convert to word_type
                 word = model.id2word.doc2bow([word])[0][0]
                 topic_bow.append((word, float(prob)))
             return topic_bow

         finance_distribution = make_topics_bow(topic_finance[1])
         water_distribution = make_topics_bow(topic_water[1])

         # the finance topic in bag of words format looks like this:
         finance_distribution
```

```
Out[14]: [(3, 0.196),
          (12, 0.12),
          (10, 0.1),
          (14, 0.082),
          (2, 0.067),
          (0, 0.065),
          (11, 0.056),
          (15, 0.049),
          (5, 0.046),
          (9, 0.04)]
```

Now that we've got our topics in a format more acceptable by our functions, let's use a Distance metric to see how similar the word distributions in the topics are.

```
In [15]: hellinger(water_distribution, finance_distribution)
```

```
Out[15]: 0.36453028040240248
```

Our value of roughly 0.36 means that the topics are not TOO distant with respect to their word distributions. This makes sense again, because of overlapping words like `bank` and a small size dictionary.

0.5 Some things to take care of

In our previous example we didn't use Kullback Leibler to test for similarity for a reason - KL is not a Distance 'Metric' in the technical sense (you can see what a metric is here. The nature of it, mathematically also means we must be a little careful before using it, because since it involves the log function, a zero can mess things up. For example:

```
In [16]: # 16 here is the number of features the probability distribution draws from
         kullback_leibler(water_distribution, finance_distribution, 16)
```

```
Out[16]: inf
```

That wasn't very helpful, right? This just means that we have to be a bit careful about our inputs. Our old example didn't work out because they were some missing values for some words (because `show_topics()` only returned the top 10 topics).

This can be remedied, though.

```
In [17]: # return ALL the words in the dictionary for the topic-word distribution.
         topic_water, topic_finance = model.show_topics(num_words=len(model.id2word))

         # do our bag of words transformation again
         finance_distribution = make_topics_bow(topic_finance[1])
         water_distribution = make_topics_bow(topic_water[1])

         # and voila!
         kullback_leibler(water_distribution, finance_distribution)
```

```
Out[17]: 0.19781515
```

You may notice that the distance for this is quite less, indicating a high similarity. This may be a bit off because of the small size of the corpus, where all topics are likely to contain a decent overlap of word probabilities. You will likely get a better value for a bigger corpus.

So, just remember, if you intend to use KL as a metric to measure similarity or distance between two distributions, avoid zeros by returning the ENTIRE distribution. Since it's unlikely any probability distribution will ever have absolute zeros for any feature/word, returning all the values like we did will make you good to go.

0.6 So - what exactly are Distance Metrics?

Having seen the practical usages of these measures (i.e, to find similarity), let's learn a little about what exactly Distance Measures and Metrics are.

I mentioned in the previous section that KL was not a distance metric. There are 4 conditons for for a distance measure to be a matric:

1. d(x,y) >= 0
2. d(x,y) = 0 <=> x = y
3. d(x,y) = d(y,x)
4. d(x,z) <= d(x,y) + d(y,z)

That is: it must be non-negative; if x and y are the same, distance must be zero; it must be symmetric; and it must obey the triangle inequality law.

Simple enough, right? Let's test these out for our measures.

```
In [18]: # normal Hellinger
         hellinger(water_distribution, finance_distribution)
```

```
Out[18]: 0.22491784692602151
```

```
In [19]: # we swap finance and water distributions and get the same value. It is indeed
         symmetric!
         hellinger(finance_distribution, water_distribution)
```

```
Out[19]: 0.22491784692602151
```

```
In [20]: # if we pass the same values, it is zero.
         hellinger(water_distribution, water_distribution)
```

```
Out[20]: 0.0
```

```
In [21]: # for triangle inequality let's use LDA document distributions
         hellinger(lda_bow_finance, lda_bow_bank)
```

```
Out[21]: 0.23407305272210427
```

```
In [22]: # Triangle inequality works too!
         hellinger(lda_bow_finance, lda_bow_water) + hellinger(lda_bow_water, lda_bow_bank)
```

```
Out[22]: 0.79979376323008911
```

So Hellinger is indeed a metric. Let's check out KL.

```
In [23]: kullback_leibler(finance_distribution, water_distribution)
```

```
Out[23]: 0.2149342
```

```
In [24]: kullback_leibler(water_distribution, finance_distribution)
```

```
Out[24]: 0.19781515
```

We immediately notice that when we swap the values they aren't equal! One of the four conditions not fitting is enough for it to not be a metric.

However, just because it is not a metric, (strictly in the mathematical sense) does not mean that it is not useful to figure out the distance between two probability distributions. KL Divergence is widely used for this purpose, and is probably the most 'famous' distance measure in fields like Information Theory.

For a nice review of the mathematical differences between Hellinger and KL, this link does a very good job.

0.7 Conclusion

That brings us to the end of this small tutorial. The scope for adding new similarity metrics is large, as there exist an even larger suite of metrics and methods to add to the matutils.py file. (This is one paper which talks about some of them)

Looking forward to more PRs towards this functionality in Gensim! :)