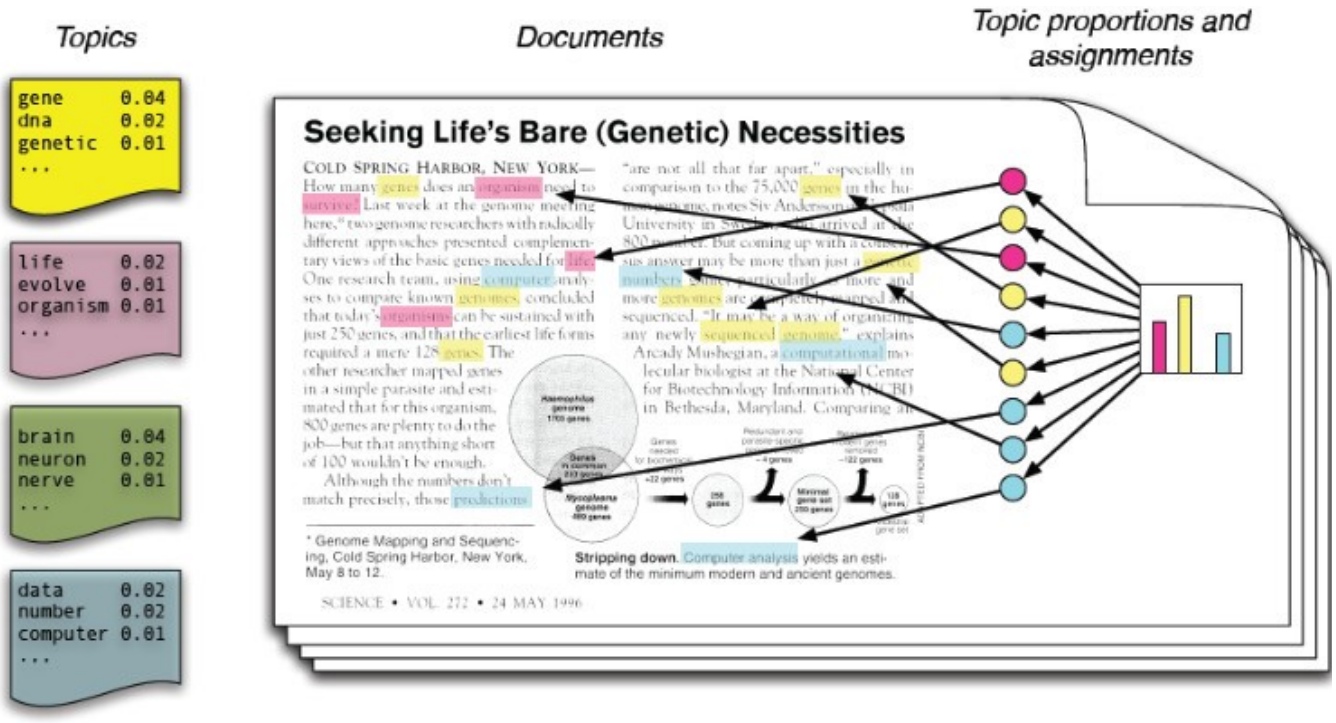

Image from Blei (2011), KDD Tutorial

A Non-Technical Introduction to LDA Topic Models with an Application in Hotel Reviews

Fatih Akici

As the rock band R.E.M. wisely put it quite some time ago, it's the end of the world as we know it. Our activities and interactions in today's new world are completely different than those we had in the old one. We engage in digital friendships, express our feelings by liking stuff with one click, and buy things online without even touching the product. In today's world **the largest taxi company owns no cabs, the largest accommodation company owns no rooms, the largest movie house owns no theaters, the most valuable retailer holds no inventory, and the largest communication providers have no infrastructure**. Not to mention that the world's

largest and most profitable **online travel agency** rents no hotel rooms, changing the **once-prevalent business model of the hospitality industry**.

The new world is built on data, and we generate enormous amounts of data every second. Digital text is a part of this big data that grows exponentially. We tweet, comment, review, blog, and write online articles, just as I am now. The big problem with this high volume of textual content is that we can't read all of it. Hence we need an algorithmic reduction approach to "summarize" this high volume of text data into insightful characteristics, such as the sentiment of a tweet, whether an email is spam or not, or what the online reviews of a product/service are talking about, that is, what **topics** the reviewers are referring to. In my article, I am going to focus on topic modeling, that is finding the topics that are hidden/latent in text documents. This is an important task, which can have immediate business implications. One survey finds that **88% of consumers say they trust online reviews as much as personal recommendations**, and yet another finding indicates that **90% of customers say their buying decisions are influenced by online reviews**. If our algorithm can detect successfully the topics of the customer reviews of a hotel/city/country, we can recommend that place to the new customers who are most relevant to those topics. After all, an email/ad campaign maximizes customer retention, satisfaction, engagement, and net conversion when it is based on a recommender engine that matches the target product/service with the right customer.

In topic modeling, more precisely Latent Dirichlet Allocation (LDA), which is the most popular form of it, our goal is to figure out what topics a text object is about. Now, this might sound like Don Quixote tilting at windmills, because when you come to think of it, what is a topic, and who decides which words are associated with it? Do we have an authority that classifies all of the words on earth under certain topics? How do we find topics, despite the fact that we don't have a word-to-topic labeling? I will explain topic modeling in words, with no math and probabilistic graphical model representation, and will present an application on real data using Python.

With the term "topic" in LDA, we don't mean one word, such as "sports" or "politics" or "science" etc., but rather a distribution/collection of a number of words. So, a topic looks more like "%45 Soccer, %35 Running, %20 Ping-pong", and we *infer* that this topic is "sports". However, there aren't strict labels that force us to call this topic "sports", one can call it "fun activities", if s/he wants. Conceptually, a topic is a distribution over a fixed vocabulary (huge list of words). Words come together according to a recipe (multinomial distribution), and generate topics.

At the same time, topics are latent generative processes that we believe to come together at certain rates, and generate a text document "behind the curtains" according to another recipe (Dirichlet distribution). For example, this article is

generated by a mixture of the topics "machine learning, business, probability, and programming". Further, the topic "machine learning" is generated by the words "algorithms, artificial intelligence, clustering, and classification"; the topic "business" is generated by the words "marketing, customer retention, profit" and so on.

In summary, a fixed vocabulary come together according to some recipe and generate topics, and then those topics come together according to some other recipe and generate the text documents. **Hence, our assumption is that any text is actually a bag of words that is generated by a distribution over distributions of a vocabulary, and our goal is to find the hidden underlying generative elements.**

Now let's see how LDA works on real data, using Python. I will share my Jupyter notebook when I hopefully publish the more technical version of this article on a major data science blog soon.

The data I chose for this article is hotel customer reviews data, which is publicly available on **this link**. The data consist of collections of text reviews and numerical ratings for, and prices of 12773 hotels in separate JSON files. A sample review looks like this:

When we arrived at this wonderful hotel, we were greeted with those magic words "we have upgraded you"! This was to room 414 and it was one of the best rooms I've ever stayed in. The whole hotel oozes class at a reasonable price and all the staff were friendly and efficient. I will always return here when in Seattle and suggest you do the same!

You guessed it right, this hotel aced it with a 5-star rating from this happy customer. Let's look at another review for the same hotel.

I booked this hotel based on TripAdvisor reviews. When I reserved the room, I mentioned it was our wedding anniversary. When we checked in, they had upgraded us to a townhouse for no extra charge! And gave us this amazing dessert to split with a card signed by the staff. I would highly recommend this hotel to everyone I know travelling to Seattle.

Another happy customer, another 5-star rating, and... Another upgrade! People seem to love upgrades. This is a wonderful observation that helps us start a data exploration. The histogram of ratings among the population of customers who used the term "upgrade" is this: {1:5505, 2:8067, 3:13996, 4:31998, 5:46868}, and visually **this**. The 46868+31998 people who gave the hotel a 4 or 5-star rating makes sense, but who are those 5505 people who used the word upgrade but rated the hotel only 1-star? Turns out they had comments like these:

"I ended up having to pay $40/night for a room UPGRADE", "management didn't quite care about upgrading anything", "was willing to pay for an upgrade to a water view room but never received a response from the hotel"

Speaking of ratings, I went ahead and looked at how hotel price summary statistics behave against customer ratings. See my matplotlib output **here**. Prices and ratings definitely correlate, and this is relevant because new customers will be willing to pay more for the higher rated hotels, and this will in turn increase the travel agency's commission. Now add the extra "topic" and "sentiment" information that can be extracted from reviews, you get a way better prediction of how much the room is worth, and how much commission the traveling agency can expect, as well as can increase profits by price differentiation. The reason to this added value of text analysis is that the new customers will not only look at the rating in their booking decision, but also read the comments and act based on the keywords (topic) and satisfaction/complaints (sentiment).

After some steps of cleaning, stop-word removal, tokenization, stemming, that are made easy by the amazing Python libraries gensim and nltk, we are left with 1,515,761 review-rating-price instances. I intentionally tied the reviews, ratings, and prices together so I can build machine learning models to understand their relationship. For purposes of this article, I built the LDA topic model on the column of reviews. I chose 10 topics in model development, after removing the words that appear too rarely or too often. Saving the intermediate steps for later to the technical blog post, I'll jump right into the results. The discovered 10 topics with their top 10 words are as follows

1- (0.016*"rate" + 0.010*"properti" + 0.010*"charg" + 0.009*"per" + 0.009*"pay" + 0.008*"fee" + 0.008*"star" + 0.007*"upgrad" + 0.007*"club" + 0.007*"internet"),

2- (0.012*"told" + 0.009*"said" + 0.008*"wait" + 0.007*"casino" + 0.007*"manag" + 0.007*"anoth" + 0.006*"left" + 0.005*"charg" + 0.005*"smoke" + 0.005*"bad"),

3- (0.021*"squar" + 0.017*"central" + 0.016*"block" + 0.016*"nyc" + 0.015*"york" + 0.014*"subway" + 0.009*"distanc" + 0.009*"within" + 0.009*"easi" + 0.008*"station"),

4- (0.024*"resort" + 0.021*"buffet" + 0.019*"drink" + 0.015*"beach" + 0.013*"show" + 0.011*"ate" + 0.009*"fun" + 0.009*"eat" + 0.008*"dinner" + 0.008*"play"),

5- (0.034*"airport" + 0.022*"bu" + 0.018*"shuttl" + 0.017*"taxi" + 0.013*"tour" + 0.012*"stop" + 0.011*"ticket" + 0.011*"cab" + 0.010*"flight" + 0.010*"ride"),

6- (0.037*"vega" + 0.034*"strip" + 0.012*"nois" + 0.011*"mirag" + 0.011*"la" + 0.011*"venetian" + 0.010*"bellagio" + 0.009*"mgm" + 0.007*"elev" + 0.007*"window"),

7- (0.184*"tower" + 0.067*"boston" + 0.053*"full" + 0.035*"wynn" + 0.025*"b" + 0.023*"wellington" + 0.021*"french" + 0.020*"showreview" + 0.020*"quarter" + 0.018*"st"),

8- (0.037*"kid" + 0.030*"famili" + 0.017*"bedroom" + 0.015*"children" + 0.012*"car" + 0.011*"kitchen" + 0.011*"adult" + 0.010*"daughter" + 0.010*"waldorf" + 0.010*"son"),

9- (0.020*"casino" + 0.011*"amaz" + 0.008*"perfect" + 0.008*"feel" + 0.008*"fantast" + 0.007*"thank" + 0.006*"husband" + 0.006*"friend" + 0.006*"ever" + 0.006*"home"),

10- (0.024*"coffe" + 0.019*"tv" + 0.009*"screen" + 0.009*"hot" + 0.008*"flat" + 0.008*"towel" + 0.008*"tabl" + 0.008*"tub" + 0.007*"light" + 0.007*"bath")

Now, what is this output that I call result? Well, remember the definition, a topic is a distribution over words. So what we see is the recipe of words that cluster together. More precisely, the "stemmed" version of words, that is roots of words (such as friend is the root of friends, friendly, friendship etc). Now, the interpretion of the discovered topics is crucial. The first topic is definitely about prices and charges. The second is about communication with the hotel, probably. The third is clearly the location of the hotel. Fourth is food and fun, fifth is transportation, sixth is about casinos and night life, seven is a little ambiguous but probably the view of the hotel room, eight is definitely family, nine is probably the feeling/impression of casinos, and ten is amenities. On a side note, what is that term "showreview" in topic 7? Who could've used that word? It doesn't look like the root of a word, so what is it? Digging a little, I find that it is the default text in the missing reviews which need to be removed. Uh-oh I need to re-build the model after fixing this. Making mistakes and starting all over again is a typical routine for a data scientist I guess. Anyway, let's see how a new review is going to be handled by the model. So I made up this review:

This place is not as big as MGM or Bellagio, but it is a perfect casino. They have great drinks, and the food is awesome too, so you can have a nice dinner there. I had fun at this amazing place.

It is clearly about casinos, night life, eating, drinking, and fun. Let's see with which topics it is associated the most, by feeding it into the trained LDA model. The resulting topic distribution is [(4, 0.35), (6, 0.22), (9, 0.35)]. That is, it is associated with the topics 4, 6, and 9 the most. Look above for those topics' meanings, this makes perfect sense! Let's make up another review:

We went to this hotel on my husband's vacation as a family. They have a shuttle service to pick you up from the airport, otherwise it would be difficult especially with the kids.

When we input this into the trained model, the topics it is associated with the most turn out to be [(4, 0.43), (7, 0.49)], hence it is highly related with the topics 4 and 7, namely the family and transportation topics - isn't that perfect?

So, what's next? Well, exploring a corpus of more than 1,5 million reviews in Python was not very efficient. So I'll work on the same exercise in (Py)Spark, in order to gain more experience in the hottest big data tool out there. I am also thinking about a recommender engine design on topics, keeping an eye on how one can A/B test it. One can ask a number of questions to this data set and answer using machine learning, as when it comes to creativity and turning insights into action to create business value, the sky is the limit.

In this article I tried to explain a highly mathematical model using plain language, emphasize its business value, and present an application of it using Python. If you like it, please share; and if you have any questions, please comment.