

Predicting what user reviews are about with LDA and gensim (<http://www.vladsandulescu.com/topic-prediction-lda-user-reviews/>)

I was rather impressed with the impressions and feedback I received for my [Opinion phrases](http://www.vladsandulescu.com/opinion-phrases/) (<http://www.vladsandulescu.com/opinion-phrases/>) prototype - code repository [here](https://github.com/vladsandulescu/phrases) (<https://github.com/vladsandulescu/phrases>). So yesterday, I have decided to rewrite my previous post on topic prediction for short reviews using Latent Dirichlet Analysis and its implementation in [gensim](http://radimrehurek.com/gensim/) (<http://radimrehurek.com/gensim/>).

I have previously worked with topic modeling for my [MSc thesis](http://www.vladsandulescu.com/opinion-spam-detection-through-semantic-similarity/) (<http://www.vladsandulescu.com/opinion-spam-detection-through-semantic-similarity/>) but there I used the [Semilar toolkit](http://www.semanticsimilarity.org/) (<http://www.semanticsimilarity.org/>) and a looot of C# code. Having read many articles about gensim, I was itchy to actually try it out.

Why would we be interested in extracting topics from reviews?

It is becoming increasingly difficult to handle the large number of opinions posted on review platforms and at the same time offer this information in a useful way to each user so he or she can make a decision fast whether to buy the product or not. Topic-based aggregations and short review summaries are used to group and condense what other users think about the product in order to personalize the content served to a new user and shorten the time he needs to make a buying decision.

A short example always works best. Suppose a review says: *The mailing pack that was sent to me was very thorough and well explained,correspondence from the shop was prompt and accurate,I opted for the cheque payment method which was swift in getting to me. All in all, a fast efficient service that I had the upmost confidence in,very professionally executed and I will suggest you to my friends when there mobiles are due for recycling :-)*

Some of the topics that could come out of this review could be *delivery, payment*

method and customer service.

In short, knowing what the review talks helps automatically categorize and aggregate on individual keywords and aspects mentioned in the review, assign aggregated ratings for each aspect and personalize the content served to a user. Or simply calculate the efficiency of each of the departments in a company by what people write in their reviews - in this example, the guys in the customer service department as well as the delivery guys would be pretty happy.

What do I need to run the code?

You can clone the [repository \(https://github.com/vladsandulescu/topics\)](https://github.com/vladsandulescu/topics) and play with the Yelp's dataset which contains many reviews or use your own short document dataset and extract the LDA topics from it.

Get the [Yelp academic dataset \(http://www.yelp.com/dataset_challenge\)](http://www.yelp.com/dataset_challenge) and import the reviews from the json file into your local MongoDB by running the **yelp/yelp-reviews.py** file. Use [MongoDB \(http://www.mongodb.org/\)](http://www.mongodb.org/), take my word for it, you'll never write to a text file ever again! You will also need [PyMongo \(http://api.mongodb.org/python/current/\)](http://api.mongodb.org/python/current/), [NLTK \(http://www.nltk.org/\)](http://www.nltk.org/), [NLTK data \(http://www.nltk.org/data.html\)](http://www.nltk.org/data.html) (in Python run `import nltk`, then `nltk.download()`). I personally have these Corpora modules installed: Brown Corpus, SentiWordNet, WordNet, as well as the following Models: Treebank Part of Speech Tagger (HMM), Treebank Part of Speech Tagger (Maximum Entropy), Punkt Tokenizer Models. Finally, don't forget to install [gensim \(http://radimrehurek.com/gensim/\)](http://radimrehurek.com/gensim/).

OK, enough foreplay, this is how the code works. Skip to the results if you are not interested in running the prototype.

How does the prototype work?

Well, the main goal of the prototype is to try to extract topics from a large reviews corpus and then predict the topic distribution for a new unseen review. Please read [this paper \(http://www.yelp.com/html/pdf/YelpDatasetChallengeWinner_ImprovingRestaurants.pdf\)](http://www.yelp.com/html/pdf/YelpDatasetChallengeWinner_ImprovingRestaurants.pdf) first, before checking out the source code, as I have followed it rather closely and tried to reproduce their results. These guys won a prize in the Yelp dataset challenge and in order for me to check if I get similar results, I also experimented on the [Yelp academic dataset \(http://www.yelp.com/dataset_challenge\)](http://www.yelp.com/dataset_challenge).

Future plans include trying out the prototype on [Trustpilot \(http://www.trustpilot.com\)](http://www.trustpilot.com) reviews, when we will open up the Consumer APIs to the world. I plan to do another blog post then, when I will explain how you can run the

prototype on top of the Trustpilot API (<https://github.com/trustpilot/developers>) and get nice results from it.

If you clone the repository, you will see a few python files which make up the execution pipeline: `yelp/yelp-reviews.py`, `reviews.py`, `corpus.py`, `train.py`, `display.py` and `predict.py`. I have not yet made a main class to run the entire prototype, as I expect people might want to tweak this pipeline in a number of ways. For example, some may prefer a corpus containing more than just nouns, or avoid writing to Mongo, or keep more than 10000 words, or use more/less than 50 topics and so on.

You should just run these following files in order.

- **yelp/yelp-reviews.py** - gets the reviews from the json file and imports them to MongoDB in a collection called *Reviews*
- **reviews.py/ reviews_parallel.py** - loops through all the reviews in the initial dataset and for each review it: splits the review into sentences, removes stopwords, extracts parts-of-speech tags for all the remaining tokens, stores each review, i.e. reviewId, business name, review text and (word,pos tag) pairs vector to a new MongoDB database called *Tags*, in a collection called *Reviews*. If you have many reviews, try running **reviews_parallel.py**, which uses the Python multiprocessing features to parallelize this task and use multiple processed to do the POS tagging.
- **corpus.py** - loops through all the reviews from the new MongoDB collection in the previous step, filters out all words which are not nouns, uses WordNetLemmatizer to lookup the lemma of each noun, stores each review together with nouns' lemmas to a new MongoDB collection called *Corpus*.
- **train.py** - feeds the reviews corpus created in the previous step to the gensim LDA model, keeping only the 10000 most frequent tokens and using 50 topics.
- **display.py** - loads the saved LDA model from the previous step and displays the extracted topics.
- **predict.py** - given a short text, it outputs the topics distribution. Simply lookout for the highest weights on a couple of topics and that will basically give the "basket(s)" where to place the text.
- **stopwords.txt** - [stopwords list](http://www.lextek.com/manuals/onix/stopwords2.html) (<http://www.lextek.com/manuals/onix/stopwords2.html>) created by Gerard

The resulting topics

POS tagging the entire review corpus and training the LDA model takes considerable time, so expect to leave your laptop running over night while you dream of phis and thetas. It took ~10h on my personal laptop (Lenovo T420s with Intel i5 inside and 8GB of RAM) to do POS tagging for all 1,125,458 Yelp reviews (used `reviews_parallel.py` for this). I ran the LDA model for 50 topics, but feel free to choose more.

Here were the resulting 50 topics, ignore the bold words written in parenthesis for now:

- 0: (food or sauces or sides)** 0.028sauce + 0.019meal + 0.018meat + 0.017salad + 0.016food + 0.015menu + 0.015side + 0.015flavor + 0.013dish + 0.012pork
- 1: (breakfast)** 0.122egg + 0.096breakfast + 0.065bacon + 0.064juice + 0.033sausage + 0.032fruit + 0.024morning + 0.023brown + 0.023strawberry + 0.022crepe
- 2: (restaurant owner)** 0.074owner + 0.073year + 0.048family + 0.032business + 0.029company + 0.028day + 0.026month + 0.025time + 0.024home + 0.021daughter
- 3: (terrace or surroundings)** 0.065park + 0.030air + 0.028management + 0.027dress + 0.027child + 0.026parent + 0.025training + 0.024fire + 0.020security + 0.020treatment
- 4: (seafood)** 0.091shrimp + 0.090crab + 0.077lobster + 0.060seafood + 0.054nail + 0.042salon + 0.039leg + 0.033coconut + 0.032oyster + 0.031scallop
- 5: (thai food)** 0.055soup + 0.054rice + 0.045roll + 0.036noodle + 0.032thai + 0.032spicy + 0.029bowl + 0.028chicken + 0.026dish + 0.023beef
- 6: (cafe)** 0.086sandwich + 0.063coffee + 0.048tea + 0.026place + 0.018cup + 0.016market + 0.015cafe + 0.015bread + 0.013lunch + 0.013order
- 7: (service)** 0.068food + 0.049order + 0.044time + 0.042minute + 0.038service + 0.034wait + 0.030table + 0.029server + 0.024drink + 0.024waitress
- 8: (dessert)** 0.078cream + 0.071ice + 0.059flavor + 0.056dessert + 0.049cake + 0.039chocolate + 0.021sweet + 0.015butter + 0.014taste + 0.013apple
- 9: (greek food)** 0.052topping + 0.039yogurt + 0.034patty + 0.033hubby + 0.026flavor + 0.026sample + 0.024gyro + 0.022sprinkle + 0.021coke + 0.020greek
- 10: (service)** 0.055time + 0.037job + 0.032work + 0.026hair + 0.025experience + 0.024class + 0.020staff + 0.020massage + 0.018day + 0.017week
- 11: (mexican food)** 0.131chip + 0.081chili + 0.071margarita + 0.056fast + 0.031dip + 0.030enchilada + 0.026quesadilla + 0.026gross + 0.024bell + 0.020pastor
- 12: (price)** 0.082money + 0.046% + 0.042tip + 0.040buck + 0.040ticket + 0.037price + 0.033pay + 0.029worth + 0.027cost + 0.024ride
- 13: (location or not sure)** 0.061window + 0.058soda + 0.056lady + 0.037register + 0.031ta + 0.030man + 0.028haha + 0.026slaw + 0.020secret + 0.018wet

14: (italian food) 0.144pizza + 0.038wing + 0.031place + 0.029sauce + 0.026cheese + 0.023salad + 0.021pasta + 0.019slice + 0.016brisket + 0.015order
15: (family place or drive-in) 0.157car + 0.150kid + 0.030drunk + 0.028oil + 0.026truck + 0.024fix + 0.021college + 0.016vehicle + 0.016guy + 0.013arm
16: (bar or sports bar) 0.196beer + 0.069game + 0.049bar + 0.047watch + 0.038tv + 0.034selection + 0.033sport + 0.017screen + 0.017craft + 0.014playing
17: (hotel or accommodation) 0.134room + 0.061hotel + 0.044stay + 0.036pool + 0.027view + 0.024nice + 0.020gym + 0.018bathroom + 0.016area + 0.015night
18: (restaurant or atmosphere) 0.073wine + 0.050restaurant + 0.032menu + 0.029food + 0.029glass + 0.025experience + 0.023service + 0.023dinner + 0.019nice + 0.019date
19: (not sure) 0.052son + 0.027trust + 0.025god + 0.024crap + 0.023pain + 0.023as + 0.021life + 0.020heart + 0.017finish + 0.017word
20: (location or not sure) 0.057mile + 0.052arizona + 0.041theater + 0.037desert + 0.034middle + 0.029island + 0.028relax + 0.028san + 0.026restroom + 0.022shape
21: (club or nightclub) 0.064club + 0.063night + 0.048girl + 0.037floor + 0.037party + 0.035group + 0.033people + 0.032drink + 0.027guy + 0.025crowd
22: (brunch or lunch) 0.171wife + 0.071station + 0.058madison + 0.051brunch + 0.038pricing + 0.025sun + 0.024frequent + 0.022pastrami + 0.021doughnut + 0.016gas
23: (casino) 0.212vega + 0.103la + 0.085strip + 0.047casino + 0.040trip + 0.018aria + 0.014bay + 0.013hotel + 0.013fountain + 0.011studio
24: (service) 0.200service + 0.092star + 0.090food + 0.066place + 0.051customer + 0.039excellent + 0.035! + 0.030time + 0.021price + 0.020experience
25: (pub or fast-food) 0.254dog + 0.091hot + 0.026pub + 0.023community + 0.022cashier + 0.021way + 0.021eats + 0.020york + 0.019direction + 0.019root
26: (not sure) 0.087box + 0.040adult + 0.028dozen + 0.027student + 0.026sign + 0.025gourmet + 0.018decoration + 0.018shopping + 0.017alot + 0.016eastern
27: (bar) 0.120bar + 0.085drink + 0.050happy + 0.045hour + 0.043sushi + 0.037place + 0.035bartender + 0.023night + 0.019cocktail + 0.015menu
28: (italian food) 0.029chef + 0.027tasting + 0.024grand + 0.022caesar + 0.021amazing + 0.020linq + 0.020italian + 0.018superb + 0.016garden + 0.015al
29: (not sure) 0.064bag + 0.061attention + 0.040detail + 0.031men + 0.027school + 0.024wonderful + 0.023korean + 0.023found + 0.022mark + 0.022def
30: (mexican food) 0.122taco + 0.063bean + 0.043salsa + 0.043mexican + 0.034food + 0.032burrito + 0.029chip + 0.027rice + 0.026tortilla + 0.021corn
31: 0.096waffle + 0.057honey + 0.034cheddar + 0.032biscuit + 0.030haze + 0.025chicken + 0.024cozy + 0.022let + 0.022bring + 0.021kink
32: 0.033lot + 0.027water + 0.027area + 0.027) + 0.025door + 0.023(+ 0.021space + 0.021parking + 0.017people + 0.013thing
33: 0.216line + 0.054donut + 0.041coupon + 0.030wait + 0.029cute + 0.027cooky + 0.024candy + 0.022bottom + 0.019smoothie + 0.018clothes
34: 0.090phoenix + 0.077city + 0.042downtown + 0.037gem + 0.026seating + 0.025tourist + 0.022convenient + 0.021joke + 0.020pound + 0.017tom

35: 0.072*lol* + 0.056*mall* + 0.041*dont* + 0.035*omg* + 0.034*country* + 0.030*im* + 0.029*didnt* + 0.028*strip* + 0.026*real* + 0.025*choose*
36: 0.159*place* + 0.036*time* + 0.026*cool* + 0.025*people* + 0.025*nice* + 0.021*thing* + 0.021*music* + 0.020*friend* + 0.019'*m* + 0.018*super*
37: 0.138*steak* + 0.068*rib* + 0.063*mac* + 0.039*medium* + 0.026*bf* + 0.026*side* + 0.025*rare* + 0.021*filet* + 0.020*cheese* + 0.017*martini*
38: 0.075*patio* + 0.064*machine* + 0.055*outdoor* + 0.039*summer* + 0.038*smell* + 0.032*court* + 0.032*california* + 0.027*shake* + 0.026*weather* + 0.023*pretzel*
39: 0.124*card* + 0.080*book* + 0.079*section* + 0.049*credit* + 0.042*gift* + 0.040*dj* + 0.022*pleasure* + 0.019*charge* + 0.018*fee* + 0.017*send*
40: 0.081*store* + 0.073*location* + 0.049*shop* + 0.039*price* + 0.031*item* + 0.025*selection* + 0.023*product* + 0.023*employee* + 0.023*buy* + 0.020*staff*
41: 0.048*az* + 0.048*dirty* + 0.034*forever* + 0.033*pro* + 0.032*con* + 0.031*health* + 0.027*state* + 0.021*heck* + 0.021*skill* + 0.019*concern*
42: 0.037*time* + 0.028*customer* + 0.025*call* + 0.023*manager* + 0.023*day* + 0.020*service* + 0.018*minute* + 0.017*phone* + 0.017*guy* + 0.016*problem*
43: 0.197*burger* + 0.166*fry* + 0.038*onion* + 0.030*bun* + 0.022*pink* + 0.021*bacon* + 0.021*cheese* + 0.019*order* + 0.018*ring* + 0.015*pickle*
44: 0.069*picture* + 0.052*movie* + 0.052*foot* + 0.034*vip* + 0.031*art* + 0.030*step* + 0.024*resort* + 0.022*fashion* + 0.021*repair* + 0.020*square*
45: 0.054*sum* + 0.043*dim* + 0.042*spring* + 0.034*diner* + 0.032*occasion* + 0.029*starbucks* + 0.025*bonus* + 0.024*heat* + 0.022*yesterday* + 0.021*lola*
46: 0.071*shot* + 0.041*slider* + 0.038*met* + 0.038*tuesday* + 0.032*doubt* + 0.023*monday* + 0.022*stone* + 0.022*update* + 0.017*oz* + 0.017*run*
47: 0.152*show* + 0.050*event* + 0.046*dance* + 0.035*seat* + 0.031*band* + 0.029*stage* + 0.019*fun* + 0.018*time* + 0.015*scene* + 0.014*entertainment*
48: 0.099*yelp* + 0.094*review* + 0.031*ball* + 0.029*star* + 0.028*sister* + 0.022*yelpers* + 0.017*serf* + 0.016*dream* + 0.015*challenge* + 0.014'*m*
49: 0.137*food* + 0.071*place* + 0.038*price* + 0.033*lunch* + 0.027*service* + 0.026*buffet* + 0.024*time* + 0.021*quality* + 0.021*restaurant* + 0.019*eat*

All right, they look pretty cohesive, which is a good sign. Now comes the manual topic naming step where we can assign one representative keyword to each topic. This is useful when predicting the topics of new unseen reviews. I have suggested some keywords based on my instant inspiration, which you can see in the round parenthesis. I got bored after half of them, but I feel I made the point. You only need to set these keywords once and *summarize* each topic. I suggested some keywords while watching over the Kung Pao Chicken and having a beer...so my keywords may not match yours! Anyway, you get the idea.

It's up to you how you choose the keywords: you can be broader or more precise about what you are interested in the topic, select the most frequent word in the topic and setting that as the keywords, etc..

Predicting the topics of new unseen reviews

OK, now that we have the topics, let's see how the model predicts the topics distribution for a new review:

It's like eating with a big Italian family. Great, authentic Italian food, good advice when asked, and terrific service. With a party of 9, last minute on a Saturday night, we were sat within 15 minutes. The owner chatted with our kids, and made us feel at home. They have meat-filled raviolis, which I can never find. The Fettuccine Alfredo was delicious. We had just about every dessert on the menu. The tiramisu had only a hint of coffee, the cannoli was not overly sweet and they had this custard with wine that was so strangely good. It was an overall great experience!

The output of the **predict.py** file given this review is: [(0, 0.063979336376367435), (2, 0.19344804518265865), (6, 0.049013217061090186), (7, 0.31535985308065378), (8, 0.074829314265223476), (14, 0.046977300077683241), (15, 0.044438343698184689), (18, 0.09128157138884592), (28, 0.085020844956249786)]

Thus, the review is characterized mostly by topics 7 (32%) and 2 (19%). The missing topics, such as 1, 3, 4, 5 and so on are all zero, that's why they are missing. Well, what do you know, those topics are about the **service** and **restaurant owner**.

Another one:

Either the quality has gone down or my taste buds have higher expectations than the last time I was here (about 2 years ago). Now that SF has so many delicious Italian choices where the pasta is made in-house/homemade, it was tough for me to eat the store-bought pasta. The pasta lacked texture and flavor, and even the best sauce couldn't change my disappointment. The gnocchi tasted better, but I just couldn't get over how cheap the pasta tasted. I discovered another spot in North Beach called X and I was really impressed with their pasta, so that's my new go-to spot.

Distribution: [(2, 0.049949761363727557), (14, 0.67415587326751736), (28, 0.14795291772795682), (33, 0.044461283686581303), (44, 0.044349729171608801)]

Clearly, the review is about topic 14, which is **italian food**.

Third time's the charm:

Really superior service in general; their reputation precedes them and they deliver. It can indeed be tough to get seating, but I find them willingly accommodating when they can be, and seating at the bar can be really enjoyable, actually. So many wonderful items to choose from, but don't forget to save room for the over-the-top chocolate souffle; elegant

and wondrous. Oh and hello, roast Maine lobster, mini quail and risotto with dungeness crab. De-lish.

```
[(0, 0.12795812236631765), (4, 0.25125769311344842), (8, 0.097887323141830185),  
(17, 0.15090844416208612), (24, 0.12415345702622631), (27,  
0.067834960190092219), (35, 0.06375000000000007), (41, 0.06375000000000007)]
```

The topics predicted are topic 4 - **seafood** and topic 24 - **service**. Right on the money again. I just picked the first couple of topics but these can be selected based on their distribution, i.e. taking all above a set threshold.

It isn't generally this sunny in Denmark though... Take a closer look at the topics and you'll notice some are hard to summarize and some are overlapping. This is where a bit of LDA tweaking can improve the results.


While this method is very simple and very effective, it still needs some polishing, but that is beyond the goal of the prototype. LDA is however one of the main techniques used in the industry to categorize text and for the most simple review tagging, it may very well be sufficient.



Vlad Sandulescu

 [Twitter \(http://twitter.com/vladsandulescu\)](http://twitter.com/vladsandulescu)

 [LinkedIn \(http://linkedin.com/in/vladsandulescu\)](http://linkedin.com/in/vladsandulescu)

 [Github \(http://github.com/vladsandulescu\)](http://github.com/vladsandulescu)

Predicting what user reviews are about with LDA and gensim was published on September 09, 2014 and last modified on September 09, 2014 by [Vlad Sandulescu \(http://www.vladsandulescu.com/about/\)](http://www.vladsandulescu.com/about/).

YOU MIGHT ALSO ENJOY

[\(VIEW ALL POSTS \(HTTP://WWW.VLADSANDULESCU.COM/POSTS/\)\)](http://www.vladsandulescu.com/posts/)

- [Opinion spam detection - Literature review \(http://www.vladsandulescu.com/opinion-spam-detection-literature-review/\)](http://www.vladsandulescu.com/opinion-spam-detection-literature-review/)
 - [Opinion phrases \(http://www.vladsandulescu.com/opinion-phrases/\)](http://www.vladsandulescu.com/opinion-phrases/)
 - [Opinion spam detection through semantic similarity \(http://www.vladsandulescu.com/opinion-spam-detection-through-semantic-similarity/\)](http://www.vladsandulescu.com/opinion-spam-detection-through-semantic-similarity/)
-

