Userless Classification of the Yelp's Restaurants

Nicolas Drizard & Virgile Audi

Final Project for CS281, Harvard University
nicolasdrizard@g.harvard.edu vaudi@g.harvard.edu

Abstract

Our objective is to improve the categorization system of the Yelp businesses using machine learning techniques. To achieve our goal, we extract significant information out of the text reviews posted on Yelp using our own coded-up Latent Dirichlet Allocation (LDA) using online variational inference (OVI), and benchmark it with the LDA using a Gibbs sampling method (GS) and the Non Negative Matrix Factorization (NMF). We then present two methods for classifying restaurants. We first resorted to a supervised approach with one-vs-all classifiers on the text reviews combined with characteristic features of businesses - such as localisations, prices, customer type or check-in counts. The second approach is unsupervised, using a community detection algorithm on the network of restaurants, the Walktrap algorithm. If the features extraction methods perform differently for the supervised and unsupervised approaches, we observe some very promising results. Theses insights could be used as the first step to an automated categorization system on the Yelp website.

1. Introduction

The classification system in Yelp is manual. When the owner of a business creates a page for his restaurant or bar, he can provide a certain number of labels. Yelp users can then alter or even add new tags based on their experiences. As part of the Yelp Data Set Challenge Round 6, we decided to tackle the issue of entries classification. Our ultimate objective is to create an automatised method to label the businesses, based on their properties and most importantly on their reviews from Yelpers. The new tags could be added to the existing tags to build a finer classification.

This project can be divided in three parts. First, to extract the features, we applied various dimensionality reduction methods such as Latent Dirichlet Allocation (LDA) or Non-negative Matrix Factorization (NMF) to embed the reviews into a geometrical space. Then, we applied supervised learning methods to evaluate how much information the features carry. Last, we built an unsupervised clustering method using a community detection algorithm.

2. Data

Yelp provides data on all kinds of venues such as dentists or bars in 5 cities in the United-States and Europe. The data can be summarised as follows:

- 1.6 million reviews;
- 61 000 businesses;
- 481 000 attributes;
- 168 customer check-in counts, corresponding to the average number of checkins per hour per day of the week.

For simplicity, we chose to focus on restaurants and bars in the city of Las Vegas, Nevada. This reduced the data to a dataset of 3 822 businesses, with about 20 thousands reviews and 250 other features per entry such as hours, parking availability, take-out, ambience. If the numerical data provided by Yelp is mostly cleaned, it was not the case for the text data. Cleaning the reviews represented a significant part of this project. Here are some of the steps we followed in the process:

- We aggregated all the reviews about a particular business into a "super" review, in order to get a general sense of what the restaurant was about (this was possible because we don't need to differentiate the users writing the reviews);
- We applied spelling corrections, removed common stopwords and generic words such as "table" or "restaurants";
- We attempted at only selecting common words, removing the adjectives, in order to focus on what the venues were about and not the users' opinions (this process was done taking advantage of Spark to

reduce the computing time);

- We transformed the corpus of the "super reviews" into a document-term matrix that we could exploit;
- As we aggregated the reviews, some words may be more prevalent than others, so we scaled the words counts into the range [0, 100] for each review. (this step produced a finer words distribution for each topic of the LDA)

3. Feature Extraction

3.1. Approach

To extract content from the reviews, we applied two methods from topic modeling to reduce the document term matrix into meaningful information. We first looked at LDA, using both a Gibbs sampler (GS) and an Online Variational Inference (OVI) algorithm. Note that we used the Gibbs sampler as a way to benchmark our own version of OVI. The motivation behind OVI is twofold. First, it appeared more appropriate to the data we were working with. If some restaurants have already enough reviews to be classified accurately, most of them could have finer labels with more reviews. Moreover, some nuances may appear over the time among the clusters of restaurants. As a result, we should be able to add the new reviews to the corpus in a continuous flow. Second, it showed some extremely significant time improvements over the Gibbs sampler version in Python. Nonetheless, a major drawback of this method is the high sensitivity to the hyperparameters. We also wanted to compare the performance of LDA with a non-probabilistic approach which is the NMF.

3.2. Latent Dirichlet Allocation

The LDA is a three-level hierarchical Bayesian model. The basic idea is that documents are represented as random mixtures over latent topics and each topic is characterized by a distribution over words.[1]

The generative process for a document **w** in a corpus D is the following:

1. Choose the topics representation:
   $\phi \sim \text{Dir}(\beta)$
2. Choose the number of words:
   $N \sim \text{Poisson}(\xi)$
3. Choose the distribution of topics:
   $\theta_w \sim \text{Dir}(\alpha_w)$
4. For each of the N words:
   (a) Choose a topic assignement:
   $z_{n,w} \sim \text{Multinomial}(\theta_w)$
   (b) Choose a word:
   $w_n \sim \text{Multinomial}(\phi_{z_{n,w}})$

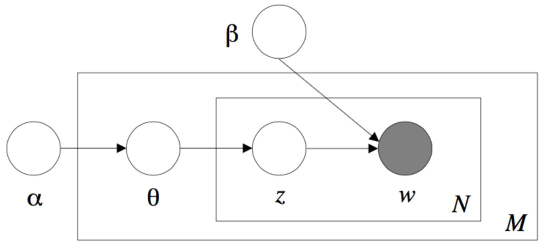


Figure 1: *Graphical Model Representation of LDA*

3.3. Inference

The main issue relies in computing the posterior distribution of the hidden variables given a document:

$$p(\theta, \mathbf{z} | \mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w} | \alpha, \beta)}{p(\mathbf{w} | \alpha, \beta)}$$

This distribution is intractable. As mentioned above, we used two methods to infer it: with a Gibbs sampler and with variational inference. In the first case, we are estimating the hyperparameters θ and ϕ with samples on the different variables. Alternatively, we chose the online variational inference method which finds the variational parameters that optimize a lower bound on the loglikelihood. The setup [2] of the variational inference is as follows.

We first approximate the true posterior by a simpler and factorised distribution:

$$q(\boldsymbol{z}, \boldsymbol{\theta}, \boldsymbol{\beta}) = q(\boldsymbol{z}) q(\boldsymbol{\theta}) q(\boldsymbol{\beta})$$

where:

$$q(z_{di} = k) = \phi_{d_{w_{di}}k} \quad q(\theta_d) = \text{Dir}(\theta_d, \gamma_d) \quad q(\beta_k) = \text{Dir}(\beta_k, \lambda_k)$$

The γ parameter rules the topic assignments for each document and the $\boldsymbol{\lambda}$ the topics themselves. We then minimise the KL divergence between the distribution q and the true posterior p. What differs in the online variational inference is that we sample a mini-batch of document at each step and perform the E-step as if this mini-batch constituted the entire corpus. We present the algorithm below:

Algorithm 1 Minibatch Online Variational Inference

```
1:  Define $\rho_t \equiv (\tau_0 + t)^{-\kappa}$
2:  Initalise $\boldsymbol{\lambda}$ randomly
3:  Sample a batch $\mathcal{S}$ of documents of size S
4:  for $t$=0 to $|D|/|\mathcal{S}|$ do
5:      for $d$ in $\mathcal{S}$ do
6:          while $\gamma_(d)$ hasn't converged do
7:              Set $\phi_{twk}^{(d)} \propto \exp\{\mathbb{E}_q \log \theta_{tk} + \mathbb{E}_q \log \beta_{kw}\}$
8:              Set $\gamma_{tk}^{(d)} = \alpha + \sum_w \phi_{twk}^{(d)} n_{tw}$
9:          Compute $\tilde{\lambda}_{kw} = \eta + \frac{D}{S} n_{tw} \phi_{twk}^{(d)}$
10:     $\boldsymbol{\lambda} = (1 - \rho_t)\boldsymbol{\lambda} + \rho_t \tilde{\boldsymbol{\lambda}}$
```

where $\kappa \in (0.5, 1]$ rules how fast we forget the old values of $\tilde{\lambda}$ and τ_0 how much weight one wants to put on the first iterations. α and η were fixed at the same value $\frac{1}{|D|}$.

3.4. Evaluation Method

We need a measure to evaluate the performance of our model and to tune the hyperparameters. We use the perplexity on held-out data as a measure of our model fit. The perplexity is defined as the geometric mean of the log likelihood of the words in the held-out set of documents given the trained model. In our case, for each document we held out 20% of the words which constitute the validation set to apply a 5-fold cross-validation.

$$perplexity(D_{test}) = \frac{\sum\limits_{d \in D_{test}} \log p(words)}{\sum\limits_{d \in D_{test}} |d|}$$

$$perplexity(D_{test}) = \frac{\sum\limits_{d \in D_{test}} \sum\limits_{w \in d} \log \left(\sum_{t \in topics} p(w|t) p(t|d) \right)}{\sum\limits_{d \in D_{test}} |d|}$$

We used this measure to optimize the number of topics K and the hyperparameters κ, τ of the optimization. We also used the perplexity on the training set to benchmark our algorithm with the Gibbs sampler in the LDA 1.0.3 package in Python:

$$\text{perp}_{OVI} = -6.78 \quad \text{and} \quad \text{perp}_{GS} = -6.45$$

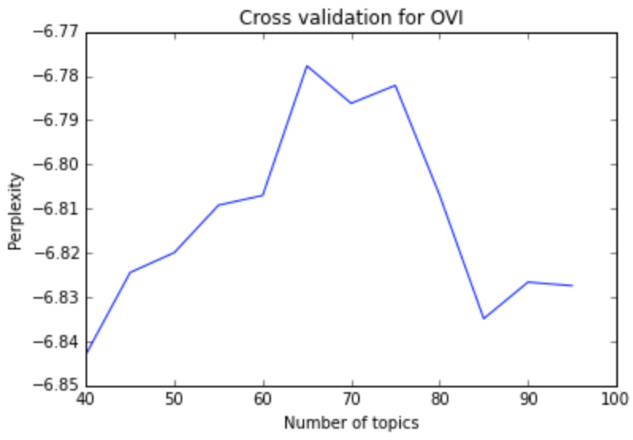
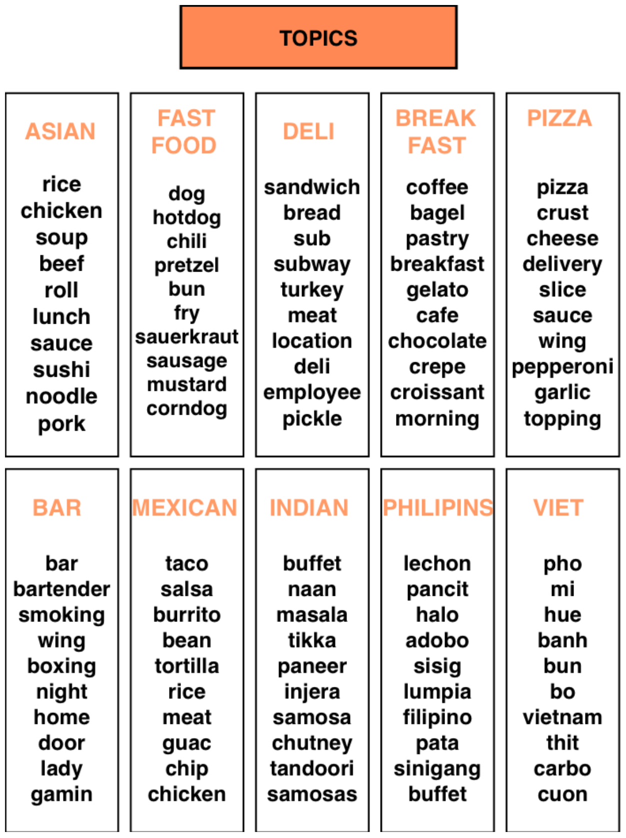


Figure 2: *5-fold Cross Validation over the number of Topics*

3.5. Results

Another way to assess the performance of our LDA was to inspect the topics outputted by the algorithm. We present some of the most significant topics:



TOPICS

ASIAN	FAST FOOD	DELI	BREAK FAST	PIZZA
rice	dog	sandwich	coffee	pizza
chicken	hotdog	bread	bagel	crust
soup	chili	sub	pastry	cheese
beef	pretzel	subway	breakfast	delivery
roll	bun	turkey	gelato	slice
lunch	fry	meat	cafe	sauce
sauce	sauerkraut	location	chocolate	wing
sushi	sausage	deli	crepe	pepperoni
noodle	mustard	employee	croissant	garlic
pork	corndog	pickle	morning	topping

BAR	MEXICAN	INDIAN	PHILIPINS	VIET
bar	taco	buffet	lechon	pho
bartender	salsa	naan	pancit	mi
smoking	burrito	masala	halo	hue
wing	bean	tikka	adobo	banh
boxing	tortilla	paneer	sisig	bun
night	rice	injera	lumpia	bo
home	meat	samosa	filipino	vietnam
door	guac	chutney	pata	thit
lady	chip	tandoori	sinigang	carbo
gamin	chicken	samosas	buffet	cuon

Figure 3: *Topics extracted from the Las Vegas reviews using OVI*

We obtained similar results using the NMF approach. As mentioned in the introduction, the topics themselves were not our primary interested, we were using these methods for dimensional reduction purposes.

We would like to make a general comment about the online variational algorithm. We noticed that its algorithm is extremely sensitive to the hyperparametrisation, i.e the batch size S, the forgetfullness parameter κ and the early iterations weight parameter τ. This was also noted by Tamara Broderick, Nicholas Boyd, Andre Wibisono and Ashia C. Wilson in their paper[3]. Validating the number of topics and these parameters was indeed crucial to obtain valuable results.

We now present two different approaches to answer the question of classification using supervised and unsupervised machine learning methods.

4. Classification

Once the features have been extracted, we can apply supervised learning methods to classify the Yelp restaurants considering the category labels as output. Several approaches are possible to solve this multi-label classiffication task depending on the nature and patterns of the features.

4.1. Approach

The features are divided into two main parts: on one hand, the topics assignements extracted from the reviews with the topic modeling algorithm (LDA or NMF) and on the other hand, the properties provided by Yelp. Preprocessing the latter part provided a 198 dimensional vector. These features contained both numerical values (rating, space coordinates, number of reviews, customer check-in...) and categorical ones which we converted into indicator variables.

Given our numerical features, many discriminative and generative models could be applied on our data. Nonetheless, the indicator features are sparse and the data are really noisy. We therefore needed to consider a penalized and/or aggregated model to reduce the overfitting.

We chose a one-vs-all approach to tackle this multi-label task and built one binary classifier for each label. We then assigned to the entries the labels predicted by each model. The goal is here to explore how the different models perform and to tune them to come up with the best estimator. We developped our own L2 penalized logistic regression and used *scikit-learn* [4] for the other models.

4.2. Results

In our experiments, we focused on the 10 most represented labels in the Las Vegas restaurants, which constitute our set of outputs. A same restaurant may have more than one label as we consider a multi-label classification.

Dimensionality Reduction We decided to reduce the dimension of the second part of the dataset (the properties provided by Yelp) before fitting our model. As the high-dimensionality is largely due to the customer check-in counts and as they share a common structure, we decided to project them in an euclidean space. Combining these embedded features with those provided by the reviews produced then a more consistent dataset. This dimensionality reduction was done with a principal component analysis (PCA), keeping **95%** of the variance which reduces the dimension from **168** to **19**.

Models Evaluation Here we provide the different results we obtained with our models. We ran one-vs-one classifiers for the 10 labels. Then we evaluated for each classifier its accuracy and averaged them to get the evaluation metric printed in the table. We also plotted the ROC curves for a specific model.
We tested them on three different kinds of features provided by the reviews depending on the method used to build them (*see the columns*). Then, for each model, we used both the raw features provided by Yelp and those embedded by the PCA projection (*see the rows*). We also show for comparison the baseline model where we do not assign any label. This result is really high because of the skewness of the considered sample: among the 10 categories considered for 3 822 restaurants, the most represented (Fast Food) contains 553 entries and the least (American New) 280 entries.

	OVI LDA	GS LDA	NMF
Baseline: all False	90,02 %	90.02 %	90.02 %
Kernel rbf SVM	91.00%	90.73%	90.67%
Kernel rbf SVM + PCA	90.62%	90.85%	90.85%
Logistic Regression	91.39%	91.86%	91.86%
Logistic Regression + PCA	92.37%	92.89%	93.16%
Lasso	91.18%	93.75%	92.40%
Lasso + PCA	92.53%	**94.83**%	93.35%
Random Forest	92.12%	93.58%	93.37%
Random Forest + PCA	92.84%	94.66 %	94.06%

Figure 4: *Classification results for different features*

The first outcome of this comparative study is the differences between each feature extraction method. The model built with the GS features seems to outperform on average the other two, which perform similarly to each other. These results are not surprising as the variationnal inference learns the closest model to the underlying one in terms of KL divergence. With the Gibbs sampling approach, we approximate the true underlying distribution at each iteration to refine our estimator. As a result, the latter method is by nature better to find the best model as long as it has enough iterations to converge and there actually exists an underlying mixture models. Nevertheless, the OVI remains a good proxy with regard to its time execution (53s against 50min for the Gibbs Sampler).

If we then compare the models to each other, we see that the support vector machine performs poorly and that the linear version was even closer to the baseline. This confirms that the data are not linearly separable and even their projection by the rbf function used in sklearn library (gamma function) is quite impossible to separate. The logistic regression performs better under the l1 penalization because of the sparsity of our features. Lastly, the random forest managed to reduce the common overfitting of the decision trees while aggregating them and provides on the PCA projection a very good estimator.

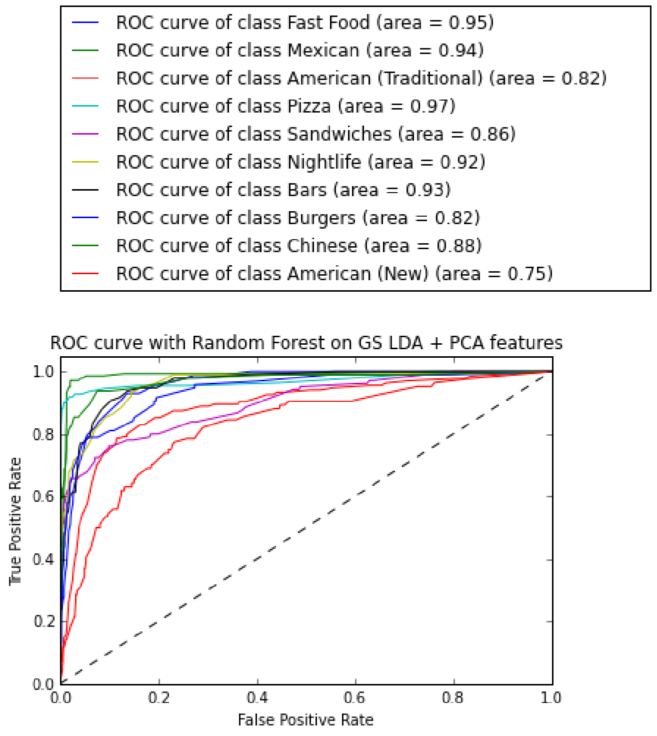


Figure 5: *Roc curve for the Random Forest one-vs-all classifiers*

We plotted here the ROC curves of the Random forest for the 10 classes considered. The ranking with respect to the area under the curve is very close to the one induced by the number of positive elements for each class. This shows that the more positive elements we have, the better the model actually performs. Nonetheless, learning the model on a smaller and more balanced train set with regards to the predicted class, i.e. with a lower number of negative elements, did not improve the classification accuracy.

Feature Importance Once we found a sufficiantly good estimator, we investigated the weights it assigned to each feature. This provided an answer to two important questions: *Are the feature extracted from the reviews more relevant than the one provided by Yelp to classify the restaurants by category? Among the features extracted by the reviews, can we identify for each category one or several highly discriminative features?* Moreover, we can compare the performance of the two different LDA we used.

We answer these questions with a comparative study on four specific categories (Fast Food, Pizza, Nightlife and Bars) with a Random Forest on four kinds of features, depending on the type of LDA used and the reduction with PCA. The results with the features built with an NMF are similar to those with the OVI. We used the measure provided by scikit learn to evaluate the importance of the features. It is defined as the total decrease in node impurity averaged over all the trees of the ensemble.

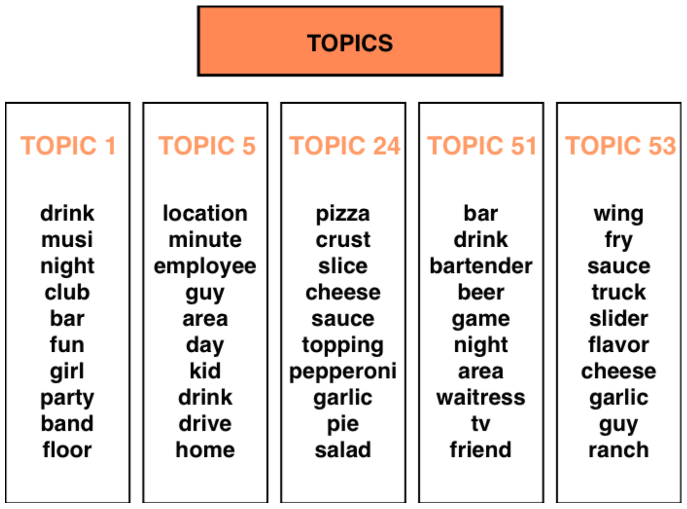


Figure 6: *Topics extracted from the Las Vegas reviews using GS*

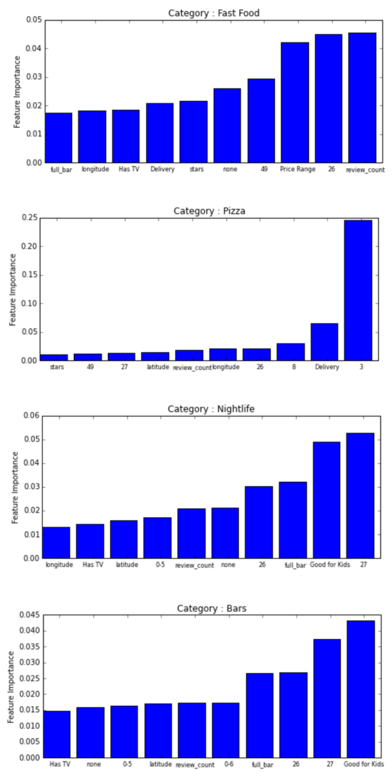


Figure 7: *Importance of the OVI LDA features in the one-vs-all Random Forest classification*

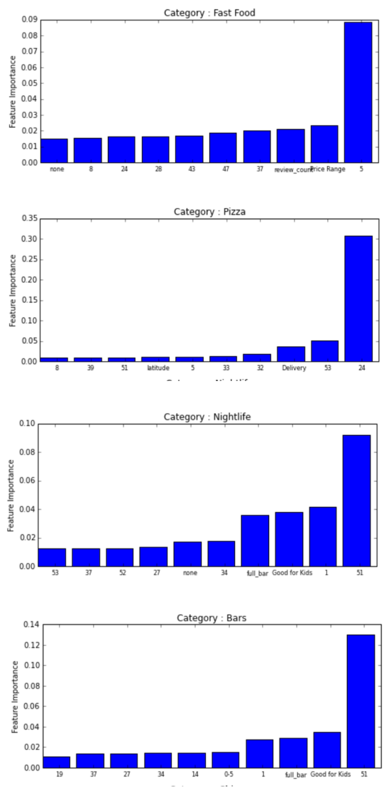


Figure 9: *Importance of the GS LDA features in the one-vs-all Random Forest classification*

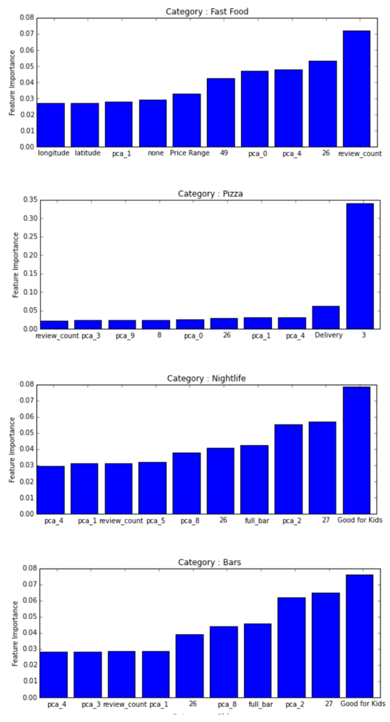


Figure 8: *Importance of the OVI LDA features with PCA in the one-vs-all Random Forest classification*

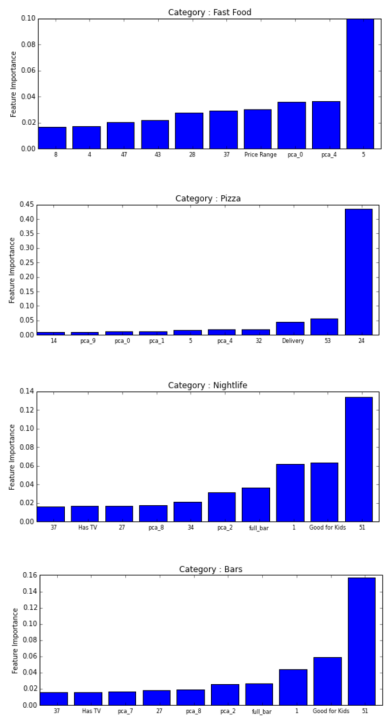


Figure 10: *Importance of the GS LDA features with PCA in the one-vs-all Random Forest classification*

First, considering the results for the OVI features, we observe that some properties provided by Yelp are discriminative for our classification. For example, the attribute *Good for Kids* helps to sort the Bars and the Nightlife categories. The Fast Food may be the restaurants with the largest number of reviews as they have the largest number of customers, the model confirms this with the high weight on the *reviews_count*. This is also a cheap restaurant, hence the importance of the price range. As a result, it is worth considering the features provided by Yelp as they help discriminate the entries.

Second, for the Pizza category with the features extracted by the OVI we observe a discriminative feature which matchs the one about Pizza provided in the figure 3. Then, for the same LDA method, there is no discriminative feature for the other categories. Now, if we consider the results for the GS features, each category has a highly discriminative feature. Moreover, if we look at the most common words for each of these topics in the figure 6, we naturally understand their importance in the classification. This result confirms our analysis on the models. The Gibbs sampling method provides a finer topic distribution over the documents (here it is a label distribution over the restaurants).
Lastly, it is worth noticing the effect of the PCA on the feature importance. For both the OVI and the GS, if we look back at the Pizza category, the results with the PCA are much more clear-cut. The weight of the most important feature and its difference with those of the other features are higher. We can explain this by the fact that the projection gave more structure to the data and reduced the noise.

This part was about confronting our data to supervised methods to grasp their patterns and also coming up with a good classification. We provided empirical results to confirm the differences between the two common LDA methods and proved that with the Gibbs sampling, the LDA can be really efficient at finding the underlying topics on a set of documents. It also performed much better than a matrix factorization method. Related to our problem, this showed us that if the numerical features provided by Yelp contained some information, the LDA features remained much more important when looking for the labels of a restaurant.

5. Unsupervised Clustering

5.1. Approach

For this part of the project, we use the same features as for the supervised classification approach, i.e. the topic assignments. We wanted to see if it was possible to retrieve the original classification only based on the reviews. To do so and in order to check if our classification attempt was successful, we sampled about 1000 restaurants having tags in the following list of 6 categories: Sushi, Steakhouse, Seafood, Mexican, Sports and Breakfast & Brunch. We then applied the following methodology:

Each restaurant is now represented as a probability distribution over the latent topics. Using the Shannon distance (symmetric Kullback-Liebler Divergence), we created a distance matrix gathering information on how close two restaurants are based on their reviews. After many experimentations, we chose to transform this distance matrix into a weighted adjacency matrix:

- $w_{ij} = 5$ if $d_{Shannon}(r_i, r_j) < 1$,
- $w_{ij} = 0.1$ otherwise.

The choice of having a non-zero weight between two restaurants even though they are not "close" is motivated by the fact that many graph clustering algorithm, in particular the WalkTrap, only work on a connected graph.

We then transformed this adjacency matrix in an unweighted graph and applied the Walktrap algorithm for community detection.

5.2. The WalkTrap Algorithm

The WalkTrap Algorithm is a community detection algorithm created by Pascal Pons and Matthieu Latapy [5]. The algortihm runs in time $O(n^2 \log n)$ and space $O(n^2)$ where n is the number of vertices in the graph. As mentioned in their paper, "the intuition behind the Walktrap is that random walks on a graph tend to get trapped into densely connected parts corresponding to communities".[5] It relies on a new metric used to evaluate the distance between two vertices in a graph:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{kj}^t)^2}{d(k)}}$$

where P_{ik}^t is the transition probability in t steps and $d(k)$ is the degree of node k (number of edges incident to the vertex). We can then generalise this distance between two vertices to a distance between two communities. Using this distance, we can now apply a hierarchical clustering algorithm.

Algorithm 2 Walktrap Algorithm

```
1: Start with a partition $\mathcal{P}_1 = \{\{v\}, v \in V\}$
2: Compute the distances between all adjacent vertices
3: while $\mathcal{P}_t \neq V$ do
4:     Merge 2 communities $\mathcal{C}_i$ and $\mathcal{C}_j$ in $\mathcal{P}_k$ and create a new partition $\mathcal{P}_{k+1}$
5:     Evaluate the distances between the communities in this new partition
```

The choice of communities to merge in step 4 is made using Ward's method. We look at minimising the mean within-cluster variance:

$$\frac{1}{n_{\mathcal{C}}} \sum_{\mathcal{C} \in \mathcal{P}_k} \sum_{i \in \mathcal{C}} r_{i\mathcal{C}}^2,$$

where $n_{\mathcal{C}}$ is the number of communities at iteration k, and $r_{i\mathcal{C}}$ is the distance from vertex i to its community $\mathcal{C}$.

5.3. Results

We ran the algorithm from the *igraph* [6] package in R on the three graphs generated with the three approaches mentionned above (NMF, OVI LDA, GS LDA). The results were extremely similar. We outputted a dendogram (cf. figure 12) representing the obtained clustering. We colored the tag of each business for a better readability. It is not yet entirely clear to us how to get the optimum number of clusters. We were puzzled to see that the Walktrap algorithm indicated that a cut into three big clusters represented the optimum partition. Knowing the structure of the sampled restaurants, we decided to further investigate the dendogram. As one can see on Figure 3, cutting the tree at different heights, we can obtain five clear colored communities, corresponding to five of the six tags we pre-selected.

We summarized this classification using a confusion matrix:

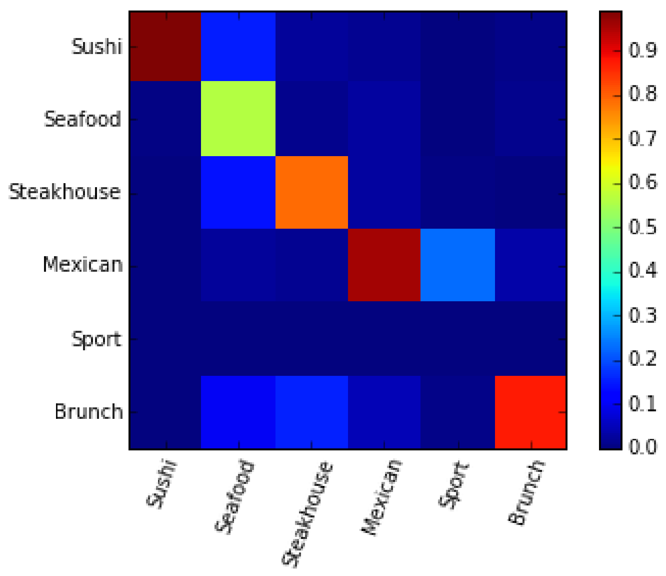


Figure 11: *Confusion Matrix using the Walktrap Algorithm*

With this method, we managed to detect 4 communities with great accuracy (over 80%). Nevertheless, this classification still has some flaws and the most obvious one is the misclassification or even non-classification of the Sports bars (yellow). We focused a bit more on these venues and explored manually the tags associated with them. We noticed that these Sports bar have most of the time other tags such as "Restaurants","Bar", "American" and some even "Mexican". Reviews for these venues might have been misallocated due to the proximity of types of food, drinks or "ambiance" with Mexican restaurants. It is nonetheless interesting to note that they were grouped almost all together inside the Mexican restaurants, and that if we cut the tree at the sixth level then they would all be gathered in 2 cliques.

It seems that this method will succeed in making a difference between communities of restaurants if they are almost exlusive. It seems very unlikely to discover a restaurant that one can identify as both a Mexican and a Sushi place or even a Sushi place specialising in Breakfast (unless its speciality is the takagoyaki, a type of Japanese omelette made by "rolling together several layers of cooked egg"). It was often the case that misclassified restaurants had the tag "Buffet" which is for our purpose of classification our worst ennemy as it will stand for many categories at once.

6. Next steps

So far, we managed to extract the categories of the restaurants as latent features with a Latent Dirichlet Allocation algorithm on the aggregated reviews for each restaurant.

Future work could focus on the sub-cliques and try to get a finer classification by combining them with other features like check-ins, parking availability, etc. As a variant, we could also try a supervised LDA, where we use the existing categories as a response variable associated with each document and infer the joint model of the documents and the responses. Lastly, as interpretability does not matter, we could try neural architectures to build the features.

7. conclusion

This study illustrates different methods to extract features from a set of documents. The results confirmed that the Latent Dirichlet Allocation manages to grasp the underlying features. The comparison with the supervised learning methods confirmed the importance of the features extraction part in a machine learning problem. It also corroborates the assumption that the features of the businesses could be used to infer their category labels. Finally, the unsupervised clustering process we developped provides encouraging results and could result in an automation tool for Yelp to label properly its database.

Code

The code for this project can be found at the following address:

www.github.com/nicodri/CS281

References

[1] M. Jordan D. Blei, A. Ng. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] D. Bach M. Hoffman, D. Blei. Online Learning for Latent Dirichlet allocation. 2010.

[3] Andre Wibisono Ashia C. Wilson Michael I. Jordan Tamara Broderick, Nicholas Boyd. Streaming Variational Bayes. 2013.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[5] M. Latapy P. Pons. Computing Communities in Large Networks Using Random Walks. *JGAA*, 10:191–218, 2006.

[6] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

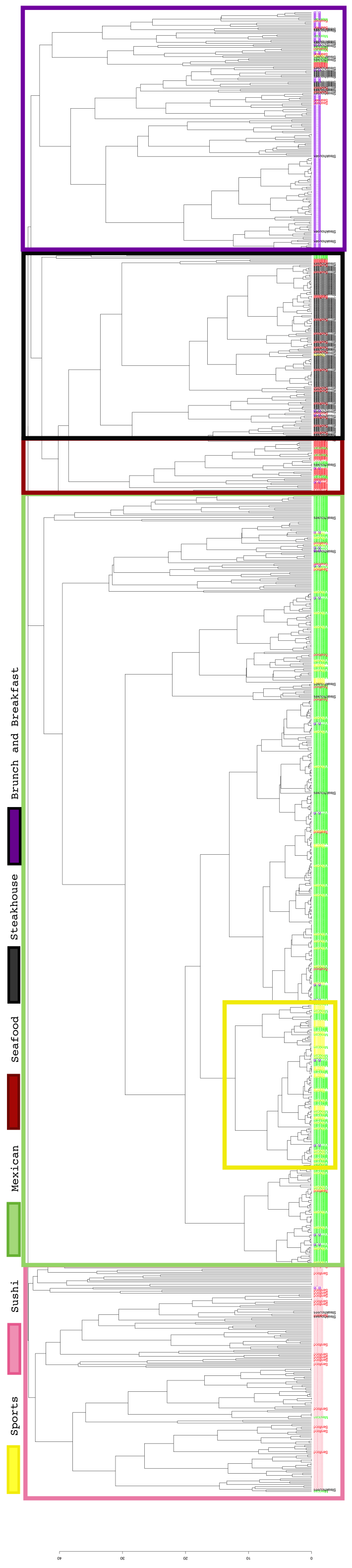


Figure 12: *Dendogram outputted by the Walktrap Algorithm*