Business Category Inference from Yelp data

Bhakti Khude
CMS 1406

Internal Guides
Dr. V. K. Jayaraman
Dr. Mihir Arjunwadkar
CMS, SPPU, Pune

External Guide
Dr. Yogesh Badhe

Persistent Systems, Pune

June 17, 2016

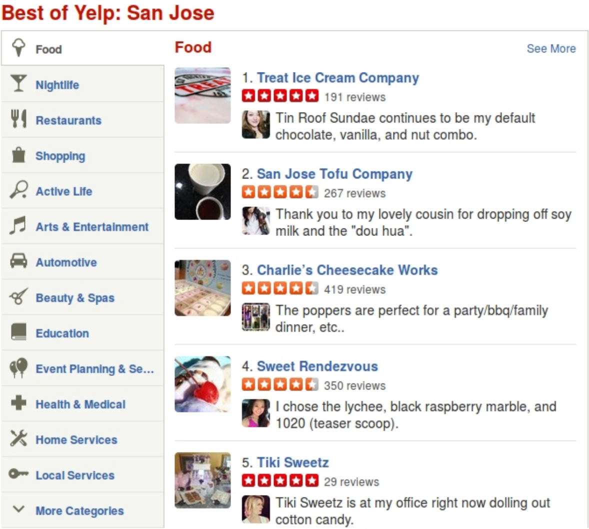
Outline

Outline

1. Introduction
   - About Yelp
   - Inferences from the Dataset
   - Problem Statement: Category Inference
   - The Dataset
     - Business dataset
     - Review & Checkin dataset
     - User & Tip dataset
     - Photo dataset

About Yelp

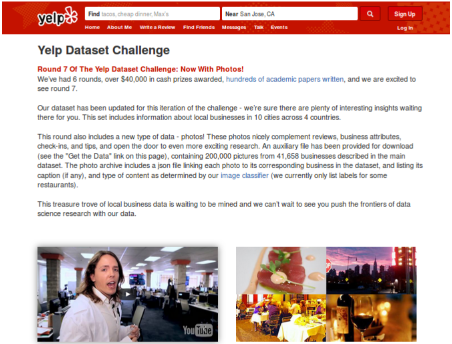
Yelp.com a crowd-sourced local business review and social networking site



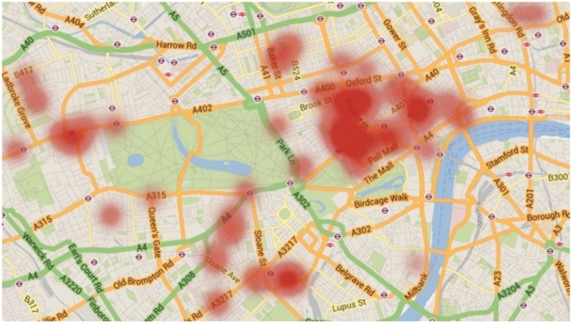
About Yelp

Yelp Dataset Challenge

- 2.2M reviews and 591K tips by 552K users for 77K businesses
- 566K business attributes, e.g. hours, parking availability, ambience
- Social network of 552K users for a total of 3.5M social edges
- Aggregated check-ins over time for each of the 77K businesses
- 2,00,000 pictures from the included businesses

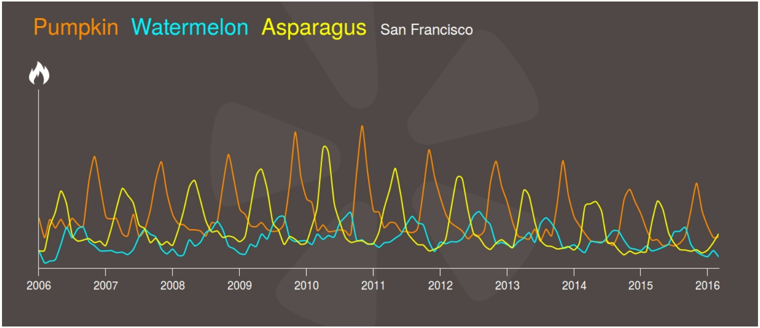


Cultural Trends

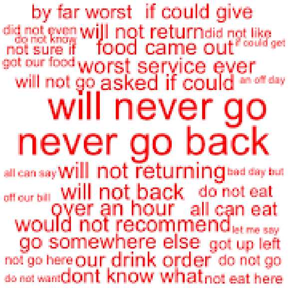


Yes, people do really use the word "posh" in London. *Image: Yelp*

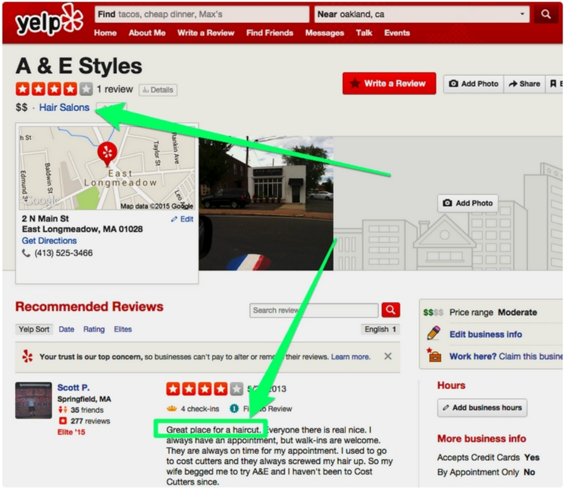
Seasonal Trends



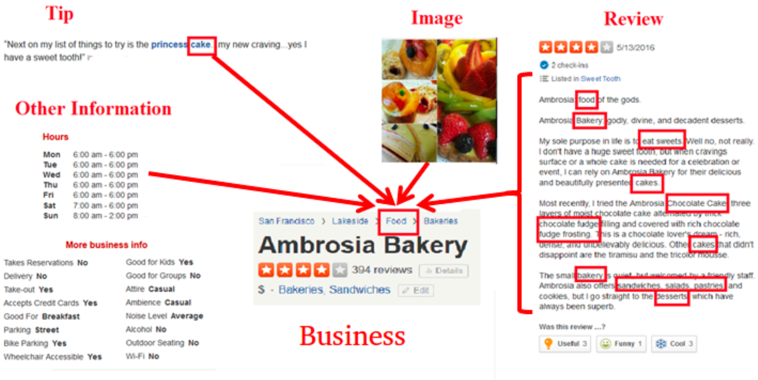
Review Sentiment Analysis



Prior Work


yelp
Find tacos, cheap dinner, Max's
Near oakland, ca
Home
About Me
Write a Review
Find Friends
Messages
Talk
Events
A & E Styles
1 review
Details
$$ · Hair Salons
Write a Review
Add Photo
Share
2 N Main St
East Longmeadow, MA 01028
Get Directions
(413) 525-3466
Edit
Add Photo
Recommended Reviews
Yelp Sort
Date
Rating
Elites
English 1
Price range Moderate
Edit business info
Hours
Add business hours
More business info
Accepts Credit Cards Yes
By Appointment Only No
Scott P.
Springfield, MA
35 friends
277 reviews
Elite '15
4 check-ins
Great place for a haircut.
always have an appointment, but walk-ins are welcome.
They are always on time for my appointment. I used to go
to cost cutters and they always screwed my hair up. So my
wife begged me to try A&E and I haven't been to Cost
Cutters since.

This Work



Automatically classify new business into relevant categories given the business attributes, reviews, tips and images.

Business dataset


```
{
    'type': 'business',
    'business_id': (encrypted business id),
    'name': (business name),
    'neighborhoods': [(hood names)],
    'full_address': (localized address),
    'city': (city),
    'state': (state),
    'latitude': latitude,
    'longitude': longitude,
    'stars': (star rating, rounded to half-stars),
    'review_count': review count,
    'categories': [(localized category names)]
    'open': True / False (corresponds to closed, not
       business hours),
    'hours': {
        (day_of_week): {
            'open': (HH:MM),
            'close': (HH:MM)  },
    },
    'attributes': {
        (attribute_name): (attribute_value), },
}
```

Review & Checkin dataset

```
{
    'type': 'review',
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'stars': (star rating, rounded to half-stars),
    'text': (review text),
    'date': (date, formatted like '2012-03-14'),
    'votes': {(vote type): (count)},
}
```

```
{
    'type': 'checkin',
    'business_id': (encrypted business id),
    'checkin_info': {
        '0-0': (number of checkins from 00:00 to 01:00 on
            all Sundays),
        '1-0': (number of checkins from 01:00 to 02:00 on
            all Sundays),
        ...
        '14-4': (number of checkins from 14:00 to 15:00 on
            all Thursdays),
        ...
        '23-6': (number of checkins from 23:00 to 00:00 on
            all Saturdays)
    }, # if there was no checkin for a hour-day block it
        will not be in the dict
}
```

User & Tip dataset

```
{
    'type': 'user',
    'user_id': (encrypted user id),
    'name': (first name),
    'review_count': (review count),
    'average_stars': (floating point average, like 4.31),
    'votes': {(vote type): (count)},
    'friends': [(friend user_ids)],
    'elite': [(years_elite)],
    'yelping_since': (date, formatted like '2012-03'),
    'compliments': {
        (compliment_type): (num_compliments_of_this_type),
        ...
    },
    'fans': (num_fans),
}
```

```
{
    'type': 'tip',
    'text': (tip text),
    'business_id': (encrypted business id),
    'user_id': (encrypted user id),
    'date': (date, formatted like '2012-03-14'),
    'likes': (count),
}
```

Photo dataset

```
[
    {
        "photo_id": (encrypted photo id),
        "business_id" : (encrypted business id),
        "caption" : (the photo caption, if any),
        "label" : (the category the photo belongs to, if any
           )
    },
    {...}
]
```

Photo-to-Business linking

```
{
        "photo_id":"-2G-S95e-6Q5lOXgRxqaZA",
        "business_id":"UUD-BLty8HGzqj5cROo34g",
        "caption":"10 toppings few classic yogurt flavors.",
        "label":"food"
}
```

-2G-S95e-6Q5lOXgRxqaZA.jpg

Outline

2. Data Extraction
   - Exploratory Analysis
   - Final Dataset Preparation

Data Extraction

- 5 text files containing information about Business, User Reviews, User Information, Tip and Checkins.
- Implementation initially in R and then in Python
- Extracted the data from JSON in a CSV format for processing it in R via the `jsonlite` package.
- Stored the data in MySQL database and accessing it in R using `RMySQL` package.

1. Total 77,445 businesses

Data Extraction

- 5 text files containing information about Business, User Reviews, User Information, Tip and Checkins.
- Implementation initially in R and then in Python
- Extracted the data from JSON in a CSV format for processing it in R via the `jsonlite` package.
- Stored the data in MySQL database and accessing it in R using `RMySQL` package.

1. Total 77,445 businesses
2. 22,25,231 reviews & 5,91,864 tips

Data Extraction

- 5 text files containing information about Business, User Reviews, User Information, Tip and Checkins.
- Implementation initially in R and then in Python
- Extracted the data from JSON in a CSV format for processing it in R via the `jsonlite` package.
- Stored the data in MySQL database and accessing it in R using `RMySQL` package.

1. Total 77,445 businesses
2. 22,25,231 reviews & 5,91,864 tips
3. 55,569 check-ins & 2,00,000 images

Data Extraction

- 5 text files containing information about Business, User Reviews, User Information, Tip and Checkins.
- Implementation initially in R and then in Python
- Extracted the data from JSON in a CSV format for processing it in R via the `jsonlite` package.
- Stored the data in MySQL database and accessing it in R using `RMySQL` package.

1. Total 77,445 businesses
2. 22,25,231 reviews & 5,91,864 tips
3. 55,569 check-ins & 2,00,000 images
4. by 5,52,339 users

Data Extraction

Business Name	Business Category
Mr Hoagie	Fast Food, Restaurants
Clancy's Pub	Nightlife
Joe Cislo's Auto	Auto Repair, Automotive
Cool Springs Golf Center	Active Life, Mini Golf, Golf
Verizon	Shopping, Home Services, Internet Service Providers, Mobile Phones, Professional Services, Electronics
Greentree Animal Clinic	Veterinarians, Pets
Kings Family Restaurant	Burgers, Breakfast & Brunch, American (Traditional), Restaurants

Business category inference based on Supervised learning approach.

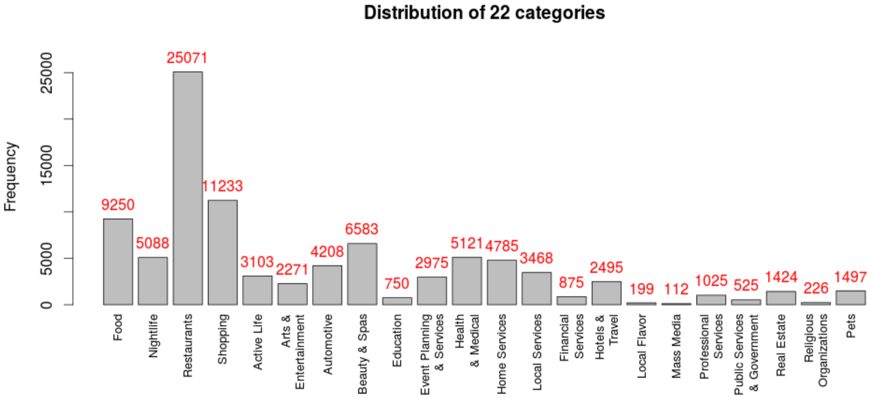
- 896 unique business categories

Business Name	Business Category
Mr Hoagie	Fast Food, Restaurants
Clancy's Pub	Nightlife
Joe Cislo's Auto	Auto Repair, Automotive
Cool Springs Golf Center	Active Life, Mini Golf, Golf
Verizon	Shopping, Home Services, Internet Service Providers, Mobile Phones, Professional Services, Electronics
Greentree Animal Clinic	Veterinarians, Pets
Kings Family Restaurant	Burgers, Breakfast & Brunch, American (Traditional), Restaurants

Business category inference based on Supervised learning approach.

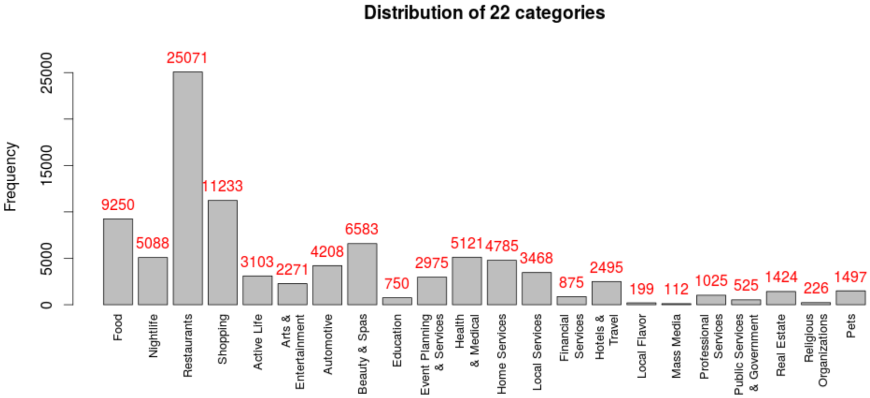
- 896 unique business categories
- Each business belongs to min. 1 and max. 11 categories.

Data Extraction

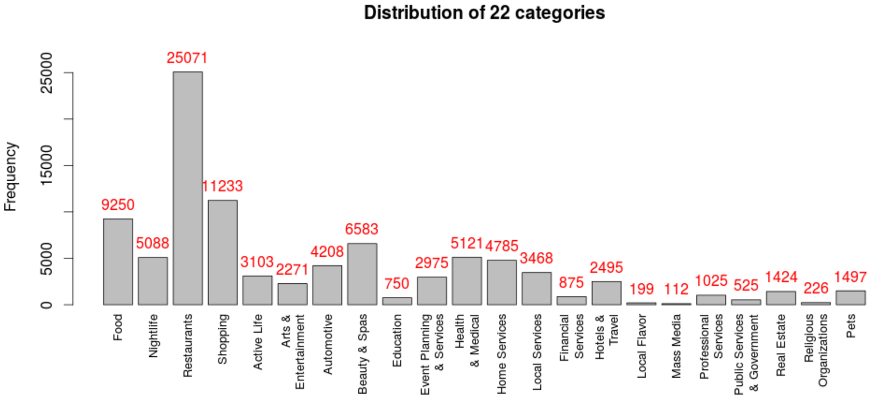


- Each business belonged to multiple categories from these 22 categories

Data Extraction



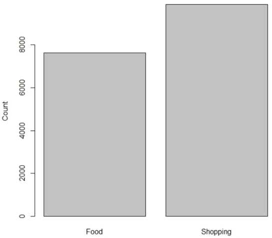
- Each business belonged to multiple categories from these 22 categories
- Multi Label Classification Problem



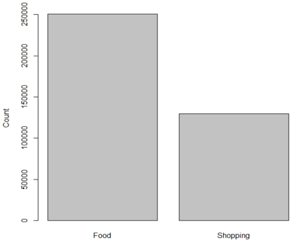
- Each business belonged to multiple categories from these 22 categories
- Multi Label Classification Problem
- “Food” & “Shopping” chosen

Exploratory Analysis

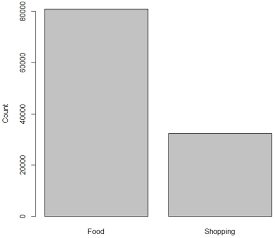
Business Count



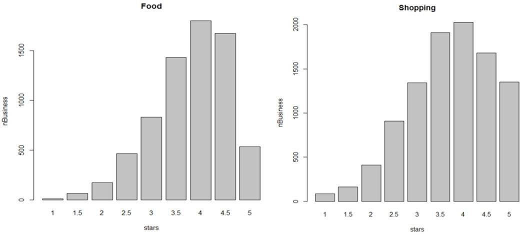
Review Count



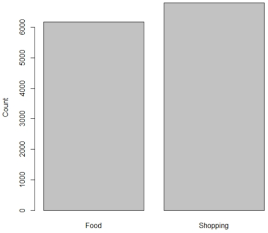
Tip Count



Star Rating



Checkin Count



Final Dataset Preparation

- Chosen categories Food and Shopping
- 19,774 businesses in total
  - 10,524 - Shopping
  - 8,541 - Food
  - 709 - Both
- On removal of **closed** businesses
  16,864 businesses in total
  - 9,238 - Shopping
  - 6,978 - Food
  - 648 - Both
  - Review count: 3,40,062
    Max 5 reviews per business: 73,576
  - Tip count: 1,08,910
    Max 5 tips per business: 36,517

	Shopping	Food
Business	9,886	7,626
Review	42,274	34,225
Tip	16,561	21,811
Check-in	7,403	6,774

Outline

3. Multi Label Classification
   - Problem Transformation Methods
   - Algorithm Adaptation Methods

Instance	Label Set
1	$\{\lambda_2, \lambda_3\}$
2	$\{\lambda_1\}$
3	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	$\{\lambda_2, \lambda_4\}$

Table : *Example Multi-label dataset*

- In multi-label classification each instance will be associated to a set of labels instead of a single label.
  Applications : text categorization, medical diagnosis, music categorization

Multi Label Classification

Instance	Label Set
1	$\{\lambda_2, \lambda_3\}$
2	$\{\lambda_1\}$
3	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	$\{\lambda_2, \lambda_4\}$

Table : *Example Multi-label dataset*

- `In multi-label classification each instance will be associated to a set of labels instead of a single label.`
  Applications : text categorization, medical diagnosis, music categorization
- Organized in three main families:
  1. Problem transformation : Transform the multi-label learning problem into one or several single-label classification problems
  2. Algorithm adaptation : Extend single-label learning algorithms for the multi-label data
  3. Ensemble methods : Use ensembles of classifiers

Problem Transformation

Instance	Label Set
1	$\{\lambda_2, \lambda_3\}$
2	$\{\lambda_1\}$
3	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	$\{\lambda_2, \lambda_4\}$

Table : *Example Multi-label dataset*

Ex.	Label
1a	λ_2
1b	λ_3
2	λ_1
3a	λ_1
3b	λ_2
3c	λ_3
4a	λ_2
4b	λ_4

(a)

Ex.	Label
1	λ_3
2	λ_1
3	λ_2
4	λ_4

(b)

Ex.	Label
2	λ_1

(c)

Ex.	Label Set
1	$\lambda_{2,3}$
2	λ_1
3	$\lambda_{1,2,3}$
4	$\lambda_{2,4}$

(d)

(a) copy (b) select-random (c) ignore (d) Label Power Set

Ranking by Pairwise Comparison (RPC)

Instance	Label Set
1	$\{\lambda_2, \lambda_3\}$
2	$\{\lambda_1\}$
3	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	$\{\lambda_2, \lambda_4\}$

Table : *Example Multi-label dataset*

Ex.	Label
1	$\lambda_{\neg 1,2}$
2	$\lambda_{1,\neg 2}$
4	$\lambda_{\neg 1,2}$

(a)

Ex.	Label
1	$\lambda_{\neg 1,3}$
2	$\lambda_{1,\neg 3}$
4	$\lambda_{\neg 1,\neg 3}$

(b)

Ex.	Label
2	$\lambda_{1,\neg 4}$
3	$\lambda_{1,\neg 4}$
4	$\lambda_{\neg 1,4}$

(c)

Ex.	Label
4	$\lambda_{2,\neg 3}$

(d)

Table : *Datasets transformed by RPC method*

Binary Relevance

Instance	Label Set
1	$\{\lambda_2, \lambda_3\}$
2	$\{\lambda_1\}$
3	$\{\lambda_1, \lambda_2, \lambda_3\}$
4	$\{\lambda_2, \lambda_4\}$

Table : *Example Multi-label dataset*

Ex.	Label
1	$\neg\lambda_1$
2	λ_1
3	λ_1
4	$\neg\lambda_1$

(a)

Ex.	Label
1	λ_2
2	$\neg\lambda_2$
3	λ_2
4	λ_2

(b)

Ex.	Label
1	λ_3
2	$\neg\lambda_3$
3	λ_3
4	$\neg\lambda_3$

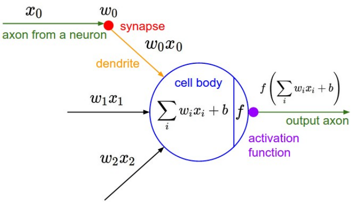
(c)

Ex.	Label
1	$\neg\lambda_4$
2	$\neg\lambda_4$
3	$\neg\lambda_4$
4	λ_4

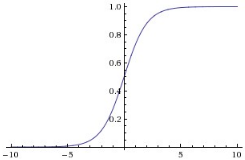
(d)

Table : *Transformed data sets produced by Binary Relevance (BR) method*

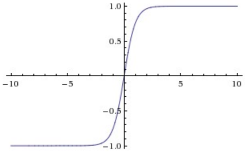
Neural Networks:



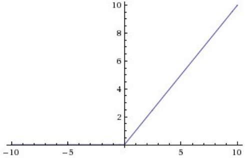
Activation Functions



sigmoid



hyperbolic tangent (tanh)

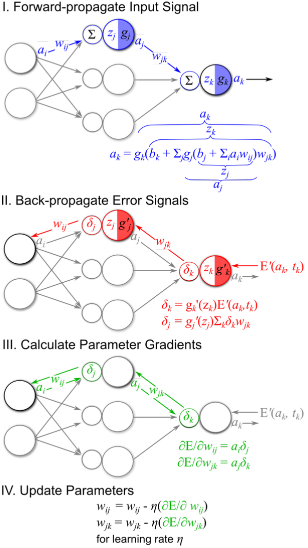


rectified linear unit (relu)

Neural Networks: Backpropagation for Multi-Label Learning (BP-MLL)

The 4 main steps of the BP algorithm are:

1. Forward propagate error signals to output
2. Calculate output error E, and backpropagate error signal
3. Use forward signal and backward signals to calculate parameter gradients
4. Update network parameters



Parameters

z_j : input to node j for layer l

g_j : activation function for node j in layer l (applied to z_j)

$a_j = g_j(z_j)$: ouput/activation of node j in layer l

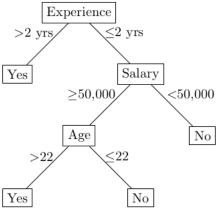
w_{ij} : weights connecting node i in layer (l-1) to node j in layer l

b_j : bias for unit j in layer l

t_k : target value for node k in the output layer

Decision Tree Algorithm

C5.0 and Random Forest algorithms are based on Decision Tree learning.



- Classify instances by sorting them down the tree
- Node in the tree specifies a test of some attribute of the instance
- Attribute selection: Based on Information Gain and Entropy

$$Entropy = \sum_j -p_j \log_2 p_j, \quad p_j \quad \text{is the probability of class} \quad j$$

$$InfoGain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where *Values(A)* is the set of all possible values for attribute A, and S_v is the subset of S for which attribute A has value v.

- C4.5 works with continuous attributes, handles missing values, supports pruning
- C5.0 supports boosting, pre-filter attributes, improved scalability

C5.0 was used in Problem Transformation
Clare et. al uses C4.5 algorithm for multi-label data with the modified entropy definition:

$$Entropy = -\sum_{j=1}^{k}\{(P(\lambda_j)\log P(\lambda_j) + (1 - P(\lambda_j))\log(1 - P(\lambda_j))\}$$

where, $P(\lambda_j)$ = probability of class λ_j. This allows to estimate the uncertainty in terms of number of bits in multi-label setting. This modified method also allows multiple labels at the leaves.

Random Forests

- Ensemble of decision trees
- Random subset of features are selected to generate a forest
- Every example passes through all trees, get the class having highest votes

Outline

4. Dimensionality Reduction

Singular Value Decomposition

The SVD of matrix A having dimensions $m \times n$:

$$A = UWV^T$$

where,
$U =$ The columns of U are the eigenvectors of AA^T. Dimension: $m \times min(m, n)$
$W =$ Vector having singular values of A. They are the square root of the eigenvalues of both AA^T and A^TA. Dimension: $min(m, n)$
$V =$ The columns of V are the eigenvectors of A^TA. Dimension: $n \times min(m, n)$

- A matrix of dimension $m(documents) \times n(terms)$
- Keep first k $(k < n)$number of terms
- Drop all the columns in V beyond k
- V matrix is truncated to $n \times k$

Data projection:

$$A^{'} = A_{[m \times n]} \times V^{'}_{[n \times k]}$$

$$A^{'} = A_{[m \times k]}$$

Outline

5 Evaluation Measures

Evaluation Measures

D being the Multi Label Dataset (MLD), L the full set of labels used in D, Y_i the subset of predicted labels for the ith instance. and Z_i the true subset of labels.

1. Hamming Loss (**HLoss**):

$$HammingLoss = \frac{1}{\mid D \mid} \sum_{i=1}^{|D|} \frac{\mid Y_i \Delta Z_i \mid}{\mid L \mid}$$

2. Accuracy (**Acc**):

$$Accuracy = \frac{1}{\mid D \mid} \sum_{i=1}^{|D|} \frac{\mid Y_i \cap Z_i \mid}{\mid Y_i \cup Z_i \mid}$$

3. Subset Accuracy (**subAcc**):

$$SubsetAccuracy = \frac{1}{\mid D \mid} \sum_{i=1}^{|D|} [\![Y_i = Z_i]\!]$$

1. Precision (**Prec**):

$$Precision = \frac{1}{\mid D \mid} \sum_{i=1}^{|D|} \frac{\mid Y_i \cap Z_i \mid}{\mid Y_i \mid}$$

2. Recall (**Rec**):

$$Precision = \frac{1}{\mid D \mid} \sum_{i=1}^{|D|} \frac{\mid Y_i \cap Z_i \mid}{\mid Z_i \mid}$$

3. F1-measure (**F1**):

$$F1 - measure = 2 * \frac{Precision \cdot Recall}{Precision + Recall}$$

4. Area Under the Curve (**AUC**): This metric calculates the area under the curve ROC (Receiver Operating Charateristic) by trapezoidal rule of integration.

$$MicroAUC = \frac{1}{\mid L \mid} \frac{\mid x', x".y', y" : rank(x', y') \geq rank(x", y"), (x', y') \in S^{+}, (x", y") \in S^{-} \mid}{\mid S^{+} \mid\mid S^{-} \mid}$$

where, $rank(x_i, y)$ returns the position of y, a certain label, in the x_i instance.

$$S^{+} = (x_i, y) : y \in Y_i, \quad S^{-} = (x_i, y) : y \notin Y_i$$

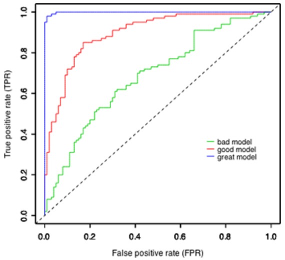
ROC curve is a graphical representation that measures performance of a binary classifier system as its discrimination threshold is varied. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings, where,

$$TPR = \frac{TP}{TP + FN}$$

and

$$FPR = \frac{FP}{FP + TN}$$

		Predicted Condition	
		Predicted Condition positive	Predicted Condition negative
True condition	condition positive	**True Positive (TP)**	**False Negative (FN)**
	condition negative	**False Positve (FP)**	**True Negative (TN)**



Outline

6 Data Preprocessing, Modeling and Results

- Business Dataset
  - Text features
  - Time features
  - Check-in features
- Review Dataset
  - User features
  - Word2vec features
- Tip Dataset
- Image Dataset

Data cleaning was performed in R and Python

Using R

- `tm` package.
- Conversion to lowercase, removal of numbers, removal of punctuations, removal of common English "stopwords" (the, is, was etc.)
- "Stemming" was performed which truncates words to their specific stems(e.g., "compute", "computes" and "computing" all become "comput")
- SnowballC package performs stemming by Porter stemmer algorithm

Using Python

- `NLTK` package used.

Using Python

- `NLTK` package used.
- Foreign accents removed: café was converted to cafe

Using Python

- `NLTK` package used.
- Foreign accents removed: café was converted to cafe
- Text converted to lowercase, numbers, punctuation and English stopwords removed as before.

Using Python

- NLTK package used.
- Foreign accents removed: café was converted to cafe
- Text converted to lowercase, numbers, punctuation and English stopwords removed as before.
- Custom stopwords ('http', 'www', 'net', 'com', 'jpg', 'etc', 'like', 'likes', 'still', 'definitely', 'always','feel', 'feels', 'difficult', 'different', 'usual', 'usually') were added to the existing list.

Using Python

- `NLTK` package used.
- Foreign accents removed: café was converted to cafe
- Text converted to lowercase, numbers, punctuation and English stopwords removed as before.
- Custom stopwords ('http', 'www', 'net', 'com', 'jpg', 'etc', 'like', 'likes', 'still', 'definitely', 'always','feel', 'feels', 'difficult', 'different', 'usual', 'usually') were added to the existing list.
- Spelling correction using `pyEnchant` package

Using Python

- `NLTK` package used.
- Foreign accents removed: café was converted to cafe
- Text converted to lowercase, numbers, punctuation and English stopwords removed as before.
- Custom stopwords (‘http’, ‘www’, ‘net’, ‘com’, ‘jpg’, ‘etc’, ‘like’, ‘likes’, ‘still’, ‘definitely’, ‘always’,‘feel’, ‘feels’, ‘difficult’, ‘different’, ‘usual’, ‘usually’) were added to the existing list.
- Spelling correction using `pyEnchant` package
- Lemmatisation was done to get the correct stem of the word. It gives a real word rather than just stemming. Consider the following examples
  eg 1: The word "better" has "good" as its lemma. This link is missed by stemming, as it requires a dictionary look-up.
  eg 2: The word "walk" is the base form for word "walking", and hence this is matched in both stemming and lemmatisation.

Using Python

- NLTK package used.
- Foreign accents removed: café was converted to cafe
- Text converted to lowercase, numbers, punctuation and English stopwords removed as before.
- Custom stopwords (‘http’, ‘www’, ‘net’, ‘com’, ‘jpg’, ‘etc’, ‘like’, ‘likes’, ‘still’, ‘definitely’, ‘always’,‘feel’, ‘feels’, ‘difficult’, ‘different’, ‘usual’, ‘usually’) were added to the existing list.
- Spelling correction using `pyEnchant` package
- Lemmatisation was done to get the correct stem of the word. It gives a real word rather than just stemming. Consider the following examples
  eg 1: The word "better" has "good" as its lemma. This link is missed by stemming, as it requires a dictionary look-up.
  eg 2: The word "walk" is the base form for word "walking", and hence this is matched in both stemming and lemmatisation.
- Parts-of-Speech (POS) Tagging using `nltk.pos_tag()` using the default `maxent_treebank_pos_tagger` tagger. It is a Maximum Entropy tagger which has been trained on Penn Treebank tagset. Only the noun, adjective and verb tags were retained.

Using Python

- NLTK package used.
- Foreign accents removed: café was converted to cafe
- Text converted to lowercase, numbers, punctuation and English stopwords removed as before.
- Custom stopwords ('http', 'www', 'net', 'com', 'jpg', 'etc', 'like', 'likes', 'still', 'definitely', 'always','feel', 'feels', 'difficult', 'different', 'usual', 'usually') were added to the existing list.
- Spelling correction using pyEnchant package
- Lemmatisation was done to get the correct stem of the word. It gives a real word rather than just stemming. Consider the following examples
  eg 1: The word "better" has "good" as its lemma. This link is missed by stemming, as it requires a dictionary look-up.
  eg 2: The word "walk" is the base form for word "walking", and hence this is matched in both stemming and lemmatisation.
- Parts-of-Speech (POS) Tagging using nltk.pos_tag() using the default maxent_treebank_pos_tagger tagger. It is a Maximum Entropy tagger which has been trained on Penn Treebank tagset. Only the noun, adjective and verb tags were retained.
- Stemming was performed like before.

N-gram is a continuous sequence of N items that co-occur together from a given text sequence.
E.g.: “a man is walking”
Unigrams from this sentence:

- a
- man
- is
- walking

Bigrams from this sentence:

- a man
- man is
- is walking

TF-IDF weighting Score the importance of words (or "terms") in a document
Term Frequency: TF(t, d) : No. of times term t occurs in document d

$$\text{TF}(t, d) = 1 \text{ if } t \text{ occurs in } d \text{ and } 0 \text{ otherwise}$$

Inverse Document Frequency: IDF(t, d) : Whether the term t is common or rare across all documents

$$\text{IDF}(t, d) = \log_2 \frac{N}{|\{d \in D : t \in d\}|}$$

where

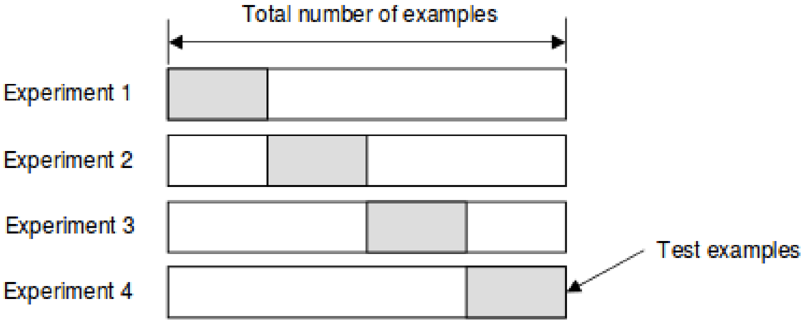
$$\text{N} = |D| \text{ total no. of documents } |\{d \in D : t \in d\}| \text{ is no. of documents where term } t \text{ appears}$$

A high IDF score implies it is a rare term.
TF-IDF score is simply the product of TF and IDF. Ideally, a term is important to a document if it has high TF and high IDF scores.

```
Text = c( "Hello world!!","Hello 123",  "The man is walking!!" )

     Terms
Docs       hello       man      walk     world
   1 0.2924813 0.0000000 0.0000000 0.7924813
   2 0.5849625 0.0000000 0.0000000 0.0000000
   3 0.0000000 0.7924813 0.7924813 0.0000000
```

k Fold Cross Validation



Business Dataset

Vector Space Model was built from business names.

Time feature generation

"Starbucks"

Day	Open	Close
Monday	06:00	21:30
Tuesday	06:00	21:30
Wednesday	06:00	21:30
Thursday	06:00	22:30
Friday	07:00	22:30
Saturday	07:00	21:00
Sunday	06:00	21:30

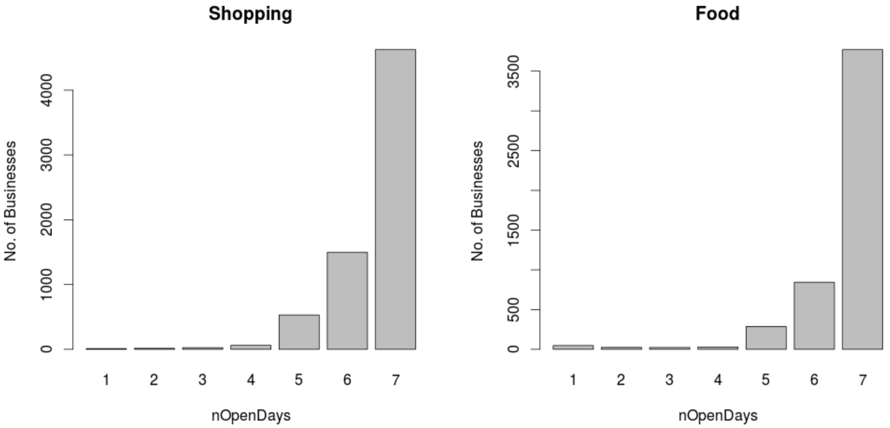
"Minerva Bakery"

Day	Open	Close
Monday	NA	NA
Tuesday	07:00	16:30
Wednesday	07:00	16:30
Thursday	07:00	16:30
Friday	07:00	16:30
Saturday	07:00	15:30
Sunday	NA	NA

Time slots created: `Morning` : 05:00 - 12:00, `Afternoon` : 12:00 - 17:00, `Evening`: 17:00 - 21:00, `Night`: 21:00 - 05:00

Business	Mor	Aft	Eve	Nit	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Starbucks	1	1	1	1	1	1	1	1	1	1	1
Minerva Bakery	1	1	0	0	0	1	1	1	1	1	0

Time feature generation



Check-in feature generation

Check-in given for every hour for every day of the week. These account for a total of 168 features ($24 \times 7 = 168$).
Generated 24 features (1 feature for every hour of the day)

Day	"9-10"	"10-11"	"11-12"
Monday	1	2	3
Tuesday	1	1	1
Wednesday	4	1	3
Thursday	4	2	1
Friday	4	4	4
Saturday	NA	2	3
Sunday	2	2	7

Check-in feature generation

Check-in given for every hour for every day of the week. These account for a total of 168 features ($24 \times 7 = 168$).
Generated 24 features (1 feature for every hour of the day)

Day	"9-10"	"10-11"	"11-12"
Monday	1	2	3
Tuesday	1	1	1
Wednesday	4	1	3
Thursday	4	2	1
Friday	4	4	4
Saturday	NA	2	3
Sunday	2	2	7

	"9"	"10"	"11"
"Starbucks"	16	14	22

Data cleaned by R
Model: C5.0 Problem Transformation

Model	HLoss	Acc	AUC	F1	Prec	Rec
1	0.200	0.800	0.800	0.807	0.812	0.801
2	0.185	0.815	0.815	0.820	0.825	0.816
3	0.150	0.850	0.850	0.855	0.856	0.855
4	0.143	0.857	0.857	0.861	0.865	0.858
5	0.143	0.857	0.857	0.862	0.864	0.859
6	0.140	0.860	0.859	0.864	0.868	0.861
7	0.141	0.859	0.859	0.864	0.865	0.862

Table : *`C5.0` results on **business** dataset (I)*

Missing values did carry valuable information to category inference indicated by decrease in the HLoss value.

Data cleaned by R
Model: C5.0 Problem Transformation

Model	HLoss	Acc	AUC	F1	Prec	Rec
1	0.200	0.800	0.800	0.807	0.812	0.801
2	0.185	0.815	0.815	0.820	0.825	0.816
3	0.150	0.850	0.850	0.855	0.856	0.855
4	0.143	0.857	0.857	0.861	0.865	0.858
5	0.143	0.857	0.857	0.862	0.864	0.859
6	0.140	0.860	0.859	0.864	0.868	0.861
7	0.141	0.859	0.859	0.864	0.865	0.862

Table : *`C5.0` results on **business** dataset (I)*

Missing values did carry valuable information to category inference indicated by decrease in the HLoss value.

Data cleaned by R
Model: DNN Algorithm Adaptation

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	200,250	100	-	-	0.145	0.855	0.854	0.863	0.847	0.880
2	200,80	100	-	-	0.142	0.858	0.858	0.864	0.862	0.866
3	100,50	100	-	-	0.146	0.854	0.853	0.860	0.853	0.867
4	100,50	100	0.1	-	0.145	0.855	0.855	0.860	0.861	0.867
5	100,50	100	0.2	-	0.153	0.847	0.847	0.852	0.855	0.850
6	100,50	100	0.3	-	0.148	0.852	0.852	0.859	0.853	0.865
7	100,50	100	0.1	0.1	0.150	0.850	0.849	0.858	0.844	0.873
8	100,50	100	0.2	0.1	0.149	0.851	0.850	0.858	0.848	0.871

Table : *DNN results on **business** dataset (I)*

Model-3 was selected for further development because it had optimum results and a good Recall value.

Data cleaned by R
Model: DNN Algorithm Adaptation

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	200,250	100	-	-	0.145	0.855	0.854	0.863	0.847	0.880
2	200,80	100	-	-	0.142	0.858	0.858	0.864	0.862	0.866
3	100,50	100	-	-	0.146	0.854	0.853	0.860	0.853	0.867
4	100,50	100	0.1	-	0.145	0.855	0.855	0.860	0.861	0.867
5	100,50	100	0.2	-	0.153	0.847	0.847	0.852	0.855	0.850
6	100,50	100	0.3	-	0.148	0.852	0.852	0.859	0.853	0.865
7	100,50	100	0.1	0.1	0.150	0.850	0.849	0.858	0.844	0.873
8	100,50	100	0.2	0.1	0.149	0.851	0.850	0.858	0.848	0.871

Table : *DNN results on **business** dataset (I)*

Model-3 was selected for further development because it had optimum results and a good Recall value.

Missing Value Imputation

Special attribute imputation

Attribute	Missing Information (%)
Accepts Credit Cards	06.469
Price Range	06.908
Parking Garage	12.885
Parking Valet	12.945
Parking Street	12.951
Parking Lot	12.951
Parking Validated	13.722
Wheelchair Accessible	62.951

Table : *Special attribute missing information*

63 features having missing content $\geq 70\%$ were ignored and removed
Time feature imputation
Check-in feature imputation

Missing value imputation by machine learning procedure:

1. One attributed is imputed each time.
2. Dataset is divided in two sections training and testing. Training set includes the examples without the missing value attribute. Testing set includes examples with the missing value attribute.
3. Classifier is trained on the training set with output as the attribute that needs imputation.
4. Missing value is predicted for the testing set.
5. Predicted value is imputed to the missing field in the original data.
6. Process is repeated for all the attributes having missing values.

Data cleaned by R
Model: DNN Algorithm Adaptation

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
9	100,50	100	-	-	0.116	0.884	0.884	0.889	0.885	0.894
10	100,50	100	-	-	0.100	0.900	0.900	0.903	0.905	0.902
11	180,150,100	100	-	-	0.106	0.894	0.894	0.898	0.897	0.900
12	150,70,150	100	-	-	0.103	0.897	0.897	0.901	0.904	0.898
13	100,50	100	-	-	0.188	0.812	0.812	0.820	0.817	0.822
14	100,50	100	-	-	0.107	0.893	0.893	0.896	0.899	0.894

Table : *DNN results on **business** dataset (I)*

Data cleaned by R
Model: DNN Algorithm Adaptation

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
9	100,50	100	-	-	0.116	0.884	0.884	0.889	0.885	0.894
10	100,50	100	-	-	0.100	0.900	0.900	0.903	0.905	0.902
11	180,150,100	100	-	-	0.106	0.894	0.894	0.898	0.897	0.900
12	150,70,150	100	-	-	0.103	0.897	0.897	0.901	0.904	0.898
13	100,50	100	-	-	0.188	0.812	0.812	0.820	0.817	0.822
14	100,50	100	-	-	0.107	0.893	0.893	0.896	0.899	0.894

Table : *DNN results on **business** dataset (I)*

Data cleaned by Python
Model: C5.0 & DNN

Model	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	0.145	0.855	0.855	0.861	0.858	0.864
2	W	-	0.158	0.842	0.841	0.847	0.847	0.848
3	100,50	300	0.082	0.918	0.918	0.921	0.923	0.919
4	100,50	300	0.081	0.919	0.919	0.922	0.924	0.920

Table : *C5.0 and DNN results on **business** dataset (II)*

Business Dataset: Results-IV

Data cleaned by Python
Model: C5.0 & DNN

Model	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	0.145	0.855	0.855	0.861	0.858	0.864
2	W	-	0.158	0.842	0.841	0.847	0.847	0.848
3	100,50	300	0.082	0.918	0.918	0.921	0.923	0.919
4	100,50	300	0.081	0.919	0.919	0.922	0.924	0.920

Table : *C5.0 and DNN results on **business** dataset (II)*

Vector Space model from unigrams and bigrams built from review text
In addition 3 more attributes. No missing data.
Non-Elite Dataset: Data cleaned by R

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.105	0.895	0.896	0.899	0.904	0.894
2	100,50	100	-	-	0.101	0.899	0.899	0.902	0.909	0.896
3	100,50	100	0.2	-	0.100	0.900	0.901	0.903	0.914	0.892
4	100,50	300	-	-	0.084	0.916	0.916	0.918	0.924	0.913
5	100,50	300	0.2	-	0.098	0.902	0.902	0.905	0.909	0.902
6	100,50	300	-	0.2	0.100	0.900	0.900	0.903	0.911	0.896
7	100,50	300	-	-	0.089	0.911	0.911	0.914	0.919	0.909

Table : *C5.0 and DNN results on **review** dataset (I)*

Vector Space model from unigrams and bigrams built from review text
In addition 3 more attributes. No missing data.
Non-Elite Dataset: Data cleaned by R

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.105	0.895	0.896	0.899	0.904	0.894
2	100,50	100	-	-	0.101	0.899	0.899	0.902	0.909	0.896
3	100,50	100	0.2	-	0.100	0.900	0.901	0.903	0.914	0.892
4	100,50	300	-	-	0.084	0.916	0.916	0.918	0.924	0.913
5	100,50	300	0.2	-	0.098	0.902	0.902	0.905	0.909	0.902
6	100,50	300	-	0.2	0.100	0.900	0.900	0.903	0.911	0.896
7	100,50	300	-	-	0.089	0.911	0.911	0.914	0.919	0.909

Table : *C5.0 and DNN results on **review** dataset (I)*

Non-Elite Dataset: Data cleaned by Python

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.105	0.895	0.895	0.898	0.905	0.891
2	100,50	300	-	-	0.083	0.917	0.918	0.919	0.930	0.909
3	100,50	300	-	-	0.101	0.899	0.900	0.902	0.911	0.894

Table : *C5.0 and DNN results on **review** dataset (II)*

Non-Elite Dataset: Data cleaned by Python

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.105	0.895	0.895	0.898	0.905	0.891
2	100,50	300	-	-	0.083	0.917	0.918	0.919	0.930	0.909
3	100,50	300	-	-	0.101	0.899	0.900	0.902	0.911	0.894

Table : *C5.0 and DNN results on **review** dataset (II)*

User features

An "Elite" user serves as a local expert.
More the number of years he is Elite more trusted Yelper

User	Elite years	nElite
1	2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015	11
2	-	0
3	2005, 2006, 2007, 2008, 2010, 2011, 2012	7

Table : *User dataset example*

Elite Dataset: Data cleaned by Python

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.098	0.902	0.902	0.905	0.910	0.901
2	W	-	-	-	0.105	0.895	0.895	0.899	0.903	0.896
3	100,50	300	-	-	0.075	0.925	0.925	0.927	0.931	0.924
4	100,50	300	0.2	-	0.081	0.919	0.919	0.922	0.928	0.916

Table : *`C5.0` and `DNN` results on **review** dataset (III)*

User features

An "Elite" user serves as a local expert.
More the number of years he is Elite more trusted Yelper

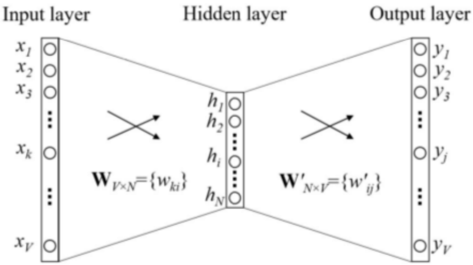
User	Elite years	nElite
1	2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015	11
2	-	0
3	2005, 2006, 2007, 2008, 2010, 2011, 2012	7

Table : *User dataset example*

Elite Dataset: Data cleaned by Python

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.098	0.902	0.902	0.905	0.910	0.901
2	W	-	-	-	0.105	0.895	0.895	0.899	0.903	0.896
3	100,50	300	-	-	0.075	0.925	0.925	0.927	0.931	0.924
4	100,50	300	0.2	-	0.081	0.919	0.919	0.922	0.928	0.916

Table : *C5.0 and DNN results on **review** dataset (III)*



Input is one hot encoded vector
Output at the Hidden layer $H^t = X^t W$
Output at the output layer $= H^t W'$
Probabilities for words in the output layer using the softmax function:

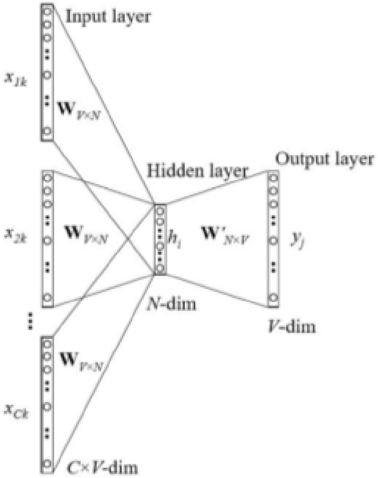
$$P(word_k \mid word_{context}) = \frac{\exp(activation(k))}{\sum_{n=1}^{V} \exp(activation(k))}$$

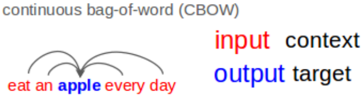
Errors calculated and the weights W and W' updated using BP

Word2vec results on review data

```
model.most_similar("food")
Out[118]:
[(u'produce', 0.6981618404388428),
 (u'meat', 0.6864877343177795),
 (u'restaurant', 0.637712836265564),
 (u'groceries', 0.6293750405311584),
 (u'coffee', 0.615966260433197),
 (u'foods', 0.6132826805114746),
 (u'yogurt', 0.6127834320068359),
 (u'seafood', 0.6104639768600464),
 (u'sushi', 0.5950002670288086),
 (u'pizza', 0.5932236313819885)]
```

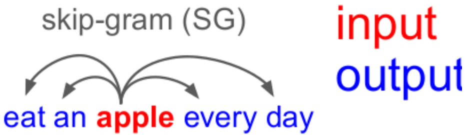
Continuous Bag of Words (CBOW)

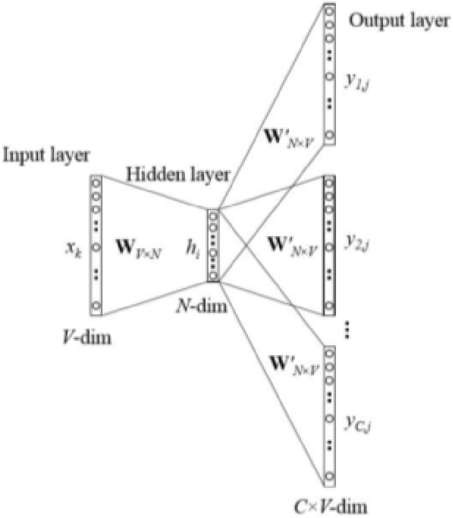




Word2vec algorithm

Skip Grams (SG)





Demo

Word2Vec features

Model	NEst	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec
1	500	-	-	0.051	0.949	0.948	0.949	0.948	0.951
2	-	100,50	100	0.053	0.947	0.947	0.948	0.951	0.945
3	-	100,50	300	0.053	0.948	0.948	0.948	0.949	0.947
4	500	-	-	0.052	0.948	0.948	0.949	0.945	0.955

Table : *Word2vec results on* ***review*** *dataset (III)*

Word2Vec features

Model	NEst	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec
1	500	-	-	0.051	0.949	0.948	0.949	0.948	0.951
2	-	100,50	100	0.053	0.947	0.947	0.948	0.951	0.945
3	-	100,50	300	0.053	0.948	0.948	0.948	0.949	0.947
4	500	-	-	0.052	0.948	0.948	0.949	0.945	0.955

Table : *Word2vec results on **review** dataset (III)*

Tip Dataset - Results I

Vector Space model from unigrams and bigrams built from tip text
In addition 3 more attributes. No missing data.
Non-Elite Dataset: Data cleaned by R

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.246	0.754	0.730	0.738	0.751	0.726
2	100,50	100	-	-	0.265	0.735	0.734	0.748	0.747	0.750
3	100,50	300	-	-	0.262	0.738	0.736	0.755	0.741	0.771
4	100,50	300	0.15	-	0.268	0.732	0.731	0.749	0.738	0.762
5	100,50	300	-	0.1	0.268	0.732	0.731	0.749	0.738	0.762
6	100,100,50	100	-	-	0.312	0.688	0.688	0.700	0.708	0.691
7	200,100	100	-	-	0.287	0.713	0.681	0.728	0.724	0.731
8	220,150	100	-	-	0.283	0.717	0.716	0.729	0.732	0.726
9	100,50,20	100	-	-	0.314	0.686	0.686	0.702	0.702	0.701
10	100,50	300	-	-	0.255	0.745	0.743	0.764	0.743	0.788

Table : *C5.0 and DNN results on **tip** dataset (I)*

Vector Space model from unigrams and bigrams built from tip text
In addition 3 more attributes. No missing data.
Non-Elite Dataset: Data cleaned by R

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.246	0.754	0.730	0.738	0.751	0.726
2	100,50	100	-	-	0.265	0.735	0.734	0.748	0.747	0.750
3	100,50	300	-	-	0.262	0.738	0.736	0.755	0.741	0.771
4	100,50	300	0.15	-	0.268	0.732	0.731	0.749	0.738	0.762
5	100,50	300	-	0.1	0.268	0.732	0.731	0.749	0.738	0.762
6	100,100,50	100	-	-	0.312	0.688	0.688	0.700	0.708	0.691
7	200,100	100	-	-	0.287	0.713	0.681	0.728	0.724	0.731
8	220,150	100	-	-	0.283	0.717	0.716	0.729	0.732	0.726
10	100,50	300	-	-	0.255	0.745	0.743	0.764	0.743	0.788

Table : *C5.0 and DNN results on **tip** dataset (I)*

Non-Elite Dataset: Data cleaned by Python

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.271	0.729	0.730	0.737	0.753	0.721
2	100,50	300	-	-	0.248	0.752	0.751	0.763	0.764	0.762
3	100,50	300	-	-	0.251	0.749	0.748	0.765	0.756	0.774

Table : *`C5.0` and `DNN` results on **tip** dataset (II)*

Non-Elite Dataset: Data cleaned by Python

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.271	0.729	0.730	0.737	0.753	0.721
2	100,50	300	-	-	0.248	0.752	0.751	0.763	0.764	0.762
3	100,50	300	-	-	0.251	0.749	0.748	0.765	0.756	0.774

Table : *C5.0 and DNN results on **tip** dataset (II)*

Elite Dataset: Data cleaned by Python

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.274	0.726	0.727	0.732	0.753	0.712
2	W	-	-	-	0.275	0.725	0.725	0.734	0.747	0.721
3	100,50	300	-	-	0.261	0.739	0.738	0.756	0.744	0.769
4	100,50	300	0.2	-	0.271	0.729	0.728	0.746	0.736	0.756

Table : *`C5.0` and `DNN` results on **tip** dataset (III)*

Tip Dataset - Results III

Elite Dataset: Data cleaned by Python

Model	Neurons	Epoch	vd	hd	HLoss	Acc	AUC	F1	Prec	Rec
1	-	-	-	-	0.274	0.726	0.727	0.732	0.753	0.712
2	W	-	-	-	0.275	0.725	0.725	0.734	0.747	0.721
3	100,50	300	-	-	0.261	0.739	0.738	0.756	0.744	0.769
4	100,50	300	0.2	-	0.271	0.729	0.728	0.746	0.736	0.756

Table : *`C5.0` and `DNN` results on **tip** dataset (III)*

Image Dataset

37,921 images were filtered belonging to “Food” and “Shopping” business categories
Only 1 image per unique business $- >$ 9,026. The distribution of these images in the two categories is as follows:

- 4,642 - Shopping
- 4,707 - Food
- 323 - Both

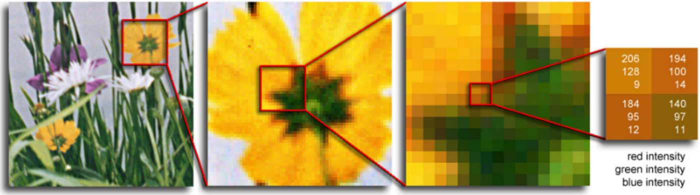
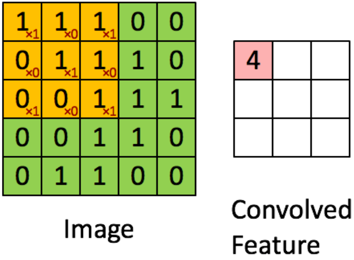
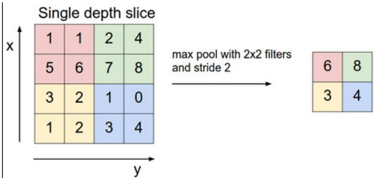
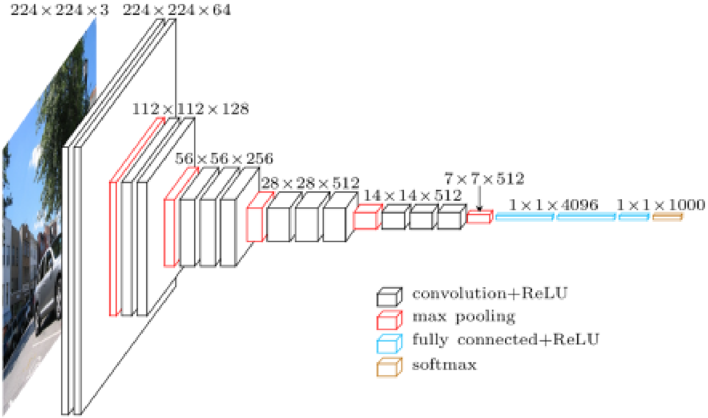


Image was resized to 224 $\times$ 224







Input to this model is a 224 $\times$ 224 fixed size colour image
4096 convolved features which can be passed to any classifier
Classified bye 2-hidden layer DNN with 4096 and 1000 neurons in each layer resp.

1. No normalization of RGB components
2. Normalization of RGB components (123.68, 116.779, 103.939)

Model	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec
1	4096,1000	100	0.247	0.753	0.753	0.763	0.745	0.781
2	4096,1000	100	0.219	0.781	0.781	0.790	0.770	0.811

Table : *DNN results on **image** dataset*

Outline

7. Ensemble Classifier

Ensemble Classifier

- The best models from business, review, tip and image models were used to build an Ensemble.
- Separate training set of 13,500 businesses with their respective 59,174 reviews, 30,499 tips and 6,946 images.
- A test set of 2,080 examples was constructed containing atleast business information and an image, review and tip were optional.
- The model is based on the assumption that the data belonged to either one or both the categories. In case, if a classifier gave negative results for both the categories for the testing set then, the label with highest frequency in training was directly assigned to that example.
- Ensemble model contained mandatorily business ("Bus"), review ("Rev") and image models. Tip ("Tip") model and word2vec review model ("w2v") was checked if it improved the overall performance. Image model was checked to see if Plain ("Img") or Normalized ("ImgN") image was giving better performance. In the end, the models that gained higher measures were kept and others discarded from the Ensemble model.

Ensemble Classifier

Data	Model	NEst	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec	SubAcc
Training											
Bus	DNN	-	100,50	300	0.085	0.915	0.915	0.918	0.922	0.914	0.902
Rev	DNN	-	100,50	300	0.074	0.926	0.927	0.929	0.937	0.927	0.905
w2v	RF	500	-	-	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Tip	DNN	-	100,50	300	0.271	0.729	0.728	0.747	0.743	0.745	0.675
Img	DNN	-	4096,1000	100	0.000	1.000	1.000	1.000	1.000	1.000	1.000
ImgN	DNN	-	4096,1000	100	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Testing											
Bus	DNN	-	100,50	300	0.155	0.845	0.845	0.847	0.849	0.843	0.830
Rev	DNN	-	100,50	300	0.068	0.932	0.932	0.933	0.935	0.930	0.919
w2v	RF	500	-	-	0.050	0.950	0.950	0.951	0.949	0.954	0.936
Tip	DNN	-	100,50	300	0.328	0.672	0.672	0.681	0.673	0.690	0.642
Img	DNN	-	4096,1000	100	0.247	0.753	0.753	0.763	0.745	0.781	0.710
ImgN	DNN	-	4096,1000	100	0.219	0.781	0.781	0.790	0.770	0.811	0.735
Ensemble											
1. Tips, No word2vec, Plain Image					0.095	0.906	0.907	0.901	0.960	0.850	0.848
2. No tips, No word2vec, Plain Image					0.072	0.928	0.928	0.929	0.925	0.934	0.908
3. No tips, No word2vec, Normalized Image					0.111	0.889	0.890	0.880	0.979	0.799	0.801
4. No tips, Word2vec, Plain Image					0.054	0.946	0.947	0.945	0.976	0.916	0.916
5. No tips, Word2vec, Normalized Image					0.044	0.958	0.958	0.959	0.961	0.956	0.948

Table : *Ensemble results*

Ensemble Classifier

Data	Model	NEst	Neurons	Epoch	HLoss	Acc	AUC	F1	Prec	Rec	SubAcc
Training											
Bus	DNN	-	100,50	300	0.085	0.915	0.915	0.918	0.922	0.914	0.902
Rev	DNN	-	100,50	300	0.074	0.926	0.927	0.929	0.937	0.927	0.905
w2v	RF	500	-	-	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Tip	DNN	-	100,50	300	0.271	0.729	0.728	0.747	0.743	0.745	0.675
Img	DNN	-	4096,1000	100	0.000	1.000	1.000	1.000	1.000	1.000	1.000
ImgN	DNN	-	4096,1000	100	0.000	1.000	1.000	1.000	1.000	1.000	1.000
Testing											
Bus	DNN	-	100,50	300	0.155	0.845	0.845	0.847	0.849	0.843	0.830
Rev	DNN	-	100,50	300	0.068	0.932	0.932	0.933	0.935	0.930	0.919
w2v	RF	500	-	-	0.050	0.950	0.950	0.951	0.949	0.954	0.936
Tip	DNN	-	100,50	300	0.328	0.672	0.672	0.681	0.673	0.690	0.642
Img	DNN	-	4096,1000	100	0.247	0.753	0.753	0.763	0.745	0.781	0.710
ImgN	DNN	-	4096,1000	100	0.219	0.781	0.781	0.790	0.770	0.811	0.735
Ensemble											
1. Tips, No word2vec, Plain Image					0.095	0.906	0.907	0.901	0.960	0.850	0.848
2. No tips, No word2vec, Plain Image					0.072	0.928	0.928	0.929	0.925	0.934	0.908
3. No tips, No word2vec, Normalized Image					0.111	0.889	0.890	0.880	0.979	0.799	0.801
5. No tips, Word2vec, Normalized Image					0.044	0.958	0.958	0.959	0.961	0.956	0.948

Table : *Ensemble results*

Outline

8. Conclusions & Future Work
   - Conclusions
   - Future Work

- Reviews are the most indicative
- Good quality reviews by Elite users
- Data cleanup, missing value imputation prove very useful
- Given more infrastructure image model can be developed further

- Aggregate the Tips to make a review
- Translation of foreign language reviews in English script
  Raw review: De las tiendas más bonitas que puedes encontrar en Edimburgo. Te quieres comprar todo lo que ves. Si te gusta el diseÃśo, la ilustraciÃşnes y las cosas bonitas este es tu sitio.... Tienen ilustraciónes para decorar la casa, accesorios, bolsos, chorradas varias, etc. Es una tienda en la que curiosear y dejarse los ahorros del mes. Sin duda alguna si tienes que hacer un regalo a alguien, quieres decorar tu casa o simplemente eres una loca de estas cosas como yo pues este es tu sitio
  English translation: Of the most beautiful shops you can find in Edinburgh. You want to buy everything you see. If you like the design, artwork and beautiful things this is your place .... Have pictures to decorate the house, accessories, handbags, chorradas groups, etc. It is a shop where you look around and let the savings of the month. No doubt if you have to make a gift to someone, you want to decorate your home or you're just crazy about these things as I do because this is your place
- Adding more categories

References

- Reviews that mention posh in London, <https://www.yelp.co.uk/wordmap/london/posh>
- Crain, K., Heh, K., and Winston, J. (2014). An Analysis of the Elite Users on Yelp.com, Stanford university project report, USA
- Sorower, M. S. (2010). A literature survey on algorithms for multi-label learning. Technical report.
- Zhang, M. and Zhou, Z. (2013). A review on multi-label learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, PP(99):1.
- Clare, A. and King, R. D. (2001). *Knowledge Discovery in Multi-label Phenotype Data*, pages 42–53. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Rong, X. (2014). Word2vec Parameter learning explained. *CoRR*, abs/1411.2738.
- Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Thank You!!