Topic Based Sentiment Summarizer for Yelp Review

Junfeng Wu
wujunfen@usc.edu

Junming Chen
junmingc@usc.edu

Yi Ding
dingyi@usc.edu

Abstract

1 Introduction

In this project, we summarized 100 restautants with most reviews in Yelp Challenge dataset based on topic, then use a QA like interface to show our results.

1.1 Motivation

There are many disorganized and redundant reviews for each business, so it is inconvenient for users to find specific reviews for some aspects they really want to know. Therefore, our task in this project is to summarize reviews to a few aspects and make sentiment analysis for each aspect. The system will help users to find useful and valuable reviews based on their demands. It is interesting because the system could provide an efficient way to utilize the review information.

Users who want to get useful information about the business in Yelp reviews could benefit from the system we built. Users could type in the question they want to ask. Then they could get summarized reviews for each aspect of the restaurant they referred in the question.

1.2 Challenges

Reviews on Yelp contain both useful and useless information. There may be many inappropriate terms occurred in reviews. Finding useful information among the reviews is time consuming currently.

1.3 Related Work

There are many exsiting aproaches on mining and summerize customer reviews. In Hu. and Liu.ś approach(Hu and Liu, 2004a), they extract product features from the customer review and identify sentiment of the dispcription of the features. And they use these features to summarize the product reviews. In the approach of Sasha Blair-Goldensohn et al.(Blair-goldensohn et al., 2008) They classify review sentences based on topic and sentiment using supervised training and summarized the reviews based on sentences extraction. They use a lexicon approach to identify sentiment and use binary classifeir on labeld data to classify review sentences into topics. The summariation is based on the polarity score of sentiment analysis, sentences with higher polarity score will be chosen first.

The framework of our approach is based on the work of Sasha Blair-Goldensohn et al.(Blair-goldensohn et al., 2008) , but we tried different approaches in each step. We first use LDA to generate 50 topic from the review data, then annotate them into 4 aspects users usually care about: food, service, atmosphere, value. In the second step we do the sentiment classification for each review and test the classification in 5 ways: baseline classification, Naive Bayes, sentiment score, Word vector and combining Naive Bayes and Word vector. After this, we use KL divergence value to extract representative 10 sentences for each aspect of each restaurant. The final step is to build a QA like interface to show our result.

2 Topic Extraction

In this section we describe the component in our system that identifies the topics of a restaurant that users typically rate. This includes finding corresponding sentences that mention these aspects. We tested two approaches. The first one is to extract nouns and adjectives in the sentence to identify the topic and the second one is to use LDA model to generate topics and classify sentences into topics based on the probabilty distribution. And we finally choose to use LDA model to generate and classify topics.

2.1 Use Frequently Shown NNs

The first try to do the summarization of review is extracting frequently described features of business and use those features to locate valuable sentences.(Hu and Liu, 2004b)(Hu and Liu, 2004a) After running small demo we found out that different kind of business vary in frequent features, so the most frequent feature among all business may not reflect the characteristics of specific business. Thus we reduce the problem to research on the top 100 restaurants that get most reviews. And we analyze NN frequency for each restaurant that can help us get some unique features like a popular dish name. In this approach, we can get collection of sentences describing a given feature. To improve, we also use dependency tree locate the adjective word of feature, and then drop the sentences using same word to describe same feature. One step further, after the sentiment score calculation, we can further drop sentences with similar adjective words.

Feature	Sentences
Food	1.The food did take a little while to come out. 2.I may be a bit too enthusiastic about Gallo Blanc ... food and reasonable prices. ...
Torta	1.I was excited to actually have a true torta bun... 2.The torta was delicious ... ...
Service	1.I have no complaints about the service.

Table 1: Result sentences of Gallo Blanc restaurant extracted using frequently showed NNs

2.2 Topic Classification Using LDA

Latent Dirichlet Allocation: Latent Dirichlet Allocation (LDA) (Blei et al., 2003) is a generative probabilistic model for text corpora. It is used as a topic model to discover the underlying topics that are covered by a text document. LDA assumes that a corpus of text documents cover a collection of K topics. Each topic is defined as a multinomial distribution over a word dictionary with $|V|$ words drawn from a Dirichlet $\beta_k \sim Dirichlet(\eta)$.

Each document from this corpus is treated as a bag of words, and is assumed to be generated by first picking a topic multinomial distribution for the document $\theta_d \sim Dirichlet(\alpha)$. Then each word is assigned a topic via the distribution θ_d, and then from that topic k, a word is sampled from the distribution β_k. θ_d for each document can be thought of as a percentage breakdown of the topics covered by the document.

With the LDA model, Blei et al. present an Expecatation Maximization algorithm that converges to the most likely parameters (word distributions per topic and topic distributions per word).

In the topic classification stage, we predict the topics of a sentence based on its topic probablity distribution with the trained LDA model.

Implementation: In the project, we use the Python library Gensim to train the LDA model, base on the work from J Huang et al. (James Huang, 2013), we tried topic models for our text corpus for a range of topic numbers $K \in [10, 500]$ and for $|V| \in [10000, 20000]$. Stopwords are removed to reduce noise. And we selet $K = 50$ and $|V| = 10000$ to generate the topic model. Table 2 show some examples of generated topics. After observing the generated topics, we mannually map these 50 topics into 4 categories and a special label: Food, Service, Value, Atmosphere, None, in which None means the topic is too general to be classified into any of the other four categories. To use the model to predict topics of sentences, we set the probability threshold to 20% to eliminate sentences which do not have much meaning. Detailed evaluation will be discussed in Subsection 6.2

Service	service + table + order +... + came
Food	cream + coffee + ice + ... + pancakes
Value	price + worth + money + ... + tip
Atm.	outdoor + drive + ... + decor

Table 2: Topic examples

3 Sentiment Classification

After the review sentences are classfied into four topics, the next stage is to classify each sentence as being positive, negative under each topic. Yelp dataset has provided stars for each review, which could indicate sentiment of each review. However, the stars label cannot directly used to identify the sentiment of each sentence because a 5-star review could still contain negative sentences. In this sec-

tion, we provided three approaches and tried to combine them to get better result.

3.1 Sentiment Score

Sentiment score(Liu and Seneff, 2009) is a way to determine the degree of sentiment. Like the word 'excellent' and 'good' express different degree of the positive sentiment. Usually, only adjectives and their negation are considered into the calculation of sentiment score, but adverbs are also a very important factor reflect it. For example, "not very good" seems to be less negative than "not good".

For adjectives:

$$Score(adj) = \frac{\sum_{i \in p} \frac{N}{n_{r_i}} \times r_i}{\sum_{r_i} \frac{N}{n_{r_i}}} \quad (1)$$

To illustrate, P represents the set of appearances of adjective adj, r_i represents the user rating in the appearance, N represents review number and nr_i represents the number of review with rating r_i.

For adverbs:

$$Score(adv) = \sum_{t \in A} \frac{count(adv, adj_t)}{\sum_{j \in A} count(adv, adj_j)} \times Pol(adj_t) \times (r(adv, adj_t) - r(adj_t)) \quad (2)$$

For negative phrase:

$$Score(neg(adv(adj))) = r(adj) + Pol(adj) \times r(adv) + Pol(adj) \times r(neg) \quad (3)$$

where count(adv,adj_t) represents the count of appearance of pair adv-adj_t, A represents the set of adjectives that co-occur with adv, r(adv,adj_t) represents the sentiment rating of pair adv-adj_t , and r(adj_t) represents the sentiment rating of the adjective adj_t calculated by last formula. Pol(adj_t) is the polarity of adj_t , it is assigned 1 if r(adj_t) $\geq$4, -1 if r(adj_t) $\leq$ 2, and 0 for the value between 2 to 3 in my implementation. For example, given the ratings <good: 4.5>, <bad: 1.5>, <very: 0.5> and <not: -3.0>, we would assign "5.0" to "very good" (score(very(good))=4.5+0.5), "1.0" to "very bad" (score(very(bad))=1.5-0.5), and "2.0" to "not very good" (score(not(very(good)))= 4.5+0.5- 3.0).

We tried the same method on 5% data that's about 5000 long reviews. However, the result is not good(Tabel 3). The score of adjectives are abnormal like 'delicious' is same as 'bad' with score 3. This problem is inevitable because there are several short sentences to compose a whole review and even if the review has score 5 it can have many negative sentences. Then due to the low reliability of adjs' scores, the score of advs are awful. Thus the final test on sentiment classification only use the adjs' scores and the negation score of -3 as the paper does.

ADJ	score	ADV	score
appreciative	5	beautifully	0.875
moderate	4	truly	0.2045
delicious	3.33	unacceptably	0.0
bad	3	very	-0.0081
lousy	1	not	-0.1011

Table 3: Part of sentiment scores

To do sentiment classification, the sentence first parsed to extract all words with part-of speech tag 'JJ'. Then check if there are negations of them and form adjective predicates. For each sentence, its sentiment score is the average of the adjective predicate score. Finally we define the sentence with score over 3 point as positive review, the rest as negative review.

3.2 Naive Bayes Classifier

Motivation: The first homework has shown that Naive Bayes Classifier has a decent performance on sentiment analysis. The diffiulty in this project is that we could only train a Naive Bayes Classifier based on the sentiment label of reviews instead of sentence. However, observation shows that even though a positive review may contain negative sentences, The portion of positive sentences in postive reviews is still greater. Based on this observation, we implement a Naive Bayes Classifier to try to improve sentiment classification on review sentences.

Implementation: After several experiment, we decided to use 2-gram and 3-gram if the review text as our feature.

$$P(d|c) = P(t_1|c)P(t_2|c)...P(t_L|c)$$

where t_l is one n-gram term in the document and c is the label. We simply use the term frequency of t_l in class c as an estimation of $P(t_l|c)$

We use this Naive Bayes Classifier trained on sentiment of reviews to predict the sentiment of

sentences. The result is slightly better than baseline as shown in Table 4). Details of evaluation will be illstrated in Subsection 6.3

Table 4: Naive Bayes Classifier Result.

Table A: Positive Label

Positive	Precision	Recall	F-score
Baseline	0.89	0.81	0.85
Naive Bayes	0.91	0.82	0.86

Table B: Negative Label

Negative	Precision	Recall	F-score
Baseline	0.57	0.73	0.64
Naive Bayes	0.57	0.76	0.65

3.3 Approaches with Word Vector

In this approach, we try to train a model with some semantic understanding of words using distributed word vectors.

Vector from words to sentences: Since the reviews are in variable length, we need to find a way to take individual word vectors and transform them into a feature set that is the same length for every review. We tried two approaches here (Kag, 2014)

1. **Vector Averaging**

   Since each word is a vector in fixed space, we can use vector operations to combine the words in each sentence. So we simply average the word vectors in a given reivew.

2. **Clustering**

   We use K-Means clustering to generate word clusters based on their distributed vectors. After the clustering, we have a cluster assignment for each word. And we use the frequency of semantically related clusters instead of individual words as feature. Since the number of clusters is fixed at K, the feature vector now in in fixed size. Here, K is set as $N/5$ where N is the size of vocabulary because cluster with about 5 words tends to performa better.

Like the Naive Bayes Classifier, we train two classifiers of both of these two features using review data and their sentiment label and predict sentence sentiment using these classifiers.

In the implementation, we use the word2vec module provided in Gensim Python library to train the word vector whole review sentences of restaurants in Yelp dataset. Each word is a vector in 100-dimensional space. We use random forest classfier to train the model on reviews and test it on the sentences dataset. The result is shown in Table 5. Details of the evaluation will be discussed in Subsection 6.3. We found that these aproaches produced results much better than chance, but underperformed baseline. In next section, we try to use these classfiers to improve the performance of Naive Bayes Classifier.

Table 5: Word Vector Classifiers Result.

Table A: Positive Label

Positive	Precision	Recall	F-score
Baseline	0.89	0.81	0.85
Vector Averaging feature	0.80	0.84	0.82
Clustering feature	0.78	0.84	0.85

Table B: Negative Label

Negative	Precision	Recall	F-score
Baseline	0.57	0.73	0.64
Vector Averaging feature	0.45	0.39	0.42
Clustering feature	0.53	0.45	0.49

3.4 Adaptation with Classfier Diversity

The difficulty of the sentiment analysis is that we are using the classifier trained on review data to predict the sentiment of sentence data. So we are inspired by the Domain Adaptation with Parser Diversity (Sagae, 2007) to try to use the Classfier Diversity to improve the result. Our approach is as follow:

1. We trained the Naive Bayes Classifier and Word Vector Classifiers with both vector averaging feature and clustering feature using the labeled review data;

2. We compared the output for the three models, and selected only identical analyses that were produced by each of the two separate models;

3. We added those analyses (about 200k words in the test domain) to the original (out-ofdomain) labeled training set;

4. We retrained these two classifiers using the new training set and used the new model to predict sentiment label on sentence test set.s

The result in Table 6 shows that after the adaption, the Naive Bayes Classifier has slight improvement and the Word Vector Classifier has significant improvement.

Table 6: Result After Adaption(Word Vector using Clustering feature has not been tested yet)

Table A: Positive Label

Positive	Precision	Recall	F-score
Baseline	0.89	0.81	0.85
Naive Bayes	0.91	0.85	0.88
Word Vector (Vector Averaging feature)	0.83	0.89	0.86

Table B: Negative Label

Negative	Precision	Recall	F-score
Baseline	0.57	0.73	0.64
Naive Bayes	0.63	0.75	0.69
Word Vector (Vector Averaging feature)	0.59	0.46	0.52

4 Summarization

Using input prepared from the modules for sentiment classification and aspect extraction, the next stage in our system is the summarizer module. The object of summarizer is to extract concise samplings of the original review texts in order to provide a high-level overview based on topic and sentiment. In this project, we implement KLSum algorithm (Haghighi and Vanderwende, 2009) to generate summarized text.

4.1 KLSum

The KLSum algorithm provides a criterion for selecting a summary S given a document collection D, this criterion is

$$S^* = \min_{S:words(S)\leq L} KL(P_D||P_S) \qquad (4)$$

here P_S is the empirical unigram distribution of the candidate summary S and $KL(P||Q)$ represents the Kullback-Lieber (KL) divergence between P and Q:

$$KL(P||Q) = \sum_w P(w) \log \frac{P(w)}{Q(w)} \qquad (5)$$

This criterion casts summarization as finding a set of summary sentences which closely match the document set unigram distribution.

4.2 Implementation

We implement a greedy algorithm based on the KL divergence criterion in Subsection 4.1. We use the first ten sentences selected by the greedy algorithm as the summarized text. The approach could be describes as:

While number of sentences is less than 10,

1. Select the senctence from the document collection that could best improve $KL(P||Q)$ between the document collection and summarized text.

2. Remove the selected sentence from the document collection and add it into the summarized text.

3. Update the term frequency distribution of the summarized text.

Stopwords are ignored in the implementation because unigram distribution of stopwords is less likely to be substantially different between the summary and the input, removing them would highlight the meaningful words in the distribution.

Table 17 shows a result of the generated summarized text. The detail of evaluation will be discussed in Subsection 6.4

5 Question Answing Interface

In this section we describe the methods we used to implement the question answering interface which provides a friendly interface for users to ask questions and a demonstration of the result we get in previous tasks.

5.1 Name Entity Recognition

We tried to use off-the-shelf Named Entity Recognition tool to recognize restaurant name at the beginning, but the result was not so good. So we finally decided to use Stanford Named Entity Recognizer (NER), also known as CRFClassifier, which provides a general implementation of (arbitrary order) linear chain Conditional Random Field (CRF) sequence models. That is, we could train our own NER model to recognize restaurant

Food	Positive	We got the Colorado lamb chops (just okay), grilled french sea,bass (delicious), steamed Scottish salmon (good)...
	Negative	Well, too much cheese and too much pepperoni and not enough,time in the oven...
Service	Positive	All I can say is the service was fast, food came quick and everything worked out great...
	Negative	We had to wait 40 min just to get the food and the order was wrong...
Value	Positive	Fusion tacos for about $1 and fusion burritos for about $4,cheap and nice size portions...
	Negative	The basket is so small you'd think they should bring 1 for,each person or at least give us a much bigger bowl of chips...
Atm.	Positive	Beautiful outdoor picnic area type seating surrounded by large,mature trees...
	Negative	...there was only one condiment and drink area...

Table 7: Summarization example

name only. As for the training data, we gathered frequently referred 100 restaurant names and 43 common sentence patterns to generate question sentences, and then tagged each word in the form of CRFClassifierś requirements to produce training data.

5.2 Parse Analysis

After correctly recognize restaurant in the former step, we parse the sentence again with original restaurant name replaced by restaurant concatenated with underscore '_'.(Hirschman and Gaizauskas, 2001)(Harabagiu et al., 2000) In this way, the parser can recognize the restaurant name as a whole and parse appropriately. Then look through the dependency tree of sentence to decide what feature of restaurant and what sentiment aspect of the feature does the question focus on. For example, in question 'Does XX restaurant provide delicious food?', we need to give answer about XX restaurant, the focused aspect is food.

The query is formed in this way:
{'Restaurant Name', 'Aspect'}

5.3 Extract answer

The first step is to locate summarized review match 'Restaurant Name'. Then if user specifies the aspect, return the related aspect summary. Otherwise we return summary of all the aspect. What's more, LDA classification can be used to predict the question aspect. If the prediction is same as user input, results of that aspect are returned. If the prediction is not exactly the same as user's input, all related aspects according to both user input and LDA classification result are returned.

For example, in question 'IS XX's food expensive?', 'food' is the exact aspect input by user, but aspect 'value' can be recognized by LDA classification so both 'food' and 'value' would be returned.

6 Evaluation

6.1 Experiment Data

We use the Yelp Challenge Dataset as our experiment dataset. We extract the reviews of the top 100 restaurants in order of the number of reviews as training data. Details of the data set is shown in Table 10. We randmonly pick 100 reviews which contain 921 sentences as our test data on Topic Classification and Sentiment Analysis

Number of Restaurant	100
Number of Reviews	114034

Table 8: Overview of Dataset

Star(s)	Number	Percentage
1	5853	5.1%
2	9182	8.1%
3	16738	14.7%
4	37894	33.2%
5	44367	38.9%

Table 9: Review rating distribution of 100 restaurants.

Star(s)	Number	Percentage
1	2	2%
2	12	12%
3	13	13%
4	32	32%
5	41	41%

Table 10: Review rating distribution of 100 reviews in test set.

6.2 Topic Classification

Table 11 shows the topic distribution in the test data, note that

Label	Number	Percentage
Food	521	56.6%
Service	165	17.9%
Value	90	9.8%
Atm.	72	7.8%
Unknown	101	11.0%

Table 11: Sentiment distribution in data set

Table 12 shows the result of the Topic Classification, we use the result of Sasha Blair-Goldensohn et al(Blair-goldensohn et al., 2008). to compare with our result. Their approach is supervised training of a binary maximum entropy classifier for every topics on labelled data. The result shows that our unsupervised LDA method still underperformed their approach.

Table 12: Review rating distribution of 100 restaurants.

Table A: Topic Classification Result using LDA in our project

	Precision	Recall	F-score
Food	0.87	0.62	0.72
Service	0.48	0.72	0.58
Value	0.73	0.29	0.42
Environment	0.46	0.50	0.48

Table B: Topic Classification Result of Sasha Blair-Goldensohn et al.

	Precision	Recall	F-score
Food	0.84	0.82	0.83
Service	0.87	0.67	0.76
Value	0.90	0.56	0.69
Environment	0.71	0.47	0.57

6.3 Sentiment Analysis

Table 13 shows the sentiment distribution in data set, note that some sentence do not show any sentiment and in this case currently we just don't take such situation into account.

Label	Number	Percentage
POS	612	66.4%
NEG	208	22.6%
Unknown	101	11.0%

Table 13: Sentiment distribution in data set

Baseline Setting: To setup the baseline, we simply classify a sentence as Positive if it belongs to a 4-star or 5-star reivew. Otherwise, we classify this sentence as negative.

Result Comparison: Table 1 shows the result comparison of each approach we use and the baseline.

Positive	Precision	Recall	F-score
Word Vector	0.80	0.84	0.82
Word Vector(after adaption)	0.83	0.89	0.86
Nave Bayes	0.91	0.81	0.86
Nave Bayes(after adaption)	0.89	0.86	0.88
Baseline	0.89	0.81	0.85
Sentiment Score	0.92	0.65	0.76

Table 14: Sentiment Result(Positive Label)

6.4 Summarization

Test Data: We annotate the summarized text on the 10 restaurants. The annotation rule is that if the sentences is picked wrong, we give it score 0, if the sentence's topic and sentiment is right, but not suitable for a summarized text, we give it score 1, if the sentence is right in topic and sentiment and suitible for a summarization, we give it score 2. The standard of whether a sentence is good for summerization is decided by a user after browsing the reviews of these 10 restaurants. The rules is described in Table 16

Table 17 shows the final result of summarization.

Negative	Precision	Recall	F-score
Word Vector	0.45	0.39	0.42
Word Vector(after adaption)	0.59	0.46	0.52
Nave Bayes	0.57	0.76	0.65
Nave Bayes(after adaption)	0.63	0.70	0.67
Baseline	0.57	0.73	0.64
Sentiment Score	0.23	0.64	0.34

Table 15: Sentiment Result(Negative Label)

Score	Description
0	Completely wrong
1	Right in topic and sentiment but not suitable for summarization
2	Right and generalized enough

Table 16: Summarization evaluation rule

6.5 Question Answering

From the user questions and our own designed questions, we extract 43 question patterns and generate a question test file with 4300 questions that each restaurant name was asked the 43 questions.

Name Entity Recognition: For all of the questions, the restaurant name is recognized correctly, so the f-score of NER step is 1. Our self-built NER model also handle some odd name in the dataset like 'eat.','Lo-Lo's Chicken & Waffles', 'Todd English P.U.B.' and so on.

Aspect Recognize For the aspect recognize part we annotated what the 43 question focused on. Then calculate the recognition f-score on the 4300 generated questions for each aspect.(Aspect recognition results have little variety on same question with different restaurant name)

The result is reasonable because it is similar to the LDA classification result. The reason why the fscore for atmosphere and value is low is that most of our question collected is about food and service. Another reason is that for those general question which should return all aspects' results, it is often that the question is classified into one specific aspect so the other aspect cannot be returned. In conclusion, our QA interface shows decent result for user during the test.

Score	Count	Percentage
0	135	33.75%
1	63	15.75%
2	202	50.50%

Table 17: Summarization evaluation result

	food	service	atm.	value
precision	0.73	0.81	0.37	0.40
recall	0.84	0.58	0.52	0.46
f-score	0.78	0.67	0.43	0.43

Table 18: F-score of Aspect Recognize

7 Future Work

Topic Classification: The result of our LDA classification is not good in some aspects. It may be improved by put LDA classification as a feature into the training set and annotate some data to do supervised classification.

Sentiment Analysis: There re some more sophiscated approaches of sentiment analysis using word vector(Maas et al., 2011)(Socher et al.,), which take other factors like sentence structure, term probablity distribuion into consideration. In the future, we could try to apply these approaches into our system.

Summarization: The summarization using KL divergence is inclined to choose long sentences. In the result, we can see most of sentences contains too much noisy information. We can try to split the long sentences into shorter sentences.

QA interface: Our aim at first is trying to give some recommendation answers for users like answering yes/no for "Is In N Out good?". However, because of the bad result of sentiment score and the lack of relation between such answer and our summary of reviews, we only kept the summary answer in QA part. Our query actually contains adjective of restaurant and aspect, so what we can try in future is to use annotated sentiment word in Wordnet or use Sentiwordnet directly to generate degree of sentiment of sentence, and then give some recommendation for users' questions.

Work Distribution

- **Junfeng Wu** : Subsection 2.2, Subsection 3.3, Subsection 3.4, Section 4

- **Junming Chen** : Subsection 2.1, Section 3.1, Section 5.2, Section 5.3, Section 6.3, Section 6.5
- **Ding Yi** : Section 1, Subsection 3.2, Subsection 5.1, Section 6

Our code has been put on a bitbucket repository, available at `https://bitbucket.org/wjf3121/csci544-project`.

References

[Blair-goldensohn et al.2008] Sasha Blair-goldensohn, Tyler Neylon, Kerry Hannan, George A. Reis, Ryan Mcdonald, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *In NLP in the Information Explosion Era*.

[Blei et al.2003] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

[Haghighi and Vanderwende2009] Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of HLT-NAACL 2009*. Association for Computational Linguistics.

[Harabagiu et al.2000] Sanda Harabagiu, , A M. Harabagiu, Marius A. Pa ca, and Steven J. Maiorano. 2000. Experiments with open-domain textual question answering. pages 292–298.

[Hirschman and Gaizauskas2001] L. Hirschman and R. Gaizauskas. 2001. Natural language question answering: The view from here. *Nat. Lang. Eng.*, 7(4):275–300, December.

[Hu and Liu2004a] Minqing Hu and Bing Liu. 2004a. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. ACM.

[Hu and Liu2004b] Minqing Hu and Bing Liu. 2004b. Mining opinion features in customer reviews. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, pages 755–760. AAAI Press.

[James Huang2013] Eunkwang Joo James Huang, Stephanie Rogers. 2013. Improving restaurants by extracting subtopics from yelp reviews.

[Kag2014] 2014. Bag of words meets bags of popcorn.

[Liu and Seneff2009] Jingjing Liu and Stephanie Seneff. 2009. Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 161–169, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Maas et al.2011] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Sagae2007] Kenji Sagae. 2007. Dependency parsing and domain adaptation with lr models and parser ensembles. In *In Proceedings of the Eleventh Conference on Computational Natural Language Learning*.

[Socher et al.] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank.