Customer Feedback Topic Modelling Using Online Latent Dirichlet Allocation

Yuhou Zhou
Department of Computer Science & Electrical Engineering
Jacobs University Bremen
Bremen, Germany
Corporate Information Management
CEWE Stiftung & Co. KGaA
Oldenburg, Germany
yu.zhou@jacobs-university.de
yuhou.zhou@cewe.de

Abstract

I develope a topic modelling system[1] for extracting topics from customer feedback. The system includes three parts which are preprocessing, modelling, and visualization. Apache PySpark helps to read large volume data, and Python spaCy is used to run preprocessing pipeline. Online Latent Dirichlet Allocation is the selected topic model, and the system uses Gensim implementation. Topic coherence serves as the evaluation metric for choosing the optimal topic number. LDAvis presents topic modelling results interactively and shows the topic distribution over two principal components. This project can serve as a base for further generalizing to extract topics from different data sources.

1 Introduction

Customer feedback is information provided by clients about their general experience with a product or a service. At CEWE, one method to collect such feedback is asking customers to complete online survey.

By collecting customer feedback, the extracted information can act as a guide to improve products. Feedback gives the company the clue of customers' general experience and the satisfaction rate. Except the description provided by sellers, the publicly posted feedback is useful information to other consumers, imparting them an overview how the product really is. This information also helps managers to make correct business decisions.

The current way of processing user feedback requires that working staff reads the feedback and later extracts topics. This process can be costly and inefficient, because of manual intervention. At one time, people can only process a small batch of text from a recent period. Thus, the information, which concerning long lasting but less frequently appeared issues, may lost.

Latent Dirichlet Allocation (LDA) can automate this topic extraction process. LDA is a generative probabilistic model for discrete data, and it is especially popular in the field of topic modelling. In this project, I automate the topic extraction process utilizing Online Latent Dirichlet Allocation with the optimal number of topics found by topic coherence.

[1] Available on *gitlab.com/yuhouzhou/cftm*

2 Related Work

Discovering "topics" from a collection of documents has been a long-standing problem in natural language understanding. Latent Semantic Analysis (LSA) acts as the base of recent topic models [1]. Hofmann proposed Probabilistic Latent Semantic Indexing (pLSA) [2], bringing topic models from deterministic to probabilistic. Blei et al. (2003) described Latent Dirichlet Allocation (LDA) which is a Bayesian alternative to pLSA [3]. LDA makes use of word co-occurrence information in documents, to generate document-topic distribution. Hoffman et al. (2010) developed an online variational Bayes algorithm for LDA [4]. This online version of LDA was demonstrated that it has advantages over batch LDA in terms of model accuracy and the modelling time. However, the lack of word co-occurrence in short texts prevents LDA yielding satisfying results [5]. Such short texts are news headlines, tweets, user feedback, questions and answers, and so on. One strategy to overcome this data sparsity problem is to concatenate a set of short texts according to available metadata [6, 7, 8]. For example, aggregating texts by timestamps, locations or users.

Perplexity was used to evaluate the accuracy of topic models [9]. However, Chang et al (2009) found that perplexity does not always correlated with the interpretability of topics [10]. Topic coherence c_uci was first proposed by Newman et al.(2010)[11], by using pairwise pointwise mutual information (PMI) between the topic words. c_uci was proven highly correlated with human judgment. Röder et al. (2015) proposed topic coherence c_v which was identified that it outperformed other topic coherence [12].

3 Online Latent Dirichlet Allocation

Latent Dirichlet Allocation is an unsupervised machine learning model [3]. It identifies latent topics in corpus. Each document in the corpus is represented as probability distribution over some other topics. All the topics are represented as a probability distribution over some words.

Theoretically, a document $\boldsymbol{w}$ in the corpus D is associated with a multinomial distribution over T topics; T is the predefined number of topics, and document is denoted by $\boldsymbol{w} = (w_1, w_2, ..., w_N)$ where w_n is the nth word in the sequence. The multinomial distribution is denoted as θ. One topic is similarly associated with a multinomial distribution ϕ over words w_n.

θ and ϕ whose hyper-parameters are α and β respectively have Dirichlet prior. For every word in one document $\boldsymbol{w}$, a topic z is sampled consequently. This generative process is repeated N times, where N is the number of words in the document [7].

A k-dimensional Dirichlet Random variable has the following probability density:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\Pi_{i=1}^{k} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \dots \theta_k^{\alpha_k - 1} \tag{1}$$

where $\Gamma(x)$ is the Gamma function. The probability of a corpus:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha) \left(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta) p(w_{dn}|z_{dn}, \beta) \right) d\theta_d \tag{2}$$

The probabilistic graphical model representation of LDA is in Figure 1. As shown in the Figure 1, LDA consists of three layers. The parameters α and β are one the level of corpus. θ_d are variables on the level of document. z_{dn} and w_{dn} are variables on the level of word.

However batch LDA still requires to feed the entire corpus into each iteration when the model is being trained. Thus, in some circumstance LDA does not perform well, such as when the datasets are large, or when new data comes in stream to the training process [4].

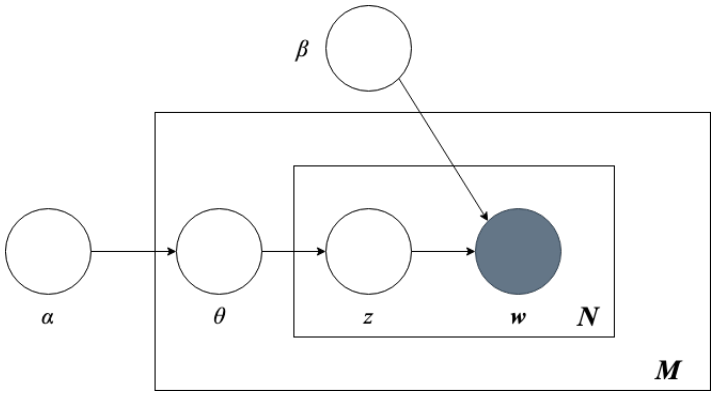

Figure 1: Graphical model representation of LDA

online version

4 Topic Coherence

By combining elementary components, Röder et al proposed the topic coherence c_v, which was proved that it outperforms other topic coherence [12].

5 Visualization

In 2014, Sievert et al. presented a web-based visualization for Latent Dirichlet Allocation [13]

6 Data File

The original data files used for this project are two parquet files of the overall size 1.47 GB. The parquet file in customer_feedbacks/ contains 101 columns, and the one in customer_feedbacks_cat/ contains 7 columns. The content of customer feedback is stored in the column ANTWORT_WERT, There are 1,917,490 feedbacks including duplicates since 14/04/2012.The sentiment of the feedback is stored in the column STIMMUNG.

7 System Architecture and Usage

Under the project root directory, there are cftm/ directory, environment.yml file and README.md file.

environment.yml files specifies the environment dependencies of the system. We can use conda to recreate the system environment. README.md provide brief introduction of the system.

The source code of the system is under cftm/cftm/ directory. It contains three python files which are pipeline.py, cftm_parser.py, preprocessing.py and one yml file, path.yml file.

pipeline.py is the entry of the system, passing arguments to this file in order to control the behavior of the system. cftm_parser.py and preprocessing.py contains modules used by pipeline. cftm_parser.py extracts specified data from data source file; preprocessing.py executes standard nlp preprocessing operations, eventually converting texts to the object which Gensim requires. path.yml file contains all the paths to store system output.

7.1 cftm_parser.py

cftm_parser.py contains one method parquet_transform() to transform data in parquet files to the data format for preprocessing and modelling.

parquet_transform(path1, path2, n=-1)

parameters:

- path1 (str) - the path of the first parquet file
- path2 (str) - the path of the second parquet file
- n (int, optional) - the number of observations to be transformed. When the passed integer is smaller than 0, all observations from data source will be transformed.

returns:

- *pandas.dataframe* - include the data frame with desired columns and the specified number of observations

cftm_parser makes use of Apache Spark, and Spark SQL to import data from parqurt file. Since the size of the parquet files are in the level of gigabytes, directly using method *pandas.read_parquet()* to extract pandas data frame requires large volume of memory. This is sometimes unrealistic when only a personal computer is available for running the system.

An alternative way to import the data is using *spark.read.parquet()*. Instead of fitting all the data to memory, Spark's operators spill data to disk, which make it possible to run over any size of data. *spark.read.parquet()* reads a parquet file from a specified path and return a *spark.data frame* object. Writing queries by Spark SQL makes the data manipulation the same as the process over relational database. After the query the size of the data drops significantly. Sequentially, duplicated observations are dropped and *spark.dateframe* is transformed to *pandas.data frame* for preprocessing procedures.

7.2 preprocessing.py

preprocessing.py utilizes python spaCy to run the standard text preprocessing pipeline, which includes tokenization, lemmatization, lowercasing, striping white spaces, and removing stopwords and punctuations. A token is the name for a sequence of characters that we want to treat as a group, i.e. word, punctuation symbol, white space, etc. Tokenization means to convert a text to a collection of tokens. Lemmatization assigns the base forms of words. For example, assign "run" to "running", "ran", or "be" to "is", "am", "were", etc. Stopwords are the words which appear with high-frequency, such as "also", "the", "to", etc. They too ubiquitously exist in texts, and therefore contain little information about the text. Stopwords and punctuations are removed to only keep words which are distinct.

spaCy[2] is a Python/Cython natural language processing package which currently offers fastest syntactic parser in the word. Choi et al. (2015) conducted an evaluation research proved the outstanding performance of spaCy [14]. The high speed of spaCy makes it especially suitable to processing large volume of text data.

Four methods are in this script. *text_preproc_maker()*, *text_aggregation()*, *gensim_prep()*, and *preprocessor()*.

text_preproc_maker(stopwords, language='de')

A currying function which returns a new function *text_preproc(sentence)* to run the text preprocessing pipeline ,including tokenization, emmatization, lowercasing, striping white spaces, and removing stopwords and punctuations, for the specified language.

parameters:

- stopwords (list of str) - a list of stopwords
- language (str, optional) - the possible inputs are the languages which spaCy supports, such as 'de', 'en', etc. Before use the specified language, first install the spaCy required language, using command *pip install spacy && python -m spacy download en*, here 'en' can be replaced with the language you would like to install.

returns:

[2] Available on *github.com/explosion/spaCy*

- *text_preproc object* - a function takes sentences as the parameter, containing the pipeline of preprocessing.

text_aggregator(df_pd, metadata='DATE', min_len=-1)

parameters:

- df_pd (pandas.dataframe) - a Pandas data frame containing text data and metadata.
- metadata (str, optional) - the column name of the metadata which is used to aggregate texts. Legal options are 'DATE' and 'SENTIMENT'.
- min_len (int, optional) - when the metadata is 'DATE', this argument decides the minimal length of the aggregated text. The default is -1, which means no aggregation, but using the original texts.

returns:

- *list of lists of str* - the text data after aggregation.

gensim_prep(texts)

This method converts list of lists string to gensim dictionary and corpus objects. **parameters:**

- texts (list of lists of str) - text data.

returns:

- *list of lists of str* - the text data.
- *gensim.corpora.dictionary.Dictionary* - A mapping between words and their integer ids.
- *list of lists of (int, int)* - Bag of words representation of the documents.

preprocessor(df_pd, stopwords, language='de', text='TEXT', metadata=None, min_len=-1)

This method wraps up preproc, text aggregator, and gensim_prep to compose a full preprocessing pipeline. It returns text data, gensim dictionary, and gensim bag of words corpus. See the information of the arguments above.

7.3 pipeline.py

pipeline.py is the entry of the system. It consists of three parts preprocessing, modelling, and visualization. Preprocessing part uses the function *preprocessor()* in preprocessing.py. For modelling, gensim implemented Online LDA is used. pyLDAvis module takes the training results from Online LDA, and produce a html file which supports interactive visualization.

7.3.1 Arguments

This program takes 9 arguments. *–pipeline* is a positional argument, and others are optional arguments.

–pipeline is a positional parameter, requiring 3 integer numbers, which switches preprocessing, modelling, and visualization respectively. If one of the numbers is 0, the corresponding function is turned off. If it is none 0, the corresponding function is turned on. For example, if the parameter is *1 1 1*, then the pipeline will be preprocessing, modelling, and visualization. If the parameter is *0 1 1*, then the pipeline will only have the procedures of modelling and visualization, when the training data is read from a pickle file, which is either got from a archive or is copied to the required path. The parameter *0 0 1* has the same logic. *1 0 0* and *1 1 0* also are the legal combinations. Designing the system in this way makes the pipeline flexible and fault tolerant. Since the data volume is big, user can choose which part of the pipeline he would like to run to avoid repeated work. Every part of the pipeline produces a pickle file to archive the outcome, so even if the program is stopped during a run, the results are saved and can be continued next run.

–path_file or *-p* is an optional parameter, requiring a str value. The default value is *./path.yml*. It accepts a path of the yml file in which all output paths are indicated. The yml file uses a dictionary to store paths, and the keys are parquet_path1, parquet_path2, archive_fore_path, data_path, model_path, pic_path, html_path, log_path.

–observation_n or *-o* indicates the number of observations used in the program, requiring a integer. The default value is -1, which means use all observations.

–agg_metadata or *-m* indicates the metadata which is used during the aggregation session. The default value is 'DATE', and the other legal value is 'SENTIMENT'.

–agg_length or *-l* indicates the minimal length of every document after an aggregation session, if the metadata 'DATE' is used. It requires an integer number, and the default value is -1, which means the aggregation is aborted.

–topic_range or *-r* indicates the topic range during the modelling session. It requires 3 integer numbers, meaning the minimal number of topics, the maximal number of topics, and the step. The default value is *1 10 1*, so the modelling session will generate 10 LDA models with topic numbers from 1 topic to 10 topics.

–chunksize or *-c* is an hyperparameter to Online LDA model. Online LDA algorithm update the model every chunksize. For example, if a corpus has 10000 documents, and the chunksize is set to 1000, then Online LDA will update the model 10 time through one complete pass of the corpus. The default value is 2000.

–iteration or *-i* indicates the number of iterations during optimization, requiring a integer number. The number of iterations should high enough to make the model converge. However, fewer iterations make the modelling time shorter. The default value is 50.

–seed or *-s* indicates the seed passed to LDA model, make the whole process replicable.

For every run of the program, parameters are saved in *commandline_args.yaml* file in the archive folder.

7.3.2 Modelling

gensim[3] is an open-source library for unsupervised topic modelling and natural language processing. This software framework was proposed by Řehůřek in 2010, featuring with its clarity, efficiency and scalability [15]. gensim supports parallel computing and is scalable through a cluster, which accelerate training process significantly. For this customer feedback topic extraction system, gensim Online LDA model is chosen.

[3] Available on github.com/RaRe-Technologies/gensim

7.3.3 Visualization

8 Results

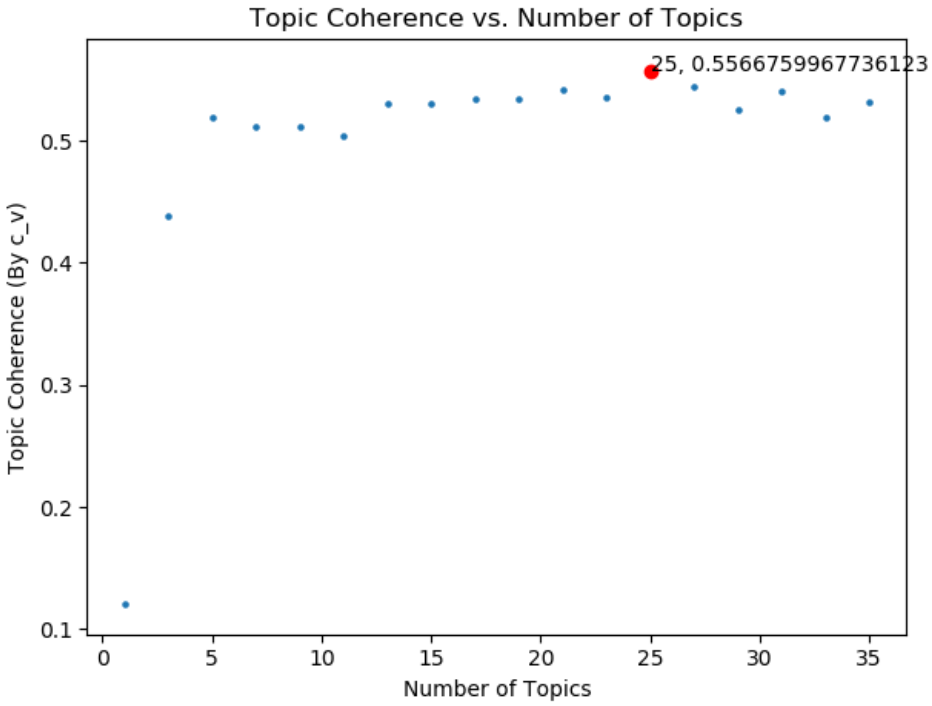

Figure 2: Topic Coherence vs. Number of Topics

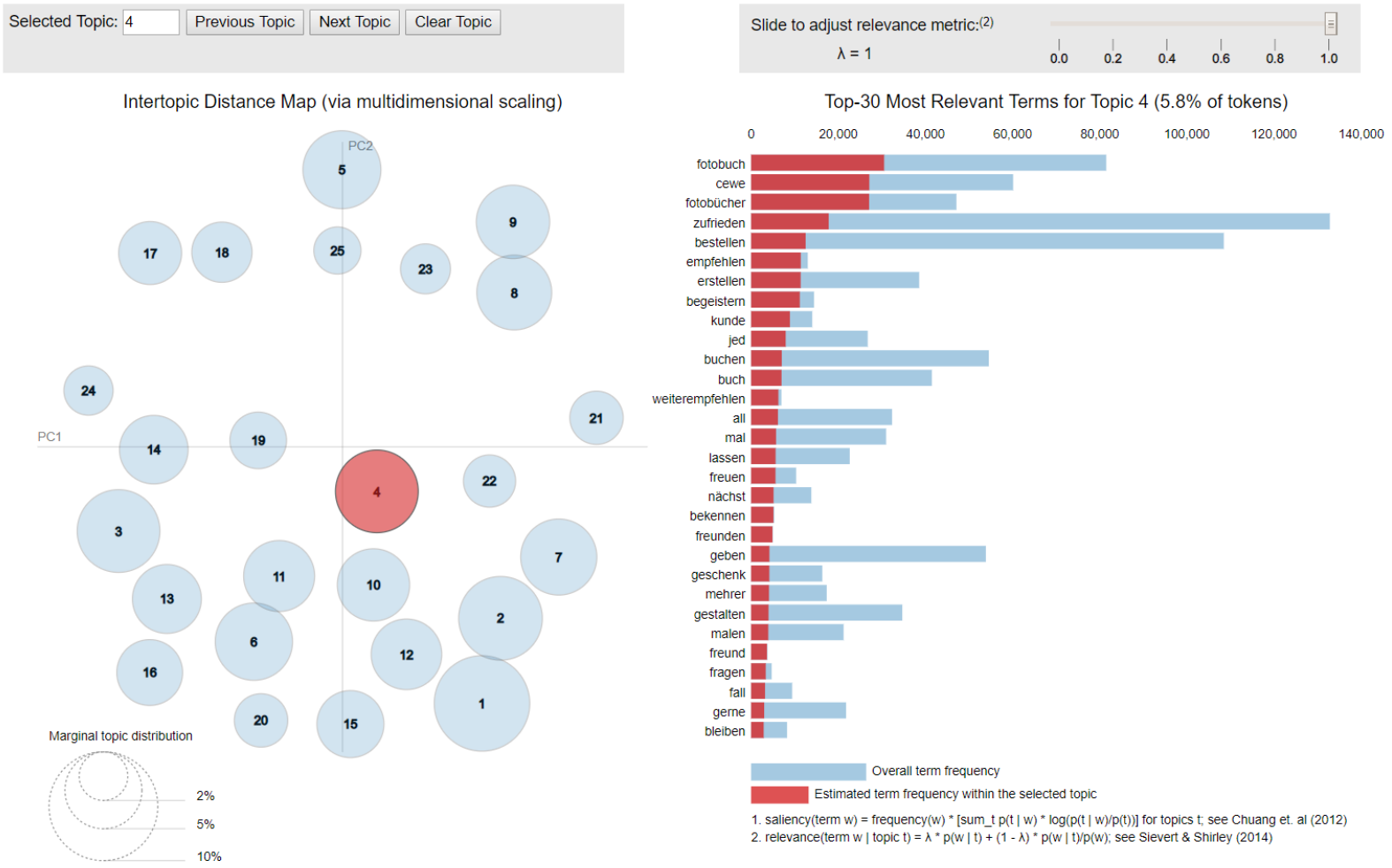

Figure 3: Screenshot of the Modelling Visualization

9 Summary

10 Outlook

Generalize the system, provide an interfaces where get different parsers.

The evaluation metrics and be improved. Some results from different topic coherence indexes conflict with each other. For example, topic coherence generated from c_umass [16] and c_v [12] are often negatively correlated.

Better user interface, utilizing tools such as Semantic UI.

Containerize the system.

Acknowledgments

References

[1] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, Sep. 1990. [Online]. Available: http://doi.wiley.com/10.1002/%28SICI%291097-4571%28199009%2941%3A6%3C391%3A%3AAID-ASI1%3E3.0.CO%3B2-9

[2] T. Hofmann, "Probabilistic Latent Semantic Indexing," *ACM SIGIR Forum*, vol. 51, no. 2, p. 8, 2017.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003. [Online]. Available: http://www.jmlr.org/papers/v3/blei03a.html

[4] M. Hoffman, F. R. Bach, and D. M. Blei, "Online Learning for Latent Dirichlet Allocation," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 856–864. [Online]. Available: http://papers.nips.cc/paper/3902-online-learning-for-latent-dirichlet-allocation.pdf

[5] C. Li, H. Wang, Z. Zhang, A. Sun, and Z. Ma, "Topic Modeling for Short Texts with Auxiliary Word Embeddings," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval - SIGIR '16*. Pisa, Italy: ACM Press, 2016, pp. 165–174. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2911451.2911499

[6] J. Weng, E.-P. Lim, J. Jiang, and Q. He, "TwitterRank: finding topic-sensitive influential twitterers," in *Proceedings of the third ACM international conference on Web search and data mining - WSDM '10*. New York, New York, USA: ACM Press, 2010, p. 261. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1718487.1718520

[7] L. Hong and B. D. Davison, "Empirical study of topic modeling in Twitter," in *Proceedings of the First Workshop on Social Media Analytics - SOMA '10*. Washington D.C., District of Columbia: ACM Press, 2010, pp. 80–88. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1964858.1964870

[8] R. Mehrotra, S. Sanner, W. Buntine, and L. Xie, "Improving LDA topic models for microblogs via tweet pooling and automatic labeling," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '13*. Dublin, Ireland: ACM Press, 2013, p. 889. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2484028.2484166

[9] H. M. Wallach, I. Murray, R. Salakhutdinov, and D. Mimno, "Evaluation methods for topic models," in *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*. Montreal, Quebec, Canada: ACM Press, 2009, pp. 1–8. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1553374.1553515

[10] J. Chang, S. Gerrish, C. Wang, J. L. Boyd-graber, and D. M. Blei, "Reading Tea Leaves: How Humans Interpret Topic Models," in *Advances in Neural Information*

Processing Systems 22, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 288–296. [Online]. Available: http://papers.nips.cc/paper/3700-reading-tea-leaves-how-humans-interpret-topic-models.pdf

[11] D. Newman, J. H. Lau, K. Grieser, and T. Baldwin, "Automatic evaluation of topic coherence," p. 9.

[12] M. Röder, A. Both, and A. Hinneburg, "Exploring the Space of Topic Coherence Measures," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15*. Shanghai, China: ACM Press, 2015, pp. 399–408. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2684822.2685324

[13] C. Sievert and K. Shirley, "LDAvis: A method for visualizing and interpreting topics," in *Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces*. Baltimore, Maryland, USA: Association for Computational Linguistics, 2014, pp. 63–70. [Online]. Available: http://aclweb.org/anthology/W14-3110

[14] J. D. Choi, J. Tetreault, and A. Stent, "It Depends: Dependency Parser Comparison Using A Web-based Evaluation Tool," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 387–396. [Online]. Available: https://www.aclweb.org/anthology/P15-1038

[15] R. Řehůřek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, http://is.muni.cz/publication/884893/en.

[16] D. Mimno, H. M. Wallach, E. Talley, M. Leenders, and A. McCallum, "Optimizing semantic coherence in topic models," p. 11.