Delving into the Q&A Network – Text Mining and Graph Analysis

Zhen Liang, Xinli Wang
Department of Statistics, Columbia University
zl2406@columbia.edu xw2341@columbia.edu

***Abstract*— Stack Exchange is a popular Q&A platform covering various fields. Getting an idea of what topics users are talking about, helping them filtering best answers and visualize question network are concerned in this paper. To accomplish these, LDA topic modeling method are applied for generating topic features. Then building random forest classification model to predict best answers. The features used in classification including answer quality, attitude, response time, answerer's reputation and topic features. The results show answer quality and response time is important in determining best answers. At last, heated topics and tags are visualized using Neo4j and d3.js for people to get ideas of the popularity and connection of the tags.**

Stack Overflow; Sentiment Analysis; Classification; LDA; Neo4j; Graph Visualization

I. INTRODUCTION

Stack Exchange is a platform where software engineers, scientists, students share knowledge and get questions answered. With increasing amount of posts, we want to get an abstract or overview of the topics and get an idea of the trend of the knowledge and technology.

As users, we are interested in what are heated discussed topics; and how to filter best answers among all the given answers. With increasing amount of activities, one question may get several answers. Helping users, especially novices, recognize best answers and rank them in high order not only solves their questions quickly, but also improves website's working efficiency.

As developers, we are interested in: The problems users are facing and how they can take such information to improve their products and documentation.

Our project addresses such problems by extracting topics out of large amount of posts and the topic distribution of each document, predicting the best answers by building a classification model and visualizing the "network" of questions, to know what's the trends and relationships among discussed topics

II. RELATED WORKS

There are several researches on answers quality using Stack Overflow dataset. Social reputation and answer length are included in selecting good information[3].

Topic models are algorithms for discovering the main themes of a large and unstructured collection of documents. "Topic models can organize the collection according to the discovered themes. A topic to be a distribution over a fixed vocabulary. Each document exhibits the topics in different proportion; each word in each document is drawn from one of the topics, where the selected topic is chosen from the per-document distribution over topics." [2]Latent Dirichlet Allocation (LDA) can be thought of as a mixed-membership model of grouped data, each group of documents exhibits multiple components in different proportions of topics.[2]

III. SYSTEM OVERVIEW

We used Stack Exchange Data Dump [9] data science category posts created from May 2014 to August 2015, and all the users and tags data related with the posts. The data structure can be simplified as the following diagram.

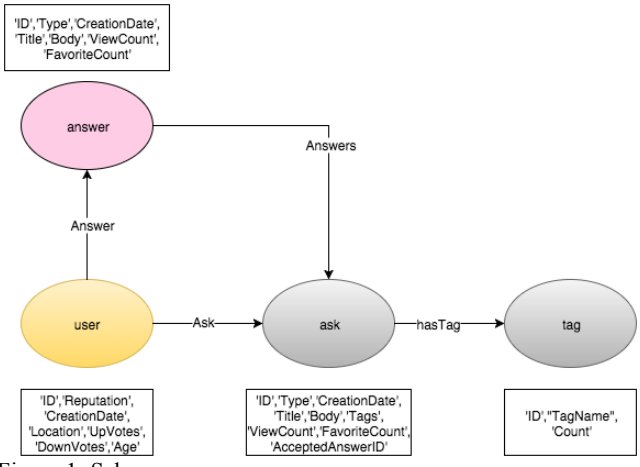
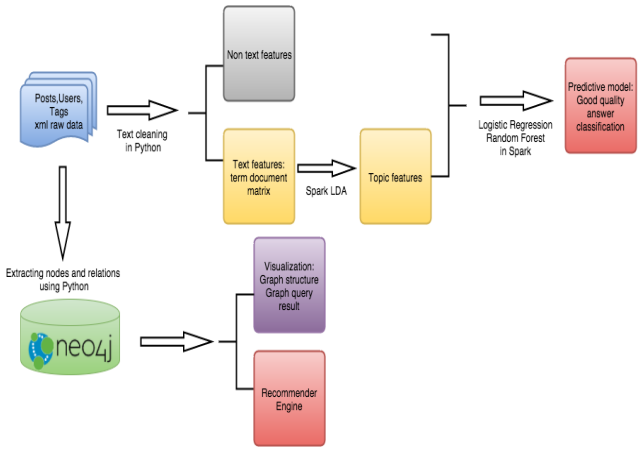


Figure 1: Schema

Our project structure is as follows:



IV. ALGORITHM

4.1 Latent Dirichlet Allocation

Given a collection of M documents denoted by D = $\{w_1,w_2,...,w_M\}$, each document w = $(w_1,w_2,...,w_N)$ is a sequence of N words denoted by where w_n is the nth word in the sequence.

The model makes "bag of words" assumption, which means the order of the words in the document does not matter.

The number of words N ~ Poisson(ξ).

The topic mixture θ ~ Dir(α).

A topic z_n ~ Multinomial(θ).

A word w_n from $p(w_n|z_n,\beta)$, a multinomial probability conditioned on the topic z_n.

Given the prior parameters α and β, and the observation of the bag of words in each documents, the joint distribution of a topic mixture θ, a set of N topics z, and a set of N words w is given by[1]:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha)\prod_{n=1}^{N} p(z_n|\theta)p(w_n|\beta, z_n)$$

The graphical model can be shown by Figure 2.

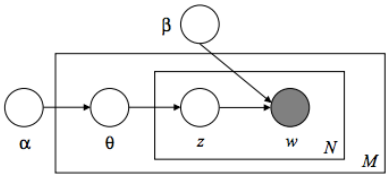


Figure 2: Graphical model representation of LDA.
The boxes are "plates" representing replicates. The outer plate represents M documents, while the inner plate represents the repeated choice of topics and words within a document. Words in each documents are observed.[1]

The inference[1] we care about is

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

4.1.1 Packages for Computation

We did inference of the word distribution in each topic given the corpus, and the proportion of each topic in each documents. We chose Spark LDA, which takes in a collection of documents as vectors of word counts and an arbitrary number of topics we wanted to compute. We tried 10 topics and 30 topics, and for the simplicity of demo, we display the results of the model of 10 topics.

4.2 Feature Selection and Random Forest

Some good quality answers are accepted by the people who raises the questions. In this part, we built a model to predict what answers would likely to be accepted, so as to predict the good quality answers.

4.2.1 Important Factors

Factors used in classification include answer quality, attitude, response time, answerer's reputation:

Answer quality includes **length, URL count[3]**, **score**, **comments count**. As longer answers might contain more details, they are more comprehensive. So longer answers can be considered as higher quality. However, longer answers also fatigue reader, which cannot be high quality.

Score is calculated by number of upvotes minus downvotes, which is quite strong indictor showing users' satisfaction.

Comments amount is included because more comments means more users concerns about the answers, whether they have follow up questions or updated answers to the questions.

URL count is another feature that affects post quality. With links included, readers get more references to the answers.

Attitude is important. It happens that harsh responses drive users away from Stack Exchange. Also, previous research shows negative emotions in comments discourage participating. In this paper, sentiment score is used to measure emotion intensity in answers. The score is between -1 and 1, which -1 means absolute negative and 1 represents absolute positive.

Response time is the time period between a question is posted and an answer is given. It is also a factor worth to be considered. As research pointed out that the longer wait to get answers, the less likely have accepted answers. Also, median waiting time for an accepted answer is 21 minutes. So faster contributors have higher possibility getting answers accepted.

Answerers' reputation is an other good filter. In Stack Exchange Community, reputation is gained primarily by posting useful answers and good questions. So in some sense, reputation is also a useful indicator.

4.2.2 Random Forest

Random forest is chosen for classifying good answers out of other answers. The reason is that random forest is an ensemble learning method for classification constructed on multiple decision trees, reducing overfitting on training set.

Dummy variable of an answer is accepted or not is used as depend variable. Five Factors, answer quality, attitude, response time, answerers' reputation and topic features are independent variables.

Results are obtained by randomly splitting whole dataset into training (80%) and test (20%) sets.

V. Software Package Description

5.1 Graph database

We set up a graph database in Neo4j using the data structure describe in Figure 1. Below is snap shot of the graph visualization of the database of the above structure. Where the yellow nodes are users, the red nodes are answers, and the grey nodes with time stamp are questions, and the other grey nodes are nodes.

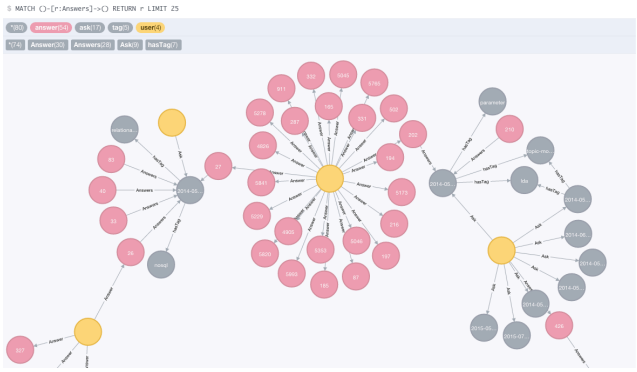


Figure 3: Snapshot of the graph database.

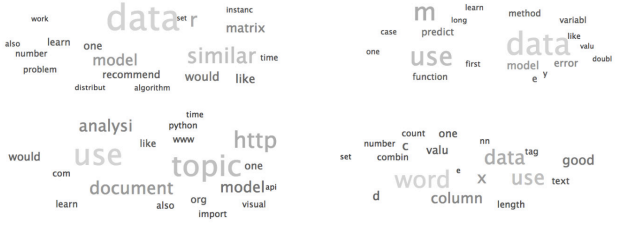
5.2 Visualization

One of our goals is to understand the database. We choose visualization as our tool.

5.2.1 Topic WordCloud

To get topics of all the posts, we visualized the topics computed by LDA model using WordCloud[10], which captures the most important words and the weights in each topic.

The 10 topics we generated are as follows:



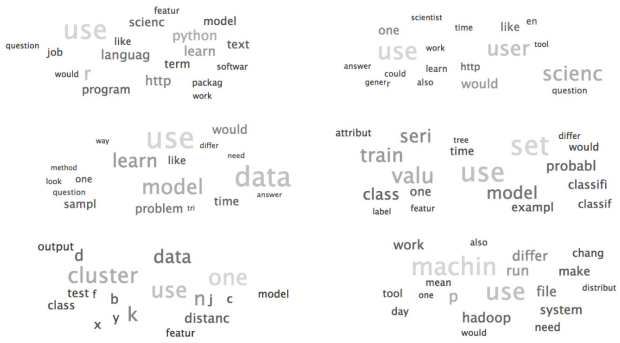


Figure 4: Topic graph

From the result we can see some topics are about data programming languages, such as R or Python, some are about machine learning and algorithm, and some are about data processing such as sampling and combining. Though some common words such as "would like", "use" are commonly shared by many topics.

5.2.2 Tag Network

An important feature of the Stack Exchange database is that most questions are tagged. We can use tags as an important feature to describe the database.

We used Py2Neo package in Python to connect to the database and use Python to process the query result, and output a proper json format as input of our d3.js interactive visualization page.

5.2.3 Force directed graph[11]

Figure 5 shows the force directed graph of the topics. The thickness of the links represents the number of co-occurrence between the two topics. The tags in the center of the cluster have more connections with other tags, which shows its popularity and importance. The the tags at the outer range of the cluster have lass connections.

One disadvantage is that we cannot see clearly the connectivity in the center of the cluster because of many edges intersect and overlaps together.

5.2.4 Edge bundling graph[12]

To overcome the disadvantage, we mentioned above, we implemented the edge bundling graph. Again, we queried the data from the database and convert to json format. From the graph, Figure 6, we can see how each tag connects with other tags. We can also get information of how popular a tag is from the thickness of the links connected with it.

5.3 Simple recommendation computed by the query using Cypher.

Another useful feature of this database is that, with the convenience of relation queries, we can easily find questions

that have no answers yet and recommend some users that could give a feasible answer.

Below is an example result of recommending users that have asked similar questions with the same tags of question that has no answer.

questionID	userID	t.TagName
16	51	machine-learning
61	51	machine-learning
115	51	machine-learning
116	51	machine-learning
159	51	machine-learning
169	51	machine-learning
196	51	machine-learning
211	51	machine-learning

Table 1: result of recommending users that have asked similar questions with the same tags of question that has no answer.

VI. Experiment Results

In the classification problem mentioned in previous session, accepted answers are labeled as group 1, and not accepted answers are labelled as group2. To get more specific results, accepted answers that are answered by users themselves are deleted. Because some users do it on purpose in order to get more score.

Also, we deleted accepted answers that are edited after accepted. The length, URL count are change after editing.

The classification result is assessed by Receiver Operating Characteristic Area Under Curve (AUC)[4] AUC = 0.50 is set as baseline. Classification was applied on subsets of training and test sets by removing one of features at a time. Thus through each performance, evaluating the contribution of each factor is obvious. Other results F score (F), precision (PREC) and recall (REC) also listed.

	AUC	**F**	**PREC**	**REC**
All Factors	0.60	0.77	0.80	0.81
w/o Reputation	0.62	0.79	0.82	0.83
w/o Quality	0.58	0.76	0.76	0.80
w/o Time	0.52	0.70	0.73	0.78
w/o Topic Features	0.73	0.85	0.85	0.86
w/o Attitude	0.64	0.79	0.80	0.82

Table 2: Prediction Results for Random Forest Models Using Different Sets of Features

From table above, the best model (0.73) based on AUC is the one without topic features. Also, time is the most influential feature. This suggests users be confident to fast answers. At the same time, when a question dose not get quick answer, it is also possible not a high quality question. And as expected, answer quality like length, URL counts, comments and score is important in filtering best answer. This guides users to pay more attention on detailed and heated discussed answers.

It is noticeable models without reputation or attitude also perform better than full model. As we mentioned before, reputation evaluates a user's performance in forum in general. So it is not a good feature when quality of answers is considered. Attitude is not as important as we think. It might be result of Stack Overflow guideline "Be nice."[6].

VII. Conclusion

In this paper, we used LDA topic modeling method and tag network visualization to get an idea of what the topics people are talking about in Stack Exchange data science platform.
We built a classification model using random forest to classify good quality answers out of other answers with topic features as well as four other kinds of non-text features. For classifying best answers, accepted answers are considered as best answers. Then random forest is applied to perform classification using features answer quality, attitude, response time, answerer's reputation and topic features.

There are some areas of our topic model we can improve on. For example, in many text analysis settings, the documents contain additional information—such as author, title, geographic location, links, and others—that we might want to account for when fitting a topic model.

Contribution:
Zhen Liang: ideas and the dataset, cleaning text and generating term-document matrix, Spark LDA modeling, Neo4j graph database, visualization part, writing corresponding part of presentation and report, recording videos, finalizing the report.

Xinli Wang: generating features including sentiment analysis, text mining and topic model. Building classification model using random forest, doing feature selection and getting results, writing corresponding presentation and report. Polishing report,

Acknowledgment

We would like to thank Professor Lin fro teaching us this semester, and all TAs for helping us out when we had problems doing our homeworks.

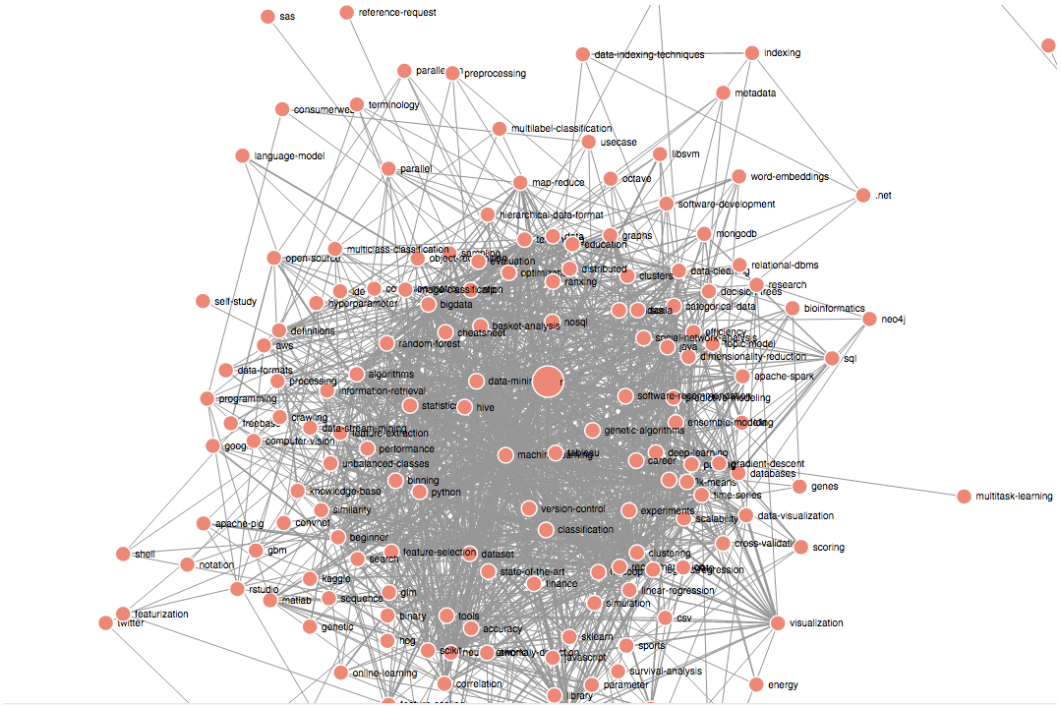


Figure 5 Tag Network

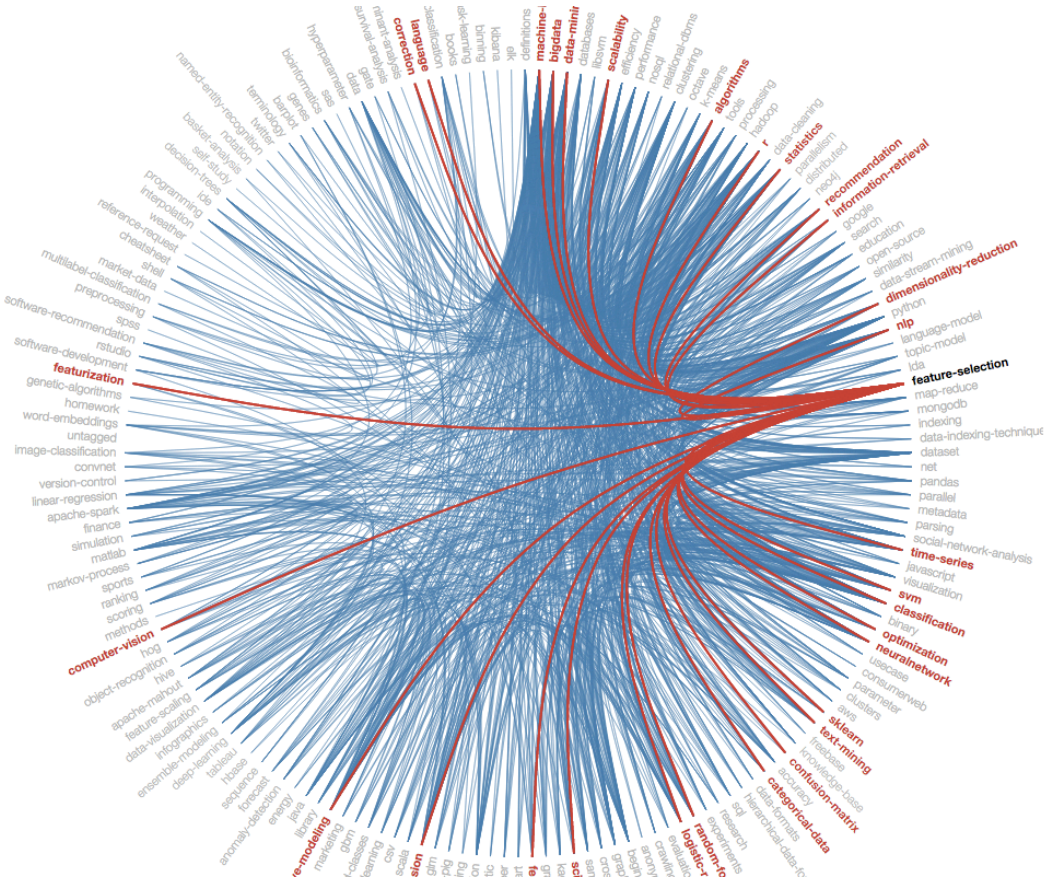


Figure 6 Edge bundling graph

APPENDIX

https://github.com/zhen-liang/stackexchange

1. LDA; intermediate results are saved in result folder

- **make_tdm.ipynb** or **make_tdm.py** clean the dataset using the module **Word2VecUtility.py**[1] and then generate the work-document matrix **matrix.csv** for the input of our LDA model.
- **lda_spark.py** uses the **matrix.csv** as input, and generate a **lda_topicMatrix.csv** where each column is a topic, and each row is a word with the same index of **matrix.csv**.
- **lda_spark.scala** uses the **matrix0.csv**, which is the matrix.csv without header, to generate a **topicDist.txt**, which is the topic distribution of each document.
- **clean_topic_dist_result.ipynb** or **clean_topic_dist_result.py** clean the raw output of **topicDist.txt**, and output the result **topic_Distribution_for_each_doc.csv**, of which the first column is the index of each document, and the following 10 columns are the distribution on the 10 topics. And then they link the result with document ID, i.e. output the topic distributions of each document ID. The output is **doc_id_topic_doc_dist.csv**.
- **get_top20_words_each_topic.ipynb** extracts the top weighted 20 words for each topic. The output is **topic_word_distribution.csv**.

2. In the "graphAnalysis" folder, the intermediate results are stored in "outputData" folder.

- **nodeCreate.ipynb** or **nodeCreate.py** extract **the** schema from **Posts.xml**, **Tags.xml** and **Users.xml** files, and output the corresponding csv files (table format) for later uses. The output includes **post.csv**, **user.csv**, **tag.csv**, **post_relation.csv**, **userPost.csv**, **post_tag_relation_frame.csv**. All the above mentioned data are in "data" folder under the root directory.
- **post_user_database.cypher** is cypher code conducting creation of nodes and links in Neo4j database.
- **useGraphDB.ipynb** walks through some examples of using Neo4j StackExchange database to do some analysis, integrating packages with Neo4j, including py2neo, ipython-cypher.

Specifically, it queries the databases to get the tag-connection network data, which can be visualized using a force-directed graph and an edge bundling graph.

3. Visualization part:

Under "Apache-tomcat/webapps/d3" folder, after running ./bin/startup.sh to start up tomcat, we can visualize our data from the route http://localhost:8080/d3/…

We visualize the StackExchange dataset by creating the following graphs:

wordCloud: the input json format can be generated by **get_top20_words_each_topic.ipynb**.

forceDirectedGraph: which is a force directed graph with input json data generated by **useGraphDB.ipynb**.

hirarlinks: which is an edge bunding graph with input json data generated by **useGraphDB.ipynb**.

REFERENCES

[1] D.M. Blei, A.Y. Ng, and M.I. Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, *3*, pp.993-1022.

[2] D.M. Blei, 2012. Probabilistic topic models. Communications of the ACM, 55(4), pp.77-84.

[3] K. Hart and A. Sarma. 2014. Perceptions of answer quality in an online technical question and answer forum. Proc. of the 7th Int'l Workshop on Cooperative and Human Aspects of Software Engineering (CHASE 2014). ACM, 103-106.

[4] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza, D. Fullerton 2014. Improving Low Quality Stack Overflow Post Detection. ICSME 2014, 541-544.

[5] F. Provost, T. Fawcett, and R. Kohavi. 1998. The case against accuracy estimation for comparing induction algorithms. Proc. of the 15th Int'l Conf. on Machine Learning (ICML), 445–453.

[6] http://stackoverflow.com/help/be-nice

[7] M. Asaduzzaman, A.S. Mashiyat, C.K. Roy, K.A. Schneider. 2013. Answering questions about unanswered questions of Stack Overflow, Proc. of MSR 2013, 97-100.B.

[8] https://github.com/phuston/DeepLearningMovies/blob/master/Word2VecUtility.py

[9] Stack Exchange Data Dump : Stack Exchange, Inc. : Free Download & Streaming : Internet Archive (Internet Archive), https://archive.org/details/stackexchange.

[10] D3 Word Cloud implementation (D3 Word Cloud implementation), http://bl.ocks.org/ericcoopey/6382449.

[11] Force-Directed Graph with Mouseover (Force-Directed Graph with Mouseover), http://bl.ocks.org/mbostock/2706022.

[12] Hierarchical Edge Bundling (Hierarchical Edge Bundling), http://bl.ocks.org/mbostock/7607999.