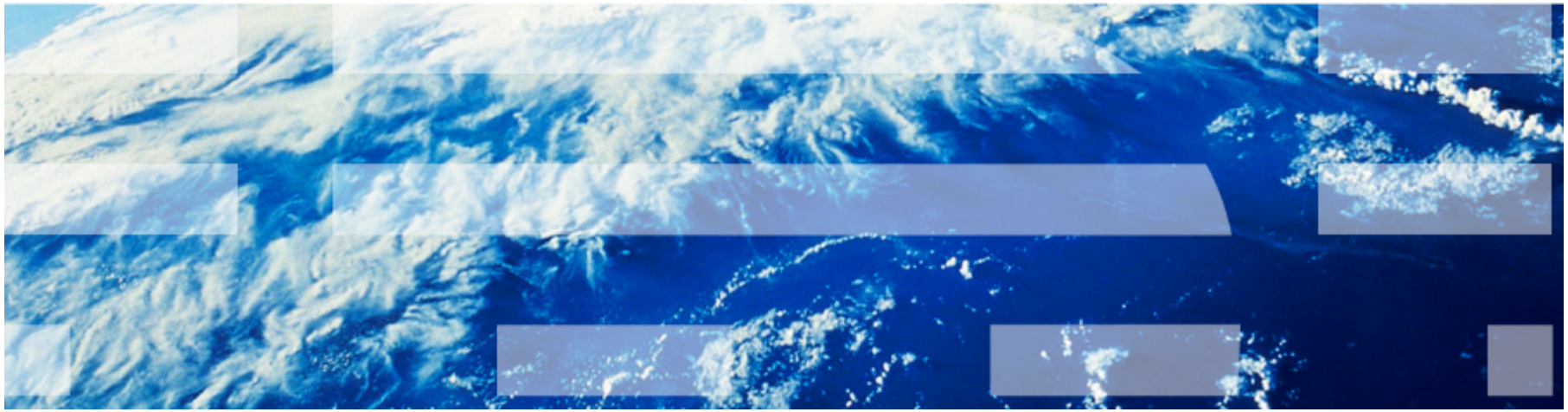
E6893 Big Data Analytics:

Delving into the Q&A network – text mining and graph analysis

Zhen Liang, Xinli Wang
Zl2406, xw2341



December 17, 2015

Agenda

- **Project Overview**
- **Motivation**
- **Tools and Data Sets**
- **Algorithms**
- **Performance**
- **Future Work**

Motivation

- Stack Exchange is a Q&A platform where software engineers, scientists, students share knowledge and get questions answered.

- As users, we are interested in:
  - What are heated discussed topics
  - How to filtering best answers among all the given answers

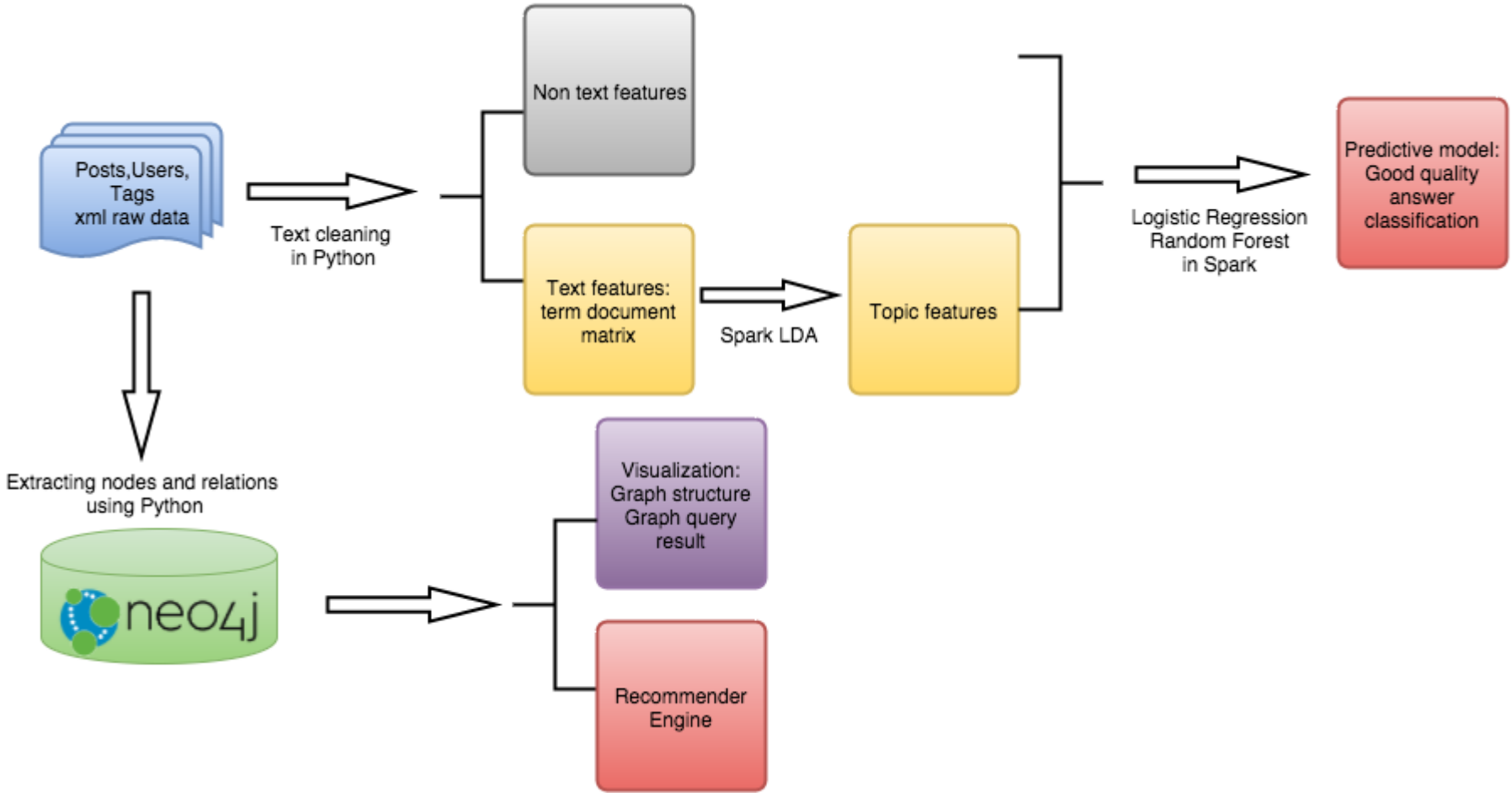
  As developers, we are interested in:
  -The problems users are facing and how they can take such information to improve their products and documentation.

- 

  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
  - Visualizing the “network” of questions, to know what’s the trends and relationships among discussed topics

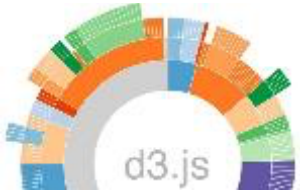
Project Overview

- Text mining
- Graph database and visualization



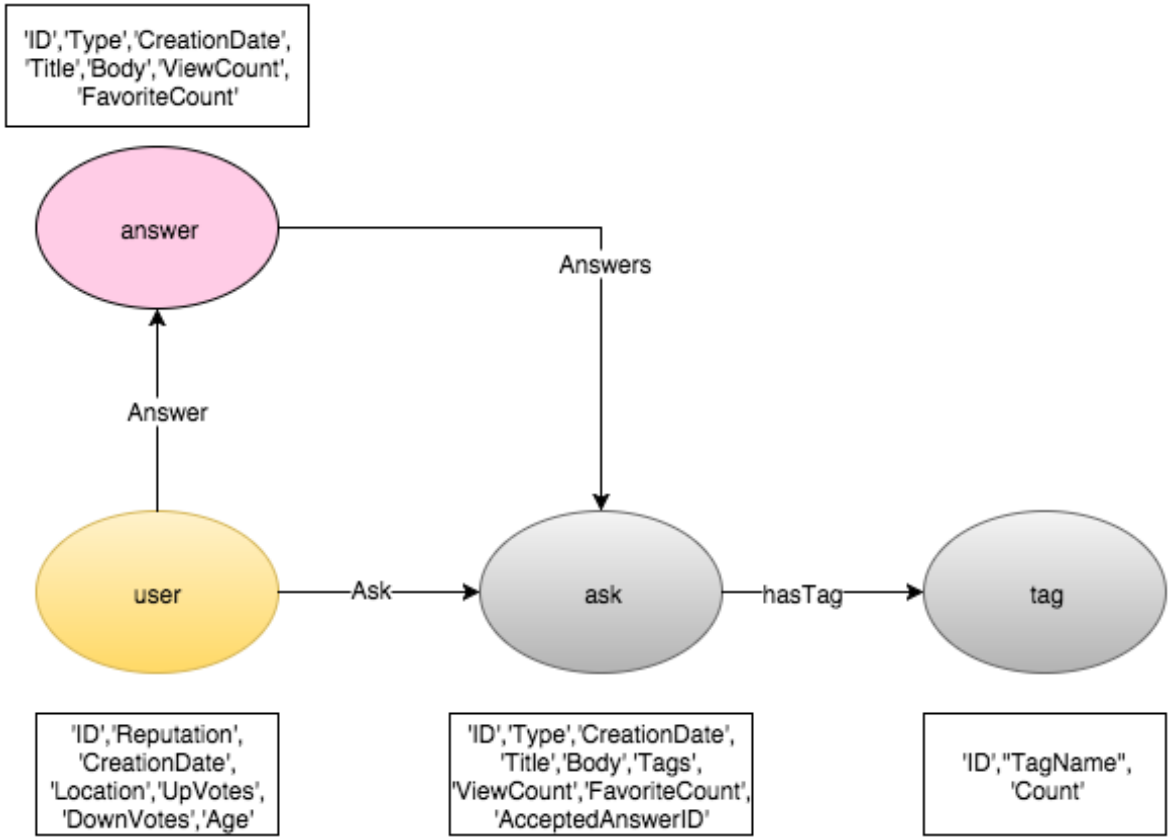
Tools

- Python
- Spark
- Cypher
- D3 and JavaScript



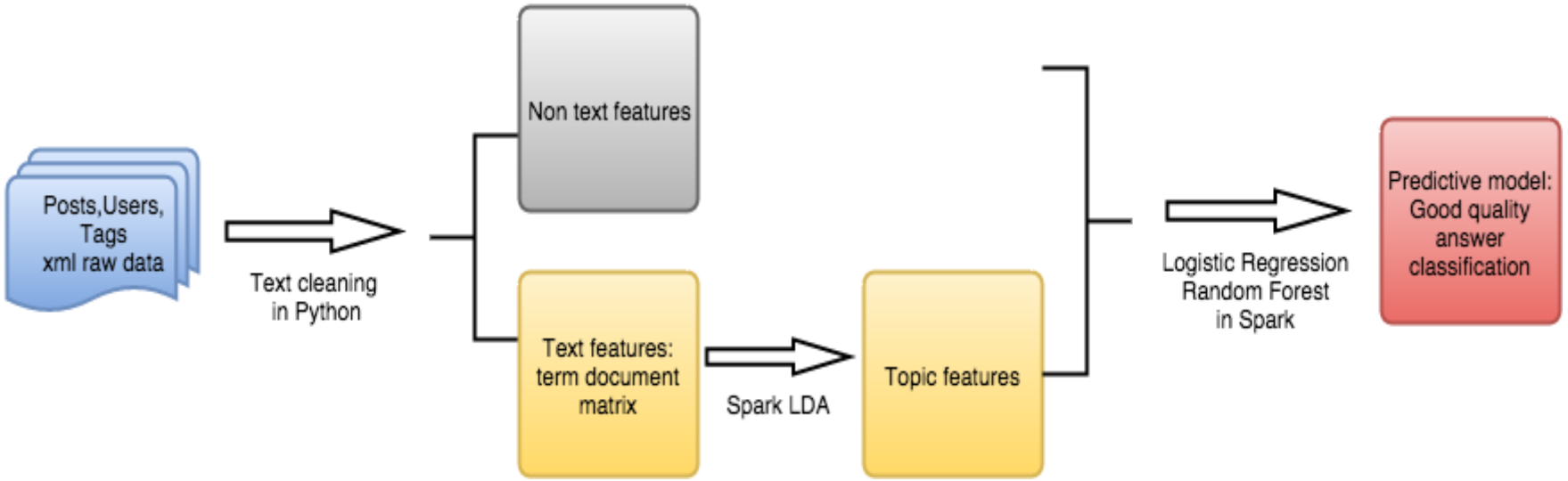
Data Sets

- Q&A data from Stack Exchange Data Dump, we choose the data science categories to conduct our analysis.
- Containing information of users, posts, and users-posts interaction.
- Specifically, information of posts, their links, votes tags, history, users information



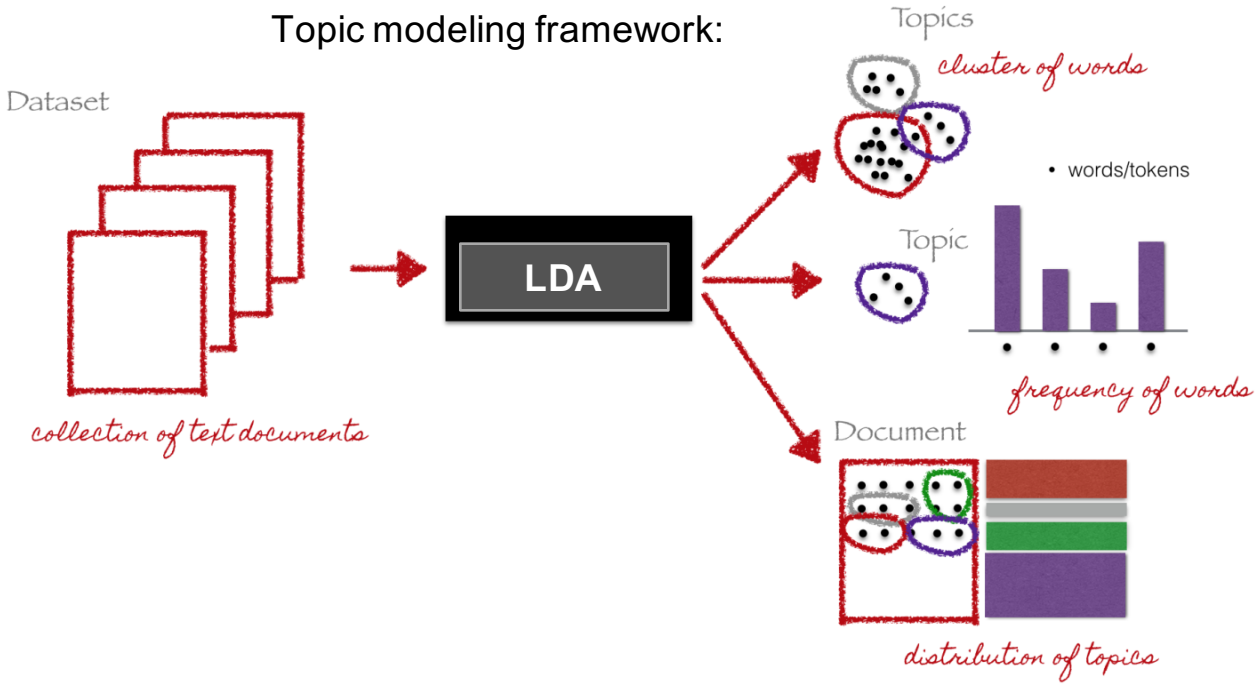
Textming and Classification:

- Algorithms for data and Video pre-processing.
- Event-related field/response analysis.
- Classification and Collaborative Filtering.



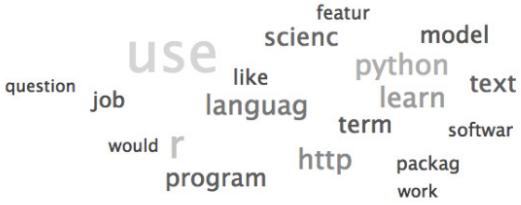
Text Feature Generation: Latent Dirichlet Allocation (LDA)

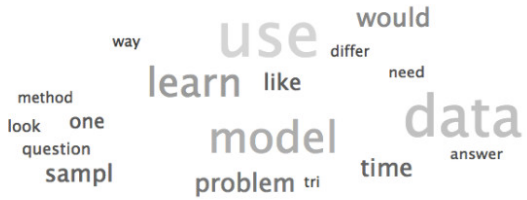
- Clean text
- Term document matrix
- Run Spark LDA using Python API to get an arbitrary number of topics (e.g. 10 topics)
- Run Spark LDA using Scala API to get the topic distribution on each document

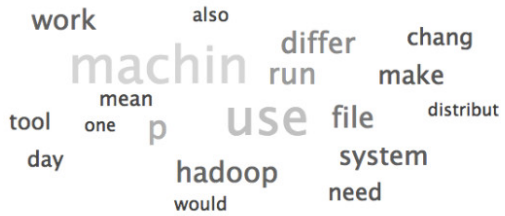


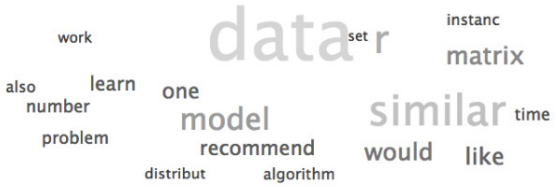
Text Feature Generation: LDA result

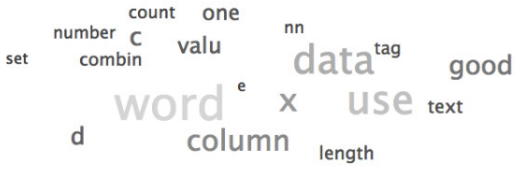
Top 10 word distribution on each topic:











Predicting the good quality answers

Some good quality answers are accepted by the people who raises the questions. Building a model to predict what answers are going to be accepted, so as to predict the good quality answers:

Objective:
Predicting good quality answers is useful for

1. ranking the answers
2. understanding the successful features that a good quality answer have so as to provide the direction to improve the overall answers quality.

Definition: Accepted answers are defined as best answers
There are two classes of answers in our dataset, accepted vs. not accepted.

Pre-processing data: delete accepted answers that are given and accepted by the same users or that have been edited after being accepted.

Feature Extraction

A. Presentation Quality

- ❏ Length (count of words in answers' posts)
- ❏ URL count
- ❏ Score = upvotes - downvotes
- ❏ Comment Count

A. Attitude

Scores are returned to describe intensity emotional style of an post. The results are between -1 and 1.
alchemyAPI is used for sentiment analysis

A. Time

The waiting time between an question posted and an answer is given (in seconds)

A. Reputation

Answerer's Reputation

E. Topic features

Methods used for Classification

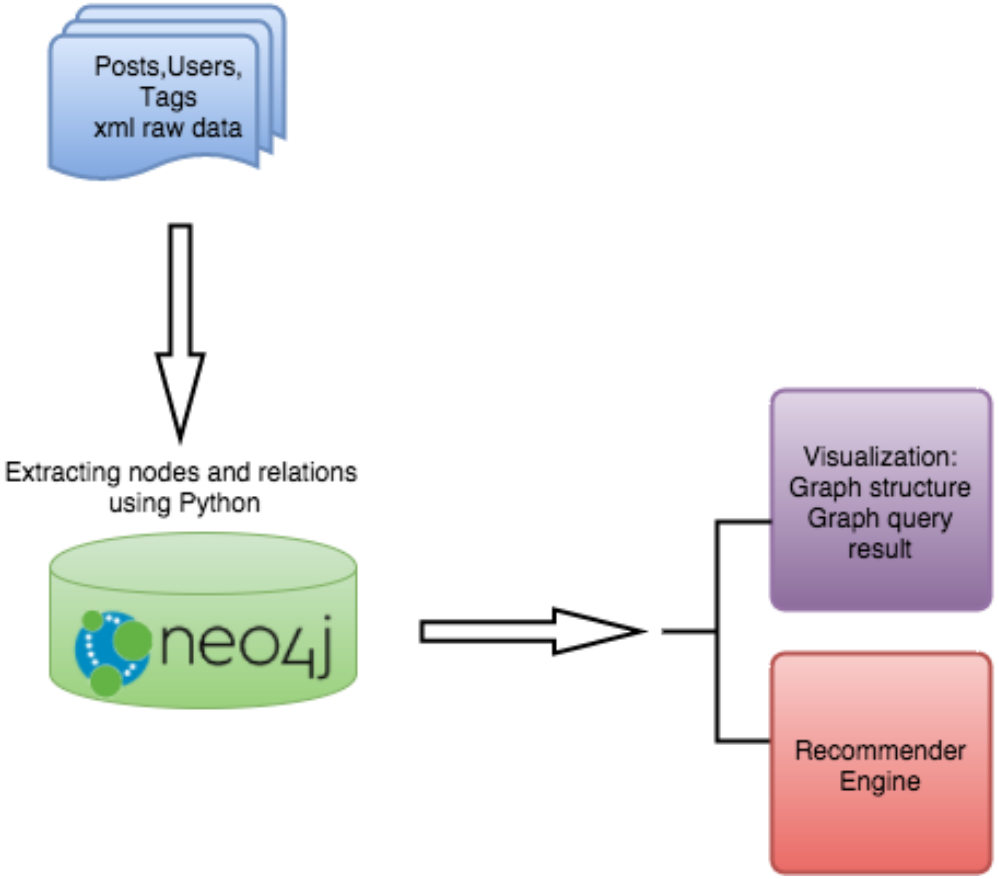
1. Splitting dataset into train and test (4: 1) randomly
2. Applying Random Forest on all features
3. Classification is applied on subsets of training and test sets by removing one of features at a time.

	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

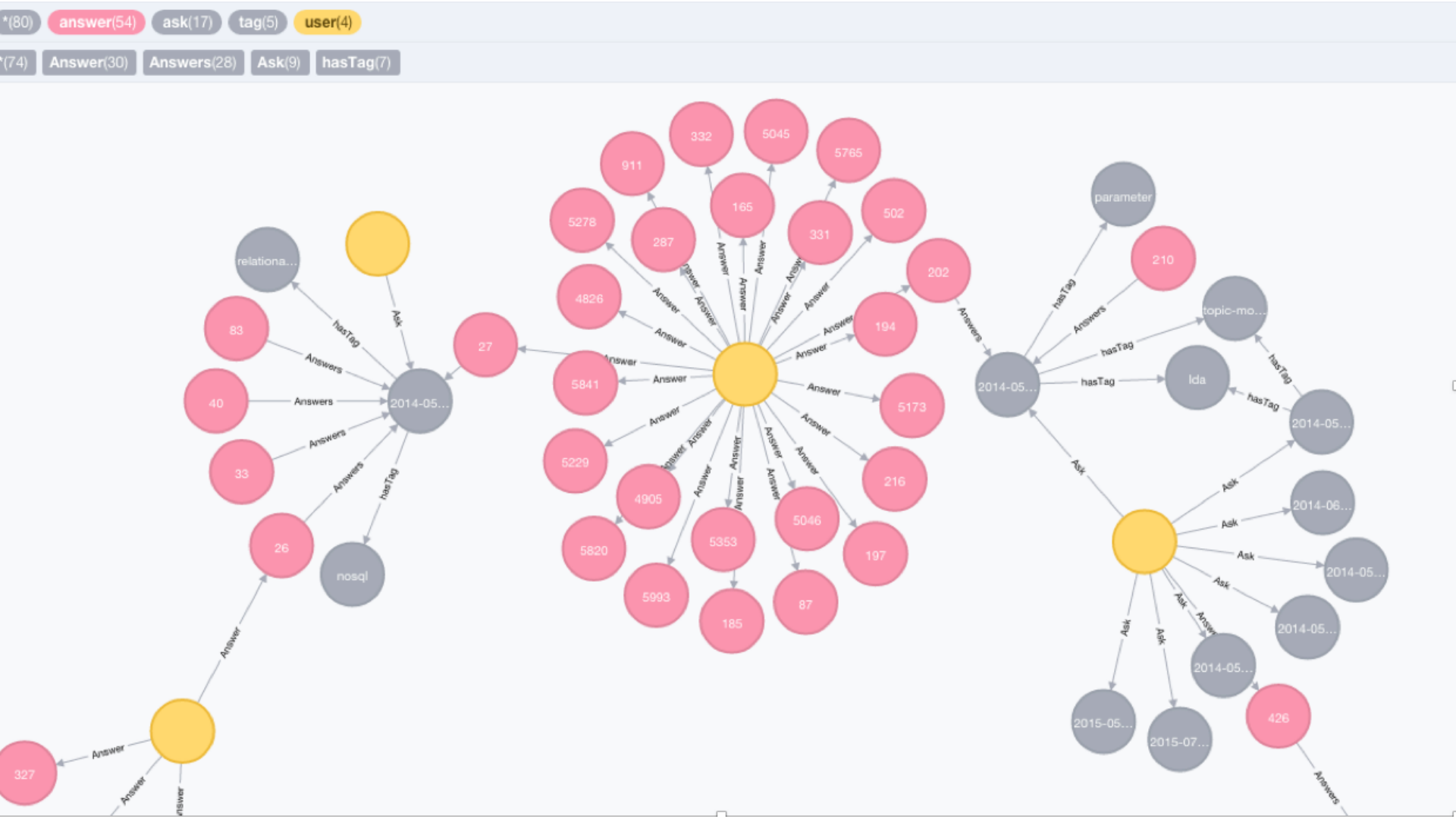
Table 1 Prediction Results for Random Forest Models Using Different Sets of Features

Algorithm

- Graph database and visualization



Neo4j Graph Database - a user-post network



Neo4j Graph Database - query results for building a tag networks

next we are going to extract data from Neo4j and draw a network graph of tags using Force-Directed Graph

In [17]:

Slide Type -

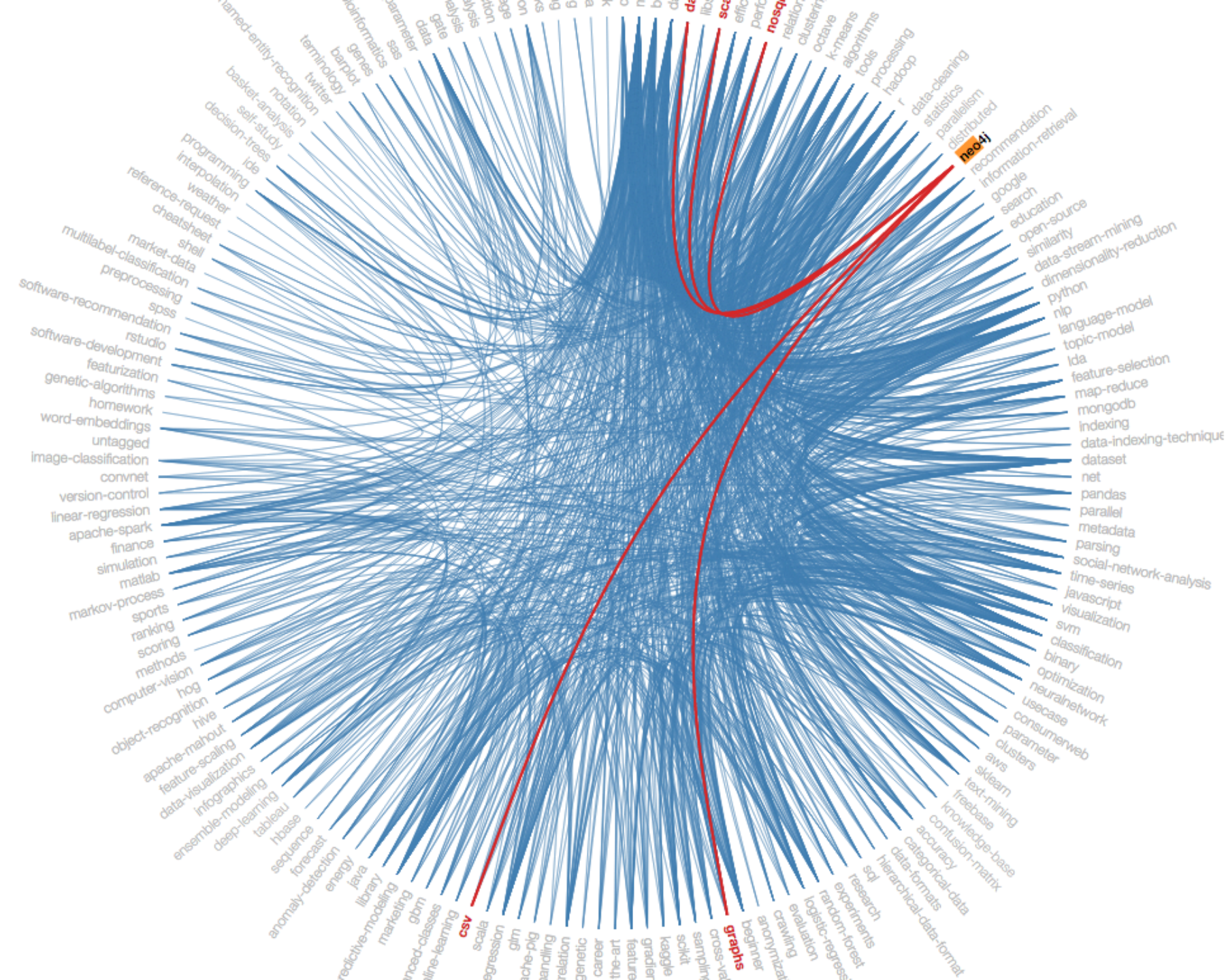
```
%load_ext cypher
# get all pairs of tags what appear in the same documents
query="""
MATCH (u:tag)<-[:hasTag]-(p:ask)-[:hasTag]->(q:tag)
Where toInt(q.ID)>toInt(u.ID)
RETURN u.TagName, q.TagName, count(*) AS tag_cooccurence
ORDER BY tag_cooccurence DESC
"""
tag_cooccurence = graph.cypher.execute(query)
tag_cooccurence
```

Out[17]:

	u.TagName	q.TagName	tag_cooccurence
1	machine-learning	classification	76
2	machine-learning	data-mining	69
3	machine-learning	r	48
4	machine-learning	python	34
5	machine-learning	neuralnetwork	34
6	machine-learning	statistics	30
7	machine-learning	algorithms	27
8	bigdata	data-mining	26
9	machine-learning	bigdata	26
10	machine-learning	predictive-modeling	26
11	machine-learning	dataset	25
12	machine-learning	nlp	25
13	machine-learning	clustering	23
14	bigdata	hadoop	22
15	data-mining	classification	22
16	data-mining	clustering	21
17	data-mining	dataset	21

Tag Force-directed graph

COLUMBIA UNIVERSITY


neo4j
graphs
csv
nosql
data
scala
data-cleaning
statistics
parallelism
distributed
recommendation
information-retrieval
google
search
education
open-source
similarity
data-stream-mining
dimensionality-reduction
python
nlp
language-model
topic-model
lda
feature-selection
map-reduce
mongodb
indexing
data-indexing-technique
dataset
net
pandas
parallel
metadata
parsing
social-network-analysis
time-series
javascript
visualization
svm
classification
binary
optimization
neuralnetwork
usecase
consumerweb
parameter
clusters
aws
sklearn
text-mining
freebase
knowledge-base
confusion-matrix
accuracy
categorical-data
data-formats
hierarchical-data-format
sql
research
experiments
random-forest
logistic-regression
evaluation
crawling
anonymization
beginner
cross-validation
sampling
scikit
kaggle
gradient-boosting
feature-engineering
the-art
career
genetic
relation
handling
sql-alchemy
postgresql
glm
regression
scala
imbalanced-classes
online-learning
gbm
marketing
predictive-modeling
library
java
energy
anomaly-detection
forecast
sequence
hbase
tableau
deep-learning
ensemble-modeling
infographics
data-visualization
feature-scaling
apache-mahout
hive
object-recognition
hog
computer-vision
methods
scoring
ranking
sports
markov-process
matlab
simulation
finance
apache-spark
linear-regression
version-control
convnet
image-classification
untagged
word-embeddings
homework
genetic-algorithms
featurization
software-development
rstudio
software-recommendation
spss
preprocessing
multilabel-classification
market-data
shell
cheatsheet
reference-request
weather
interpolation
programming
ide
decision-trees
self-study
basket-analysis
rotation
twitter
terminology
barplot
genes
bioinformatics
sas
data
gate
named-entity-recognition

sity

Neo4j Graph Database - a simple recommender

For those questions which do not have answer yet, return the users who answered questions with the same tags before.

```
query="""MATCH (q:ask)-[:hasTag]->(t:tag)<-[:hasTag]-(:ask)<-[:Answers]-()<-[:Answer]-(u:user)
Where not (q)<-[:Answer]-()
RETURN q.ID, u.ID, t.TagName"""
recommend = graph.cypher.execute(query)
recommend
```

Out[37]:

	q.ID	u.ID	t.TagName
1	16	51	machine-learning
2	61	51	machine-learning
3	115	51	machine-learning
4	116	51	machine-learning
5	159	51	machine-learning
6	169	51	machine-learning
7	196	51	machine-learning
8	211	51	machine-learning
9	215	51	machine-learning
10	265	51	machine-learning
11	266	51	machine-learning
12	310	51	machine-learning
13	319	51	machine-learning
14	323	51	machine-learning
15	326	51	machine-learning
16	369	51	machine-learning
17	406	51	machine-learning

Future Work

- Improve topic models, try models that can incorportate the name entity and meta data of each documents.
- Implement more classification algorithms and do a comparison between different methods
- Improve the recommender system

Questions?