SAND2017-9548C

LUCENE / SOLR REVOLUTION 2017

SEPTEMBER 12-15, 2017
LAS VEGAS, NV

An Intelligent, Personalized Information Retrieval Environment

Pengchu Zhang, John Herzer

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories

Agenda

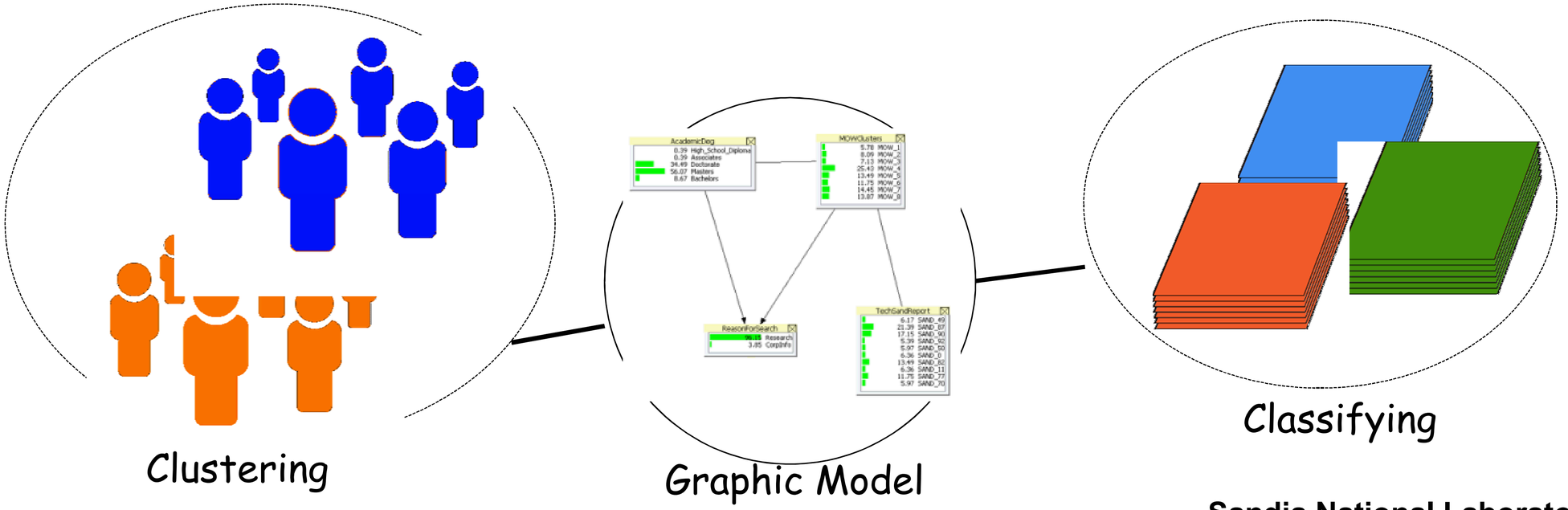
- Motivations
- Overview of Information Retrieval Models
- Unsupervised Learning for Clustering
- Supervised Learning for Classification
- Graphic Model for Predicting
- Integrating our models with Fusion
- Questions

Motivations:

- Deliver Information to Employees based on their specific needs:
  - Current search engines retrieve information with built-in algorithms solely based on the query;
  - In an organization, members of the work force have different needs for information even though they use the same/similar queries, e.g.;
    - *Managers/Leaders want to know the progress of on-going projects*
    - *Developers search for current/historical technologies*
    - *Financial analysts want to know the health of the budget*
- Leverage our analytics work along with Fusion's advanced capabilities

Solution: SPIRE

- The Sandia Personalized Information Retrieval Environment
- Matches customers with relevant content based on their personal attributes
- To accomplish this we need to cluster/classify customers, documents and build the probabilistic based predictive model



I. Profile MOWs with Personal Attributes

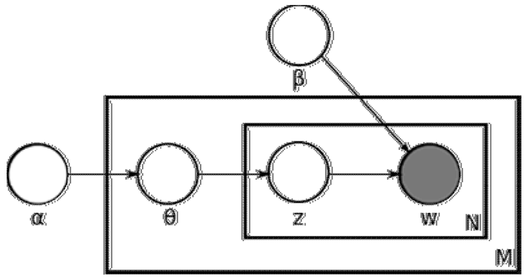
- Physical Attributes (easy to get but not very useful):
  - Education, years at job, job title…
  - Example: Sandia has about 1000 R&Ds with a job title of "Computer Science & Dev" but they are working in very different areas
- Documents generated by Employees (very useful):
  - Description of job assignments over the years;
  - Publications
  - Patents, awards, self-descriptions…

Unsupervised learning

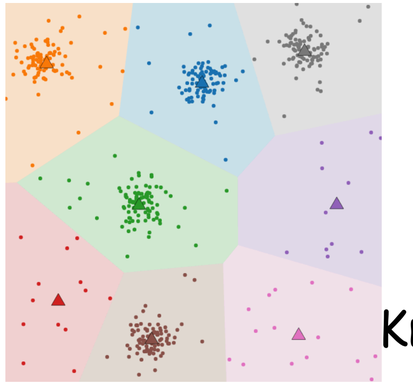
Clustering Users Based on Their Similarities

1. Job description
2. Publications, internal documents…

Users' data



LDA



Kmeans

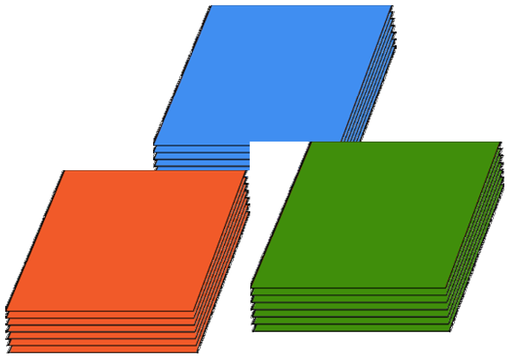
Clustering Algorithms

User Clusters

II. Document Classification

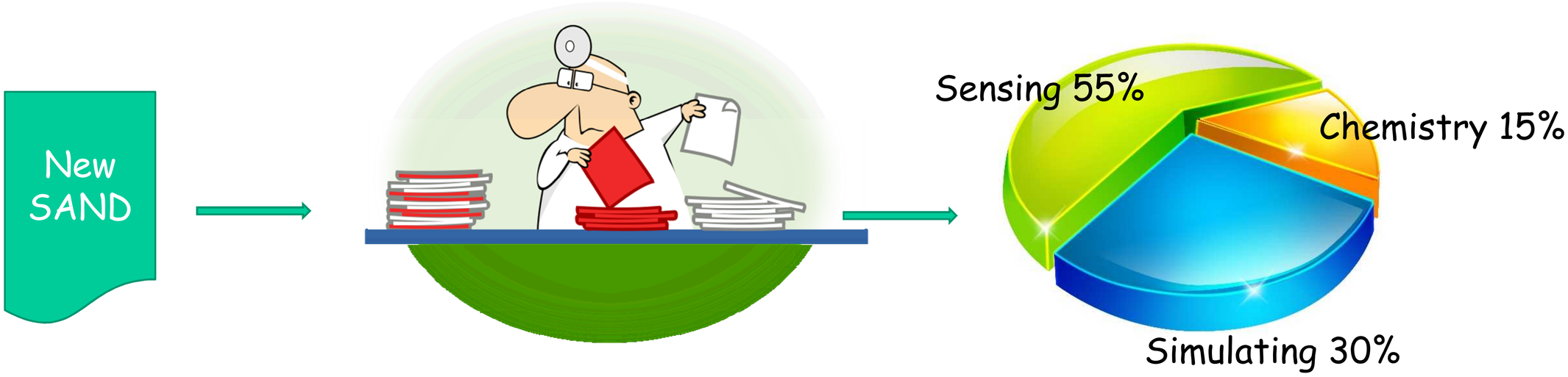
- There have been many models for classification, but:
  - Binary model such as sentiment classification: negative or positive
  - Statistical based:
    - *Need a lot of manually selected features as inputs, very expensive*
    - *Hard to scale up*

Supervised learning



Build SAND Report Classifiers with Deep Learning

1. Classify documents into proper classes
2. Recognize the document class in various formats
3. Recognize the distribution of possible classes in a document



New document

Trained Classifiers

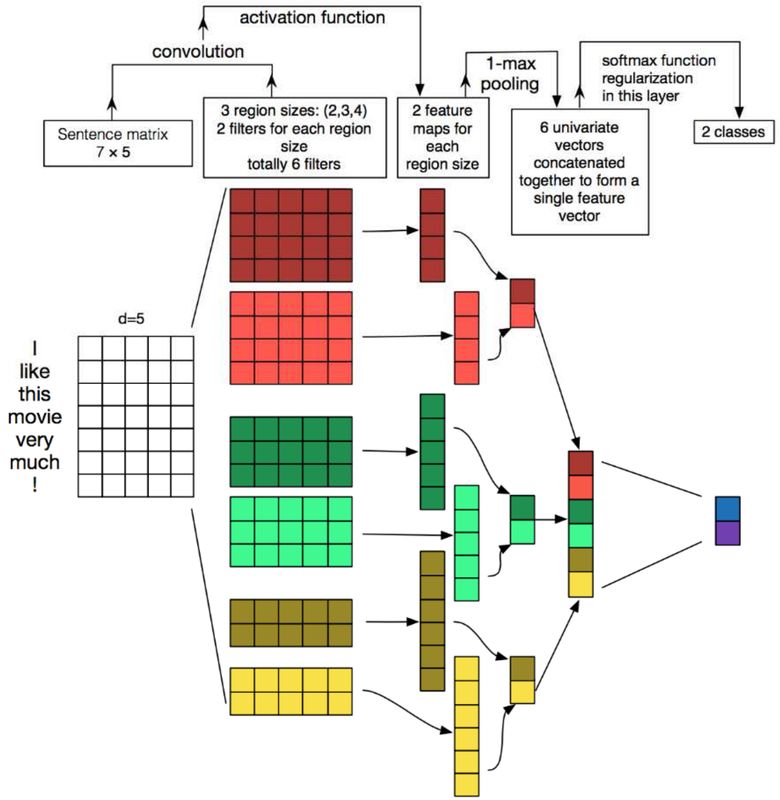
Distribution of Topics in a Document

Procedures

- Collected ~100,000 SAND reports over last 50 years
- Data cleaned and indexed with Lucene
- Built "Taxonomy" for Sandia Category Guide (SCG)
- Selected the highly ranked documents with SCG taxonomy terms/phrases as the training sets using Lucene
- Tokenized the terms in documents with a 200 dimension numerical vector
- Built a CNN network
  - 3 one dimension, 5 step convolutional layers with 128 feature maps
  - Three layers of maxpooling
  - ReLu and Softmax as the activation functions
  - Loss function = categorical_crossentropy
- Trained the network with various hyperparameters

CNN for Text Classification

http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/




```
print('Training model.')

# train a 1D convnet with global maxpooling
sequence_input = Input(shape=(MAX_SEQUENCE_LENGTH,),
dtype='int32')
embedded_sequences = embedding_layer(sequence_input)
x = Conv1D(128, 5, activation='relu')(embedded_sequences)
x = MaxPooling1D(5)(x)
x = Dropout(0.5)(x)
x = Conv1D(128, 5, activation='relu')(x)
x = Dropout(0.5)(x)
x = MaxPooling1D(5)(x)
x = Conv1D(128, 5, activation='relu')(x)
x = Dropout(0.5)(x)
x = MaxPooling1D(35)(x)
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
preds = Dense(len(labels_index), activation='softmax')(x)
model = Model(sequence_input, preds)
model.compile(loss='categorical_crossentropy',optimizer='rmsprop',
metrics=['acc'])
```

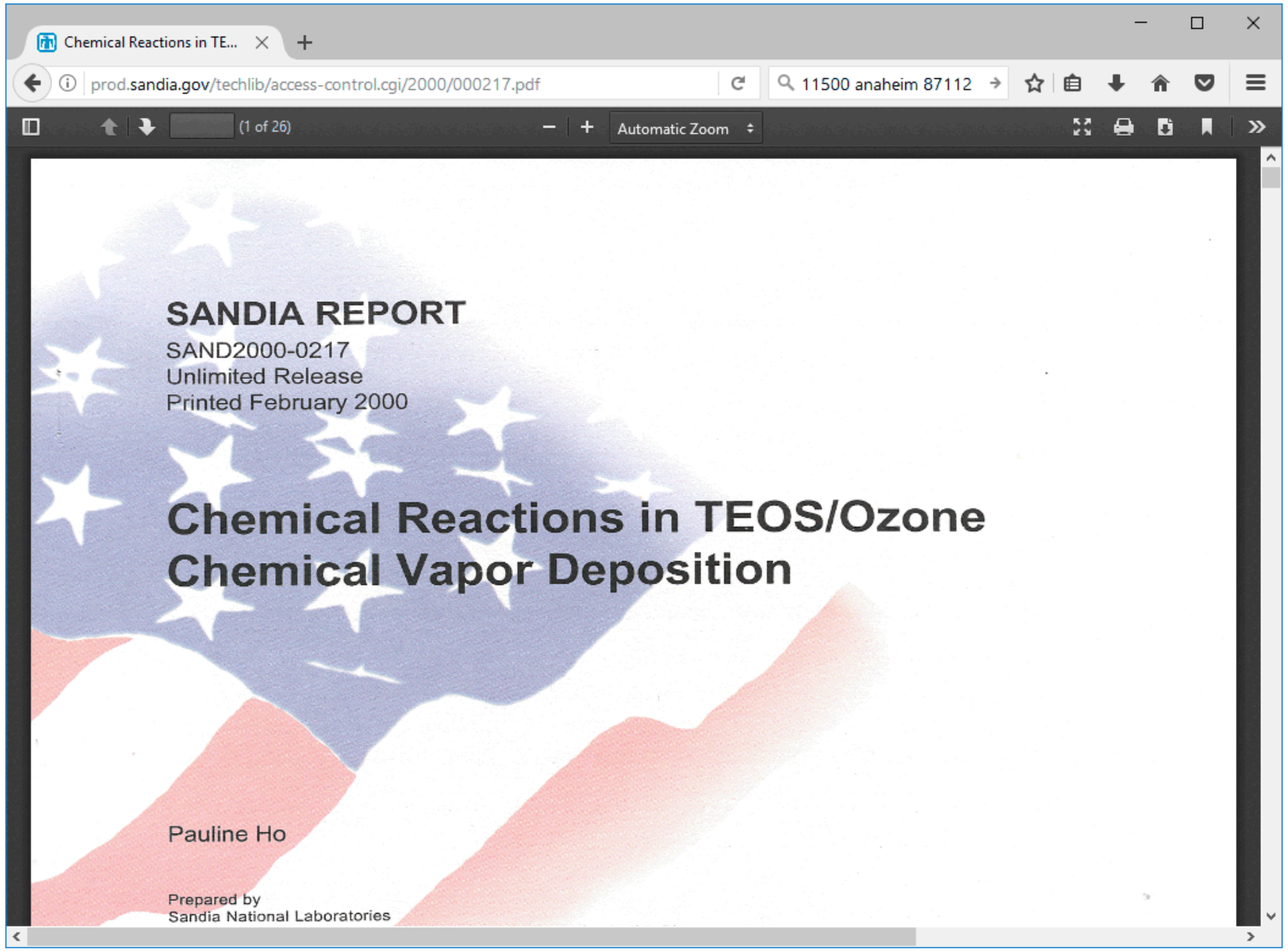
Modified from Keras example

Example of CNN Classifier Outputs

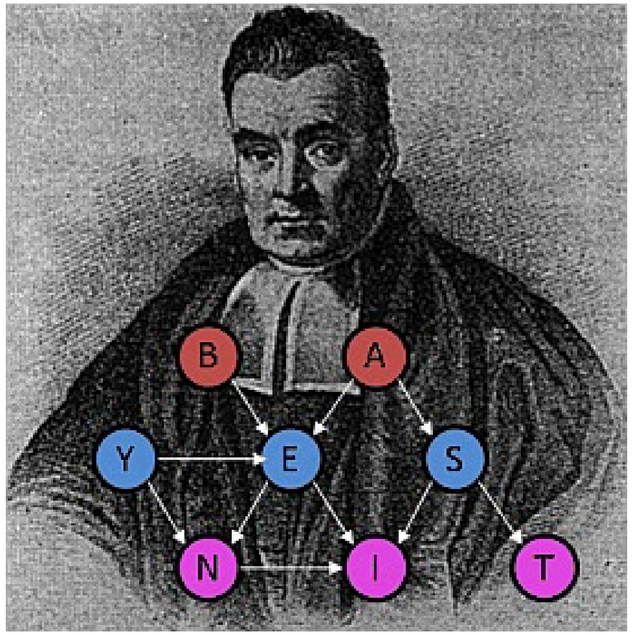
```
prediction = model.predict(data[146:150])
K=2
for p in range(0, prediction.shape[0]):

    a=np.array(prediction[p])
    b=np.argpartition(a, -K)[-K:]
    np.set_printoptions(precision=3)
    print(b, np.take(a, b)*100,'%', '\t', titles[p+146], '\t',labels_name[b[0]],'\t', labels_name[b[1]])
```

```
[37 1] [ 6.892 79.014] % SAND2000-0217.txt thermodynamics atmospheric science
[30 14] [ 6.755 91.043] % SAND2000-0218.txt particle physics electric and electronic
[16 31] [ 6.668 60.088] % SAND2000-0221.txt fluid mechanics plasma physics
[ 9 29] [ 29.797 52.055] % SAND2000-0222C.txt computer architecture optic
```

[37 1] [6.892 79.014] % SAND2000-0217.txt thermodynamics atmospheric science



III. Building Graphic Model for Prediction



Prior probability

Likelihood of the evidence of 'E'
If the hypothesis 'x' is true

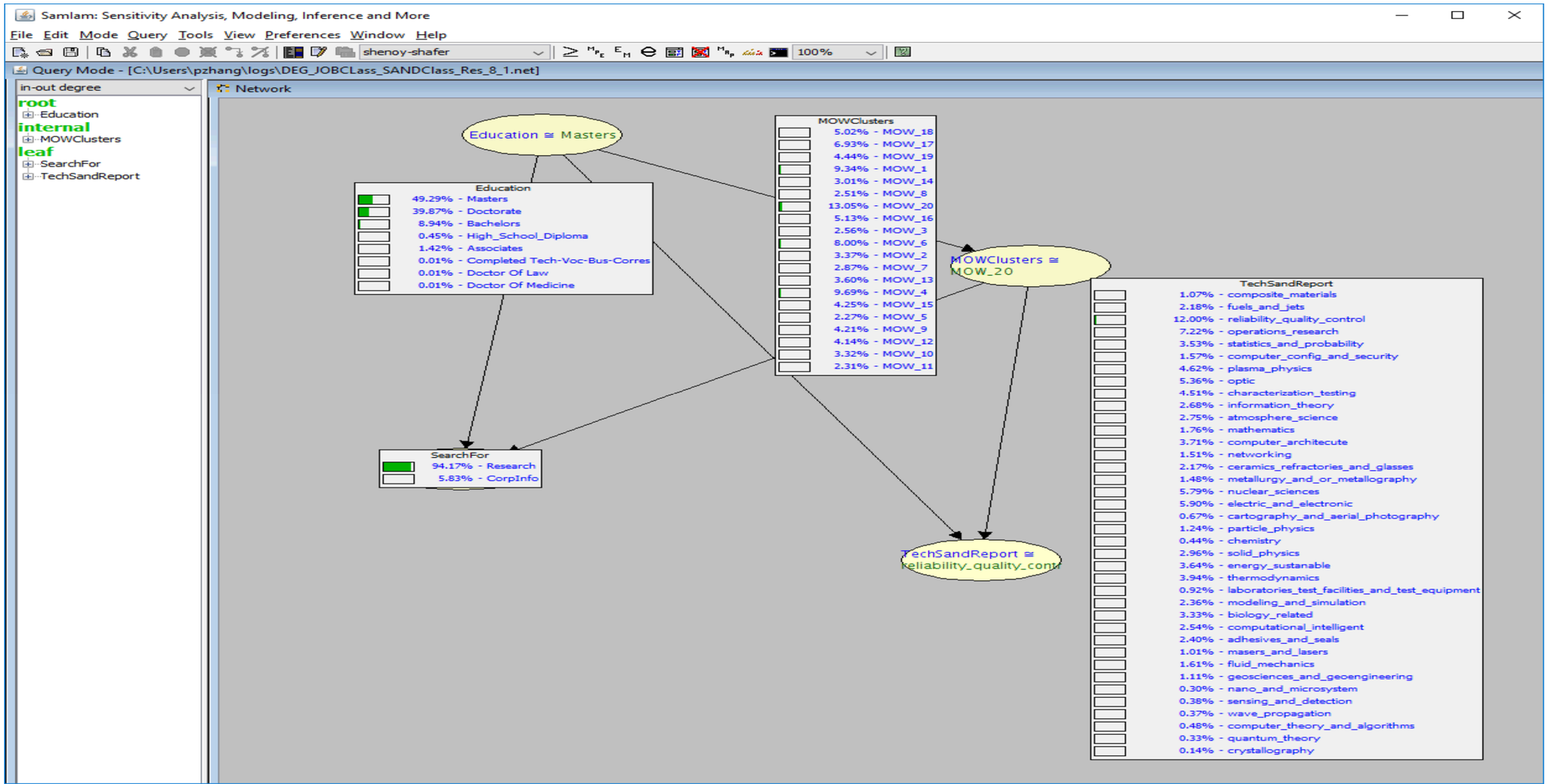
$$\mathrm{P}(x|E) = \frac{P(x) * P(E|x)}{P(E)}$$

Posterior probability of 'x'
given the evidence 'E'

Prior probability that
the evidence itself is true

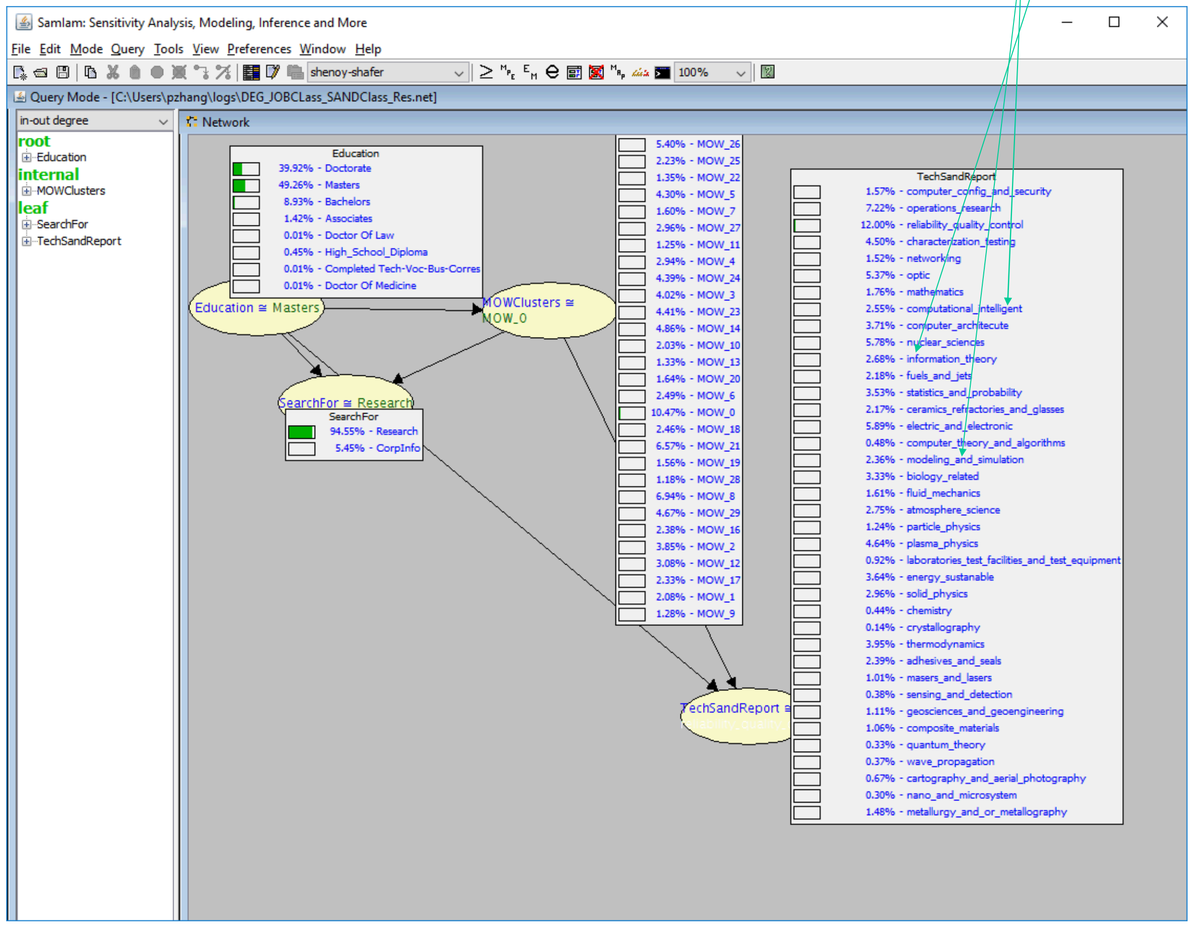
Build Graphic Models With the Conditional Probabilities

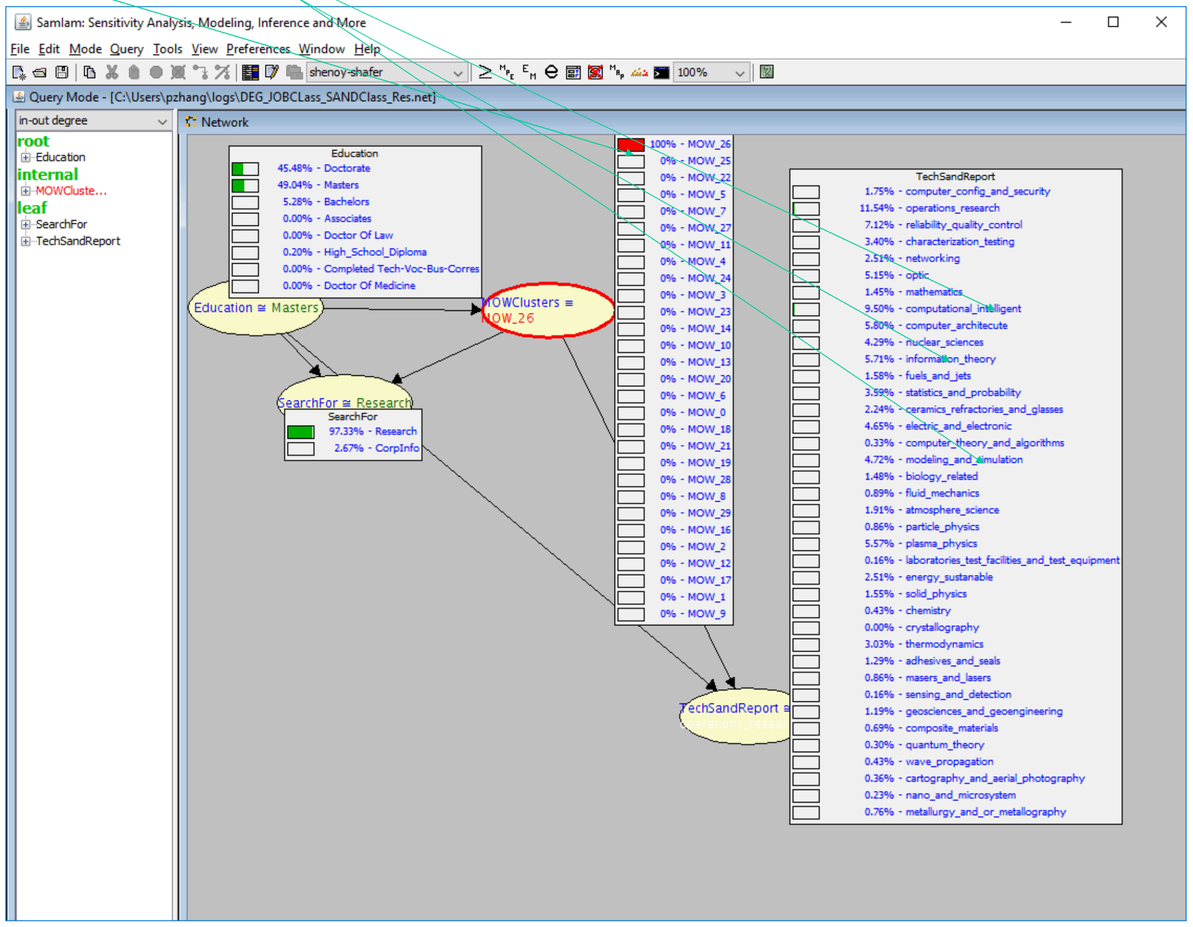
1. The behaviors of users on Solr based Enterprise Search Engine
2. Attributes of Clusters of Users
3. Build the conditional probability tables and create the Bayesian Network



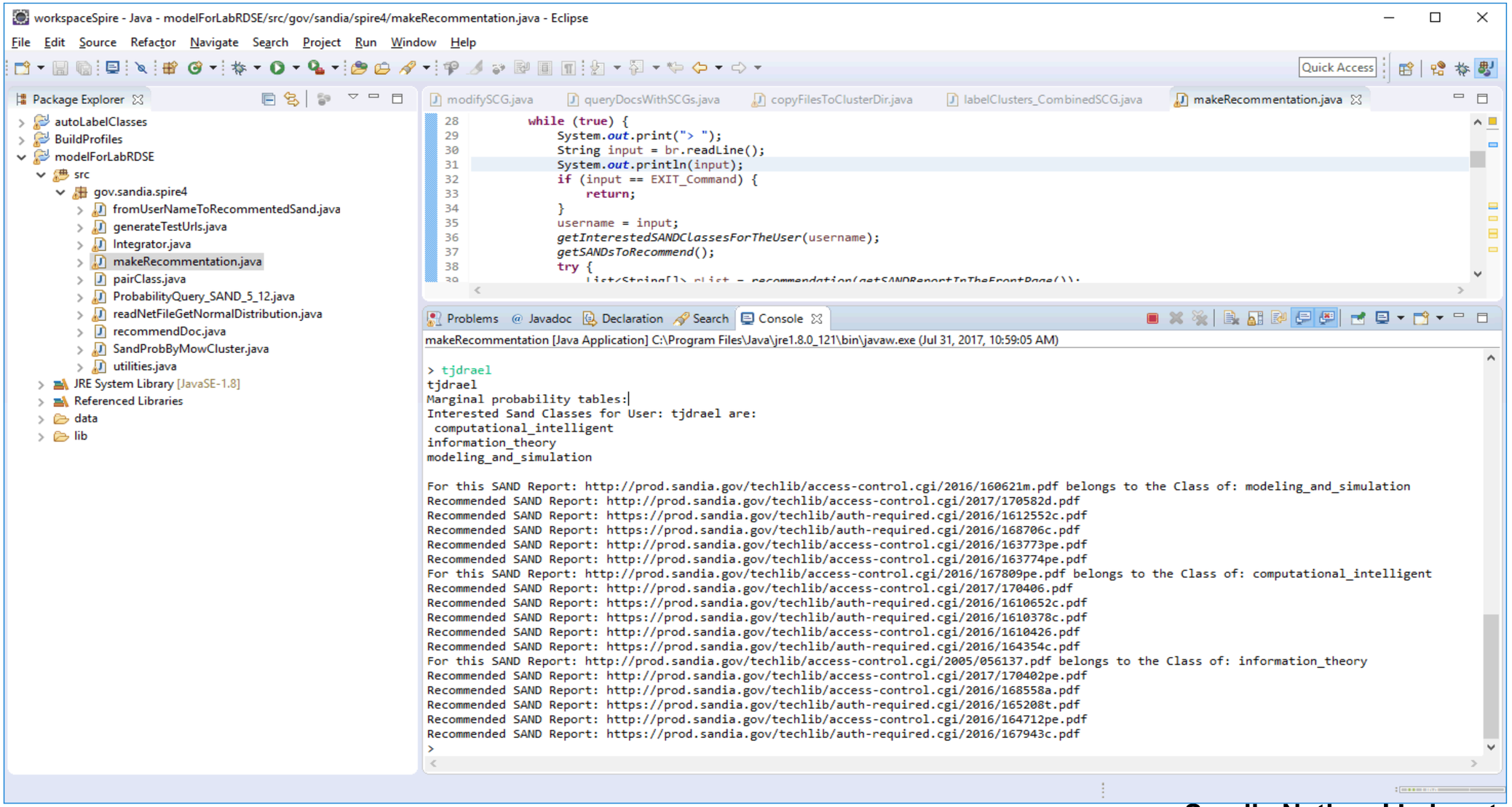
Predict Information Needed for a Given MOW

Computational Intelligent from 2.55 to 9.5%
Modeling and Simulation from 2.36 to 4.72%
Information Theory from 2.69 to 5.71%

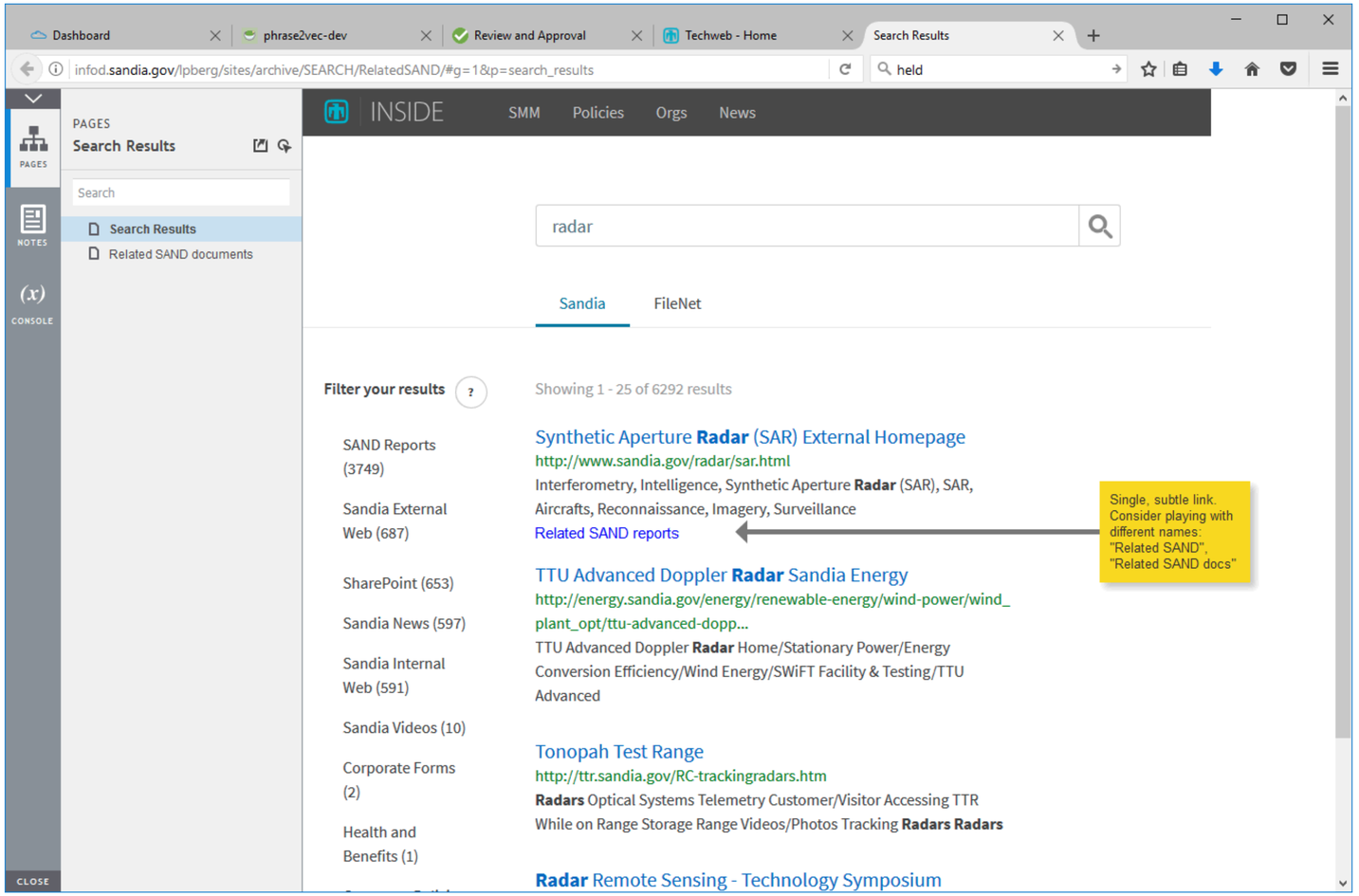




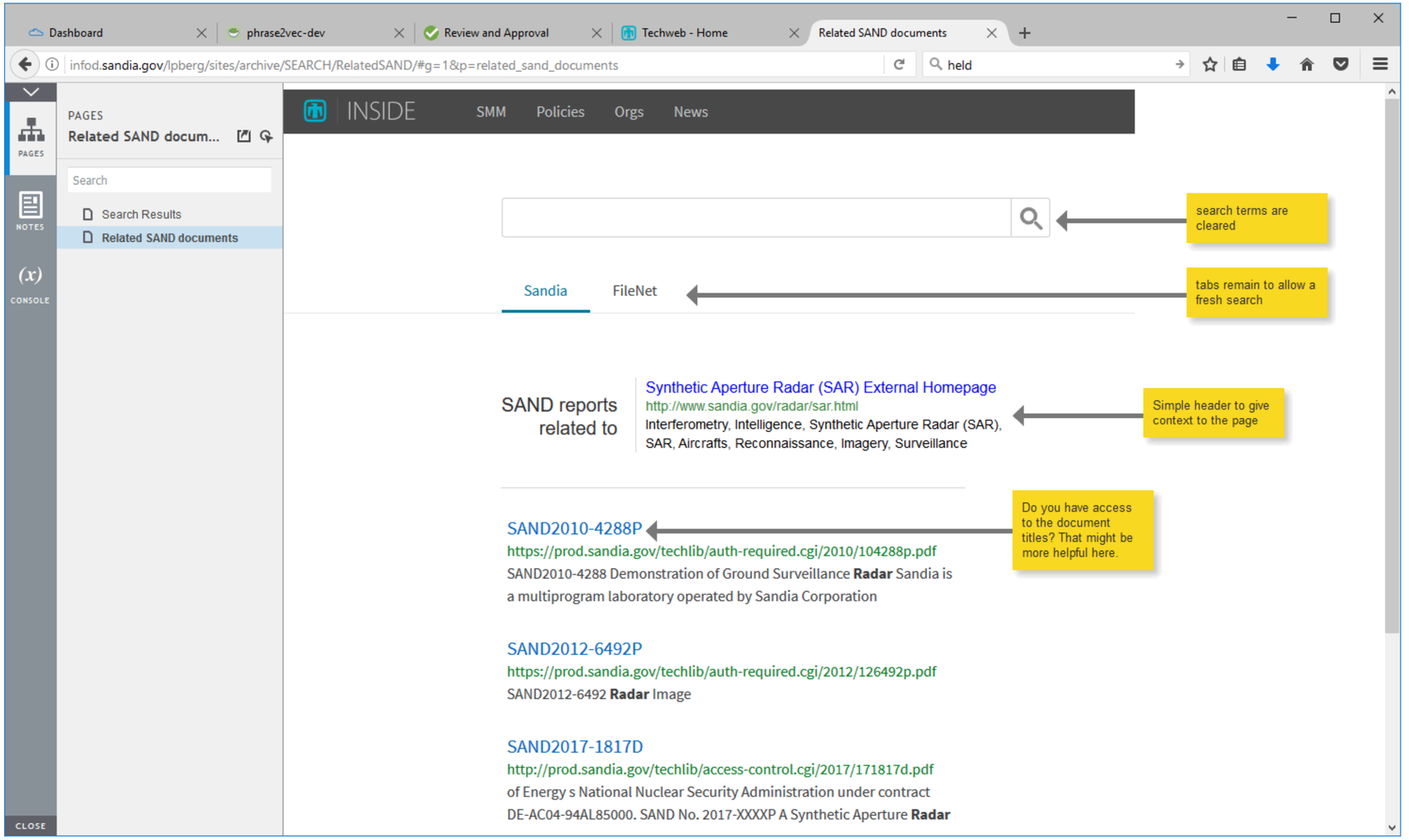
Example of Recommendation based on the Predictive Model



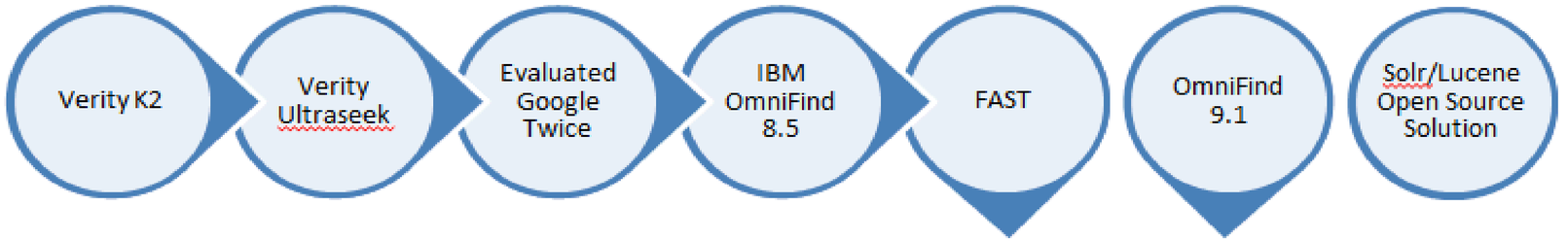
Implementation of Personalized Recommendations



Implementation of Personalized Recommendations



The Evolution of Enterprise Search at Sandia

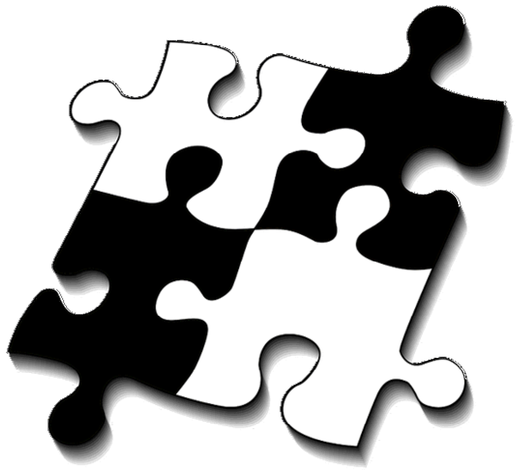


- We've gone through a number of search engine migrations
- Bad experiences with several COTS search engines
- As a result, selected Apache Solr, an open source platform
- Built our own search engine on top of Solr
- Developed custom client, crawlers, click tracking, click popularity boosting, faceting, and BestBets
- Selected Fusion based on a build vs buy decision process

Integrating our Analytics Capabilities with Fusion

Our department has ongoing analytics efforts including:

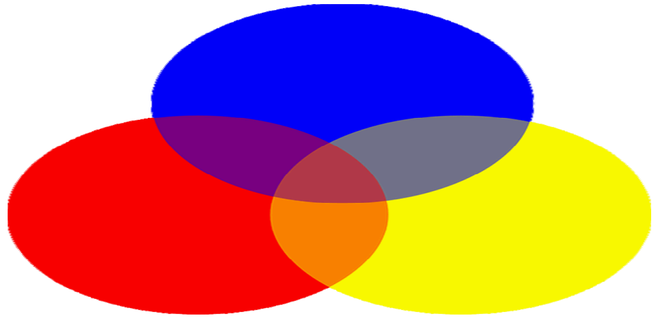
- Personalized recommendations using graphics models (SPIRE)
- Click popularity boosting (SVM in Matlab)
- Neural Network Recommender (using TensorFlow)
- PLUG recommender (collaborative filtering)
- Trending Query Terms



Handling overlapping capabilities

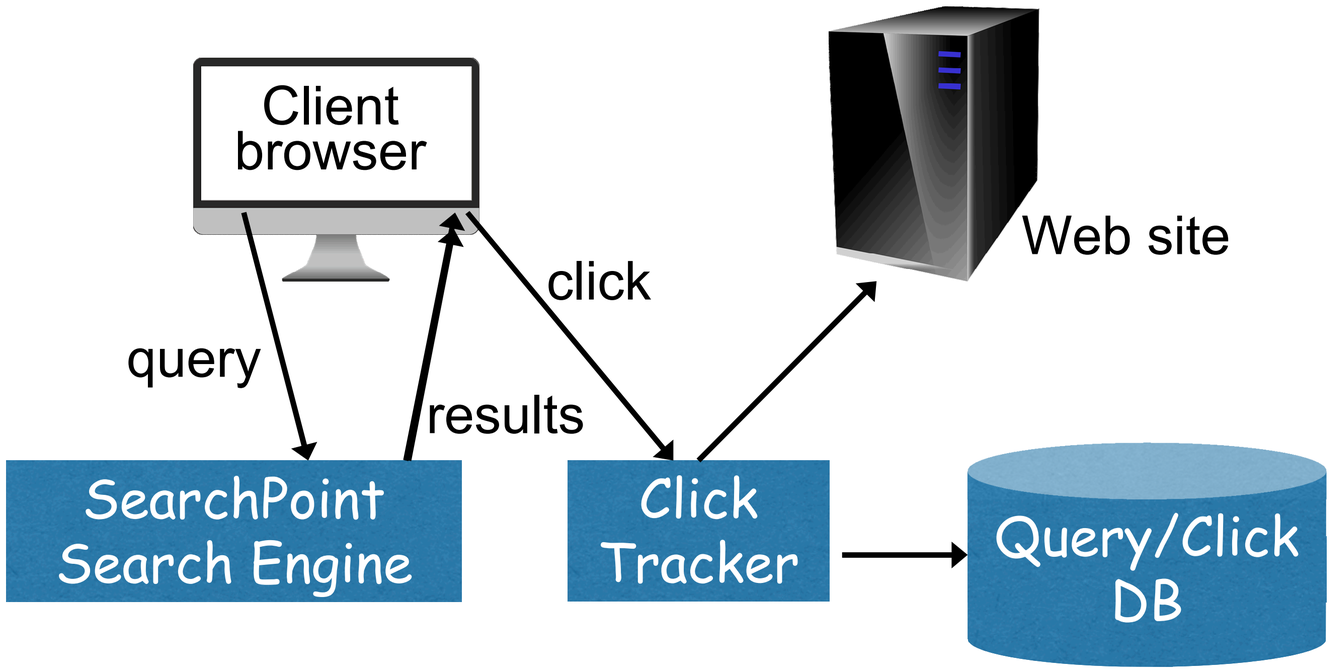
Our goal is to use as much capability from Fusion as possible
In areas where Fusion capabilities duplicate ours,
determine which capabilities perform best in our environment
- Use our Search Engine Configuration Evaluator (SECE)
- Utilize QA people to manually compare features

If our own capability works better, integrate it in with Fusion



Case Study: Integrating click tracking and Signals

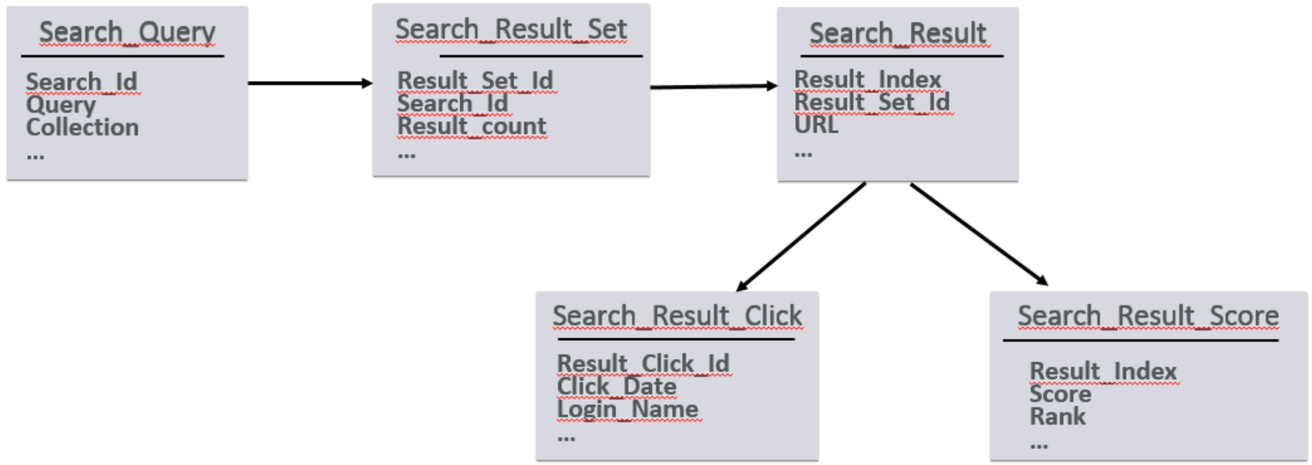
- The click tracking module stores click data in an Oracle DB
- Serves as primary source of data for many of our analytic models



Case Study: Integrating click tracking and Signals

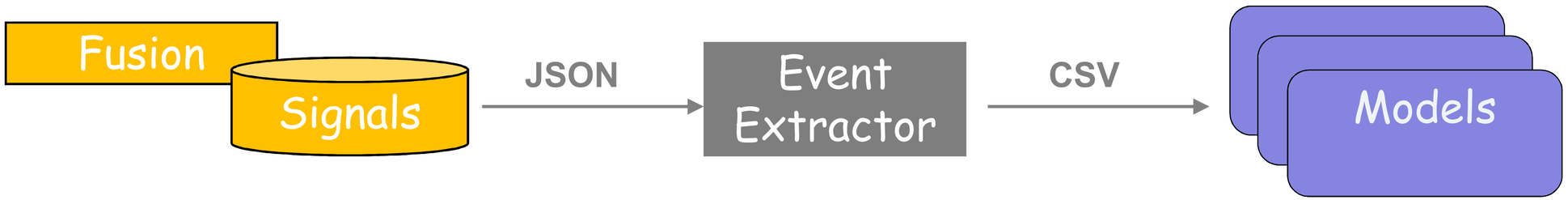
Our custom click tracking database tracks:

- Queries
- Results returned per query
- Result clicks
- Rank, Score …



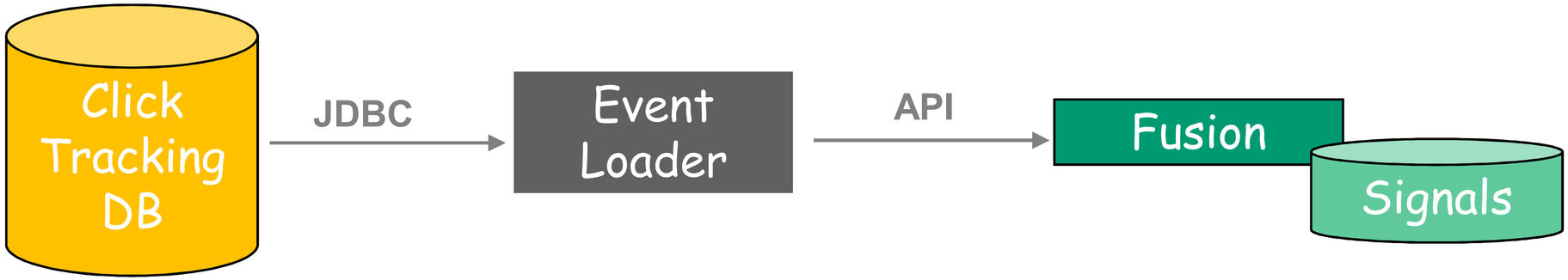
Integration option: Rely on Signals for event tracking

- Utilize the Fusion APIs to store click events directly as signals
- Maintain detailed and aggregated signals in Solr collections
- Set up jobs to export signals as needed by our local models
- May entail additional server costs for long term storage of signals



Integration option: Maintain source events in our DB

- Import events into Fusion as signals
- Aggregate signals and delete the detailed signal events
- Fusion models can utilize the imported signals directly
- Local models can still use the click tracking DB



Conclusion

- Application of combined machine learning and probabilistic models enables an enterprise search engine to intelligently satisfy users personal information needs
- Fusion's flexible environment provides a number of integration possibilities
- Using several integration points, we can leverage the analytics work we've been doing in the Fusion environment

Questions

Thank You

Sandia National Laboratories