Overview

❖ Background
❖ Algorithm
❖ Implementation
❖ Result and Evaluation
❖ Future Work
❖ Conclusion
Background

❖ Feature Extraction
  ➢ Words extracted as features

❖ Feature Selection
  ➢ Feature vectors generated from Apache Mahout

❖ Classification - enforce Solr search engine
  ➢ Naive Bayes
Algorithm

❖ Text Mining
  ➢ Naïve Bayes
  ➢ Random Forest
  ➢ Support Vector Machine (SVM)

Only serial implementation provided by Mahout
Algorithm - Naive Bayes

**Task:** Classify a new instance \( D \) based on a tuple of attribute values \( D = \langle x_1, x_2, \ldots, x_n \rangle \) into one of the classes \( c_j \in C \)

\[
c_{MAP} = \arg\max_{c_j \in C} P(c_j \mid x_1, x_2, \ldots, x_n)
\]

\[
= \arg\max_{c_j \in C} \frac{P(x_1, x_2, \ldots, x_n \mid c_j)P(c_j)}{P(x_1, x_2, \ldots, x_n)}
\]

\[
= \arg\max_{c_j \in C} P(x_1, x_2, \ldots, x_n \mid c_j)P(c_j)
\]

**MAP = Maximum Aposteriori Probability**
Implementation

- Data labeling
- Data source: cleaned data from reducing noise team.
- Collection: small collection, large collection.
- Each collection has tweets and webpages.
- Each has 8 topics (come from 8 teams).
- Each topics: 100 positive, 100 negative
- Train set: 80% samples
- Measurement of Accuracy: 5-fold Cross-Validation
Implementation

Apache Mahout Naive Bayes

Class label prediction for new data is not supported
Implementation

❖ Apache Mahout Naive Bayes

Learning Apache Mahout Classification

work for Hadoop 1.x
Implementation

❖ pangool: **100% compatible with different versions of Hadoop**
   ➢ Naive Bayes classification with MapReduce

❖ Jose Cadena from Hadoop Team
   ➢ made our program able to read in .avro file and write to .avro file
Example

Training: .txt file

<table>
<thead>
<tr>
<th>POSITIVE</th>
<th>NEGATIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>content of tweets or webpages</td>
<td>content of tweets or webpages</td>
</tr>
</tbody>
</table>

New data file stored in HDFS

All results (labeled new data) loaded into HBase by Hadoop team
## Result and Evaluation

### Tweets: Small Collection

<table>
<thead>
<tr>
<th>Team</th>
<th>Collection Topic</th>
<th>Average Accuracy</th>
<th>Filesize</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Plane Crash</td>
<td>92%</td>
<td>90M</td>
<td>13s</td>
</tr>
<tr>
<td>LDA</td>
<td>Suicide Bomb Attack</td>
<td>89%</td>
<td>13M</td>
<td>8.6s</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Jan. 25</td>
<td>73%</td>
<td>214M</td>
<td>18.6s</td>
</tr>
<tr>
<td>Solr</td>
<td>Election</td>
<td>86%</td>
<td>298M</td>
<td>34s</td>
</tr>
<tr>
<td>Reducing Noise</td>
<td>Charlie Hebdo</td>
<td>85%</td>
<td>64M</td>
<td>8.4s</td>
</tr>
<tr>
<td>NER</td>
<td>Storm</td>
<td>100%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Arabic*

*negative tweets come from other collections*
# Result and Evaluation

- **Tweets: Large Collection**

<table>
<thead>
<tr>
<th>Team</th>
<th>Collection Topic</th>
<th>Average Accuracy</th>
<th>Filesize</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification</td>
<td>Malaysia Airlines</td>
<td>78%</td>
<td>270M</td>
<td>32s</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Egypt</td>
<td>81%</td>
<td>3.1G</td>
<td>136s</td>
</tr>
<tr>
<td>Reducing Noise</td>
<td>Shooting</td>
<td>73%</td>
<td>7.4G</td>
<td>112s</td>
</tr>
<tr>
<td>LDA</td>
<td>Bomb</td>
<td>75%</td>
<td>5.8G</td>
<td>110s</td>
</tr>
</tbody>
</table>
## Result and Evaluation

### Webpages: Small Collection

<table>
<thead>
<tr>
<th>Team</th>
<th>Collection Topic</th>
<th>Average Accuracy</th>
<th>Filesize</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER</td>
<td>storm</td>
<td>83%</td>
<td>65M</td>
<td>4.7s</td>
</tr>
<tr>
<td>Classification</td>
<td>Plane Crash</td>
<td>87%</td>
<td>24M</td>
<td>9s</td>
</tr>
</tbody>
</table>
# Result and Evaluation

Webpages: Large Collection

<table>
<thead>
<tr>
<th>Team</th>
<th>Collection Topic</th>
<th>Average Accuracy</th>
<th>Filesize</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hadoop</td>
<td>Egypt</td>
<td>79%</td>
<td>140M</td>
<td>36s</td>
</tr>
<tr>
<td>Clustering</td>
<td>Diabetes</td>
<td>77%</td>
<td>305M</td>
<td>40s</td>
</tr>
</tbody>
</table>
Future Work

❖ For performance improvement
  ➢ Use larger training set
  ➢ Try using tf-idf value instead of word count in NaiveBayesGenerate.java
  ➢ Use more representative features
Conclusion

❖ Learned feature selection and classification algorithm on Apache Mahout

❖ Found a package to predict class label using Naive Bayes classifier generated from Apache Mahout
  ➢ Make it compatible with higher versions of Hadoop, instead of just 1.1.1

❖ Found a package for M/R Naive Bayes classifier
  ➢ Can be updated later for performance improvement
Thanks

❖ We give our thanks to
  ➢ Instructor: Dr. Fox

➢ TA: Mohamed Magdy and Sunshin Lee

➢ Classmates
  ■ Jose Cadena from Hadoop Team
  ■ Reducing Noise Team
We would like to thank our instructor Dr. Edward A. Fox, who brought us into this interesting project. We would like to thank our TAs, Mohamed Magdy and Sunshin Lee, for their continued support and valuable suggestions throughout the project. We would also give special thanks to Jose Cadena from the Hadoop team, who helped us with input and output formatting problems. Further, we thank the Reducing Noise team, which provided cleaned tweets and webpages for us to work on. Finally, thanks go to the support of NSF grant IIS - 1319578, III: Small: Integrated Digital Event Archiving and Library (IDEAL).
Questions