Structured Database and API to enable multilingual Search of different Political Actors with Performance Analysis of different approach

Subhasis Dutta, Jayprakash Rout, Priyanka Chadalavada, Sneha Ramesh Neranki, Chandrika Cherukuri , *The University of Texas at Dallas*

Abstract

The paper mainly focuses on mining the worldwide news press articles to identify the geographies that are sensitive to social unrest, agitations and campaigns. This is accomplished by working on multiple datasets to gather information on the various political actors regarding whom the society is alarmed and alerted. The motivation behind the project is to apply data resources and methods to help make data-driven decisions about foreign policy, civil war prevention, human rights policies, and the effects of other factors such as environmental or economic policies on these phenomena. The core of the project is to use data for the purposes of decision making which is not constrained by the language.

Index Terms

SPARK; CAMEO; JRC, Bablenet, Actor Ranking,

I. INTRODUCTION

Political event information is encoded from news reports. This can be achieved physically by people or by means of machine coding. Events are organized in view of an arrangement of word references for the activities and performing actors. The normal configuration to make these information is to decide "who did/said-what to whom". While the arrangement of activities or verbs is finite and this can be coordinated to a fixed ontology, the arrangement of source or target things is very substantial in political terms. To change news reports into valuable information to study global relations and civil conflict, distinguishing these political performing actors alongside their parts is vital.

The goal of our project is to ingest unstructured data from multiple data source and convert it into a coherent structured data stored and indexed in a database so that it can be searched by REST API calls. As the data sources are unstructured and incomplete we will be using map reduce to do different join operations. Also, to identify unidentified labels we will use different machine learning classification techniques to identify the language given a person's name.

Some of the objectives are:

1. Using Map Reduce Join a many-to-many relation between different data source

When a user searches for a person in a web interface we will retrieve the data, and visualize the data and expose APIs for other systems to consume. Based on user search for a person or organization we need to retrieve his/her name in different languages especially Spanish and Arabic. Along with this data, we are trying to combine data from CAMEO to get more details about that person. To combine data from CAMEO and JRC datasets, if we write the batch code to generate the data, it consumes a lot of time. So, we develop the Map Reduce relation between Cameo and JRC relations.

2. Using BableNet Knowledge Base identify political actors and their name variations

Using BableNet's data corpus try to identify the political actors and their name variants in different languages.

3. Using Machine Learning Identify the language of unidentified entity in the JRC Named Entity dataset

One of the main milestones of the implementation is to extract the different variants of the names of the political actors in multiple languages. We collate this information with the other available data sources to extract credentials and their political significance I am studying about how to utilize the Naive Bayes approach to perform text classification for multi class problems. My project would mainly encompass achieving accurate language classification using Naive Bayes N-gram approach.

One strategy for dealing with continuous data in naive Bayes classification would be to discretize the features and form distinct categories or to use a Gaussian kernel to calculate the class-conditional probabilities. Under the assumption that the probability distributions of the features follow a normal (Gaussian) distribution. Text classification is a typical case of categorical data; however, naive Bayes can also be used on continuous data.

Being an eager learner, naive Bayes classifiers are known to be relatively fast in classifying new instances. Eager learners are learning algorithms that learn a model from a training dataset as soon as the data becomes available. Once the model is learned, the training data does not have to be re-evaluated to make a new prediction. In case of eager learners, the computationally most expensive step is the model building step whereas the classification of new instances is relatively fast.

A lot of work is done on Text Classification using N-gram approach in Python using the nltk library.

4. User Interface to search the Named Entity dataset and display the name variation

An easy to use search interface which will make call to a REST API get the results and display it and cache the results.

II. Related Work

The Cameo data was created by researchers Department of Political Science, University of Kansas with funding from U.S. National Science Foundation. An automatic approach to the construction of Babel-Net was proposed by researchers from the department of Informatics, Sapienza University of Rome, Italy Cross-lingual Linking of Multi-Word Entities and their corresponding Acronyms was proposed by researchers from European Commission, Joint Research Centre, Ispra, Italy and Ecole Polytechnique Fédérale de Lausanne, Digital Humanities Laboratory, Lausanne, Switzerland

III. Data Source

JRC Dataset:

JRC-Named Entity Dataset is a highly multilingual named entity resource for person and organization names. It consists of large lists of names and their different spelling variants including across scripts in different languages - Latin, Greek, Arabic, Cyrillic, Japanese, Chinese, etc. JRC-Names is a technical resource that can be used to find names even if they are spelled differently, but it is also a useful ingredient for IT systems that process text, e.g. for text mining. JRC-Names contains the most important names of the EMM name database, i.e. those names that were found frequently or that were verified manually or found on Wikipedia.

CAMEO Dataset:

CAMEO (Conflict and Mediation Event Observations) Dataset is a framework designed to categorize all types of political interactions. CAMEO is built from multiple data sets sourced from government open data and proprietary individual level data. This enables us to build up an accurate and rounded picture of a person and their surrounding neighborhood. CAMEO covers over 35 countries, each with a country specific segmentation and a universal International CAMEO code.

BableNet Data:

BabelNet is both a multilingual encyclopedic dictionary, with lexicographic and encyclopedic coverage of terms, and a semantic network which connects concepts and named entities in a very large network of semantic relations, made up of about 14 million entries, called Babel synsets. Each Babel synset represents a given meaning and contains all the synonyms which express that meaning in a range of different languages.

dbPedia API – API interface to Wikipedia.

IV. Background

A. Apache-Spark [2]

Apache Spark provides programmers with an application programming interface centered on a data structure called the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way. It was limitations in the MapReduce which forces a distributed program: MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk. Spark's RDDs developed in response to cluster computing paradigm, linear dataflow structure on function as a working set for distributed programs that offers a (deliberately) restricted form of distributed shared memory.

The availability of RDDs facilitates the implementation of both iterative algorithms, that visit their dataset multiple times in a loop, and interactive/exploratory data analysis, i.e., the repeated database-style querying of data. The latency of such applications (compared to Apache Hadoop, a popular MapReduce implementation) may be reduced by several orders of magnitude. Among the class of iterative algorithms are the training algorithms for machine learning systems, which formed the initial impetus for developing Apache Spark.

Apache Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos. For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS), MapR File System (MapR-FS), Cassandra, OpenStack Swift, Amazon custom solution can be implemented. Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in such a scenario, Spark is run on a single machine with one executor per CPU core.

B. Spark Streaming [2]

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD transformations on those mini- batches of data. This design enables the same set of application code written for batch analytics to be used in streaming analytics, thus facilitating easy implementation of lambda architecture. However, this convenience comes with the penalty of latency equal to the mini-batch duration. Other streaming data engines that process event by event rather than in mini- batches include Storm and the streaming component ofFlink.Spark Streaming has support built-in to consume from Kafka, Flume, Twitter, ZeroMQ, Kinesis, and TCP/IP sockets.

D. Apache Zookeeper [4]

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. These kinds of services are used in some form or another by distributed

applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them, which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed.

E. MongoDB:

MongoDB an open source NOSQL document database. MongoDB stores data in the form of a BSON (Binary JSON-like). MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time

The document model maps to the objects in your application code, making data easy to work with Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data. MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use

F. Scala:

Scala is a pure-bred object-oriented language. Conceptually, every value is an object and every operation is a method-call. The language supports advanced component architectures through classes and traits.

Many traditional design patterns in other languages are already natively supported. For instance, singletons are supported through object definitions and visitors are supported through pattern matching. Using implicit classes, Scala even allows you to add new operations to existing classes, no matter whether they come from Scala or Java!
document. A set of documents forms a collection

G. Python:

An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly braces or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java.

H. MLlib Machine Learning Library [2]

Spark MLlib is a distributed machine learning framework on top of Spark Core that, due in large part to the distributed memory-based Spark architecture, is as much as nine times as fast as the disk-based implementation used by Apache Mahout(according to benchmarks done by the MLlib developers against the Alternating Least Squares (ALS) implementations, and before Mahout itself gained a Spark interface), and scales better than Vowpal Wabbit. Many common machine learning and statistical algorithms have been implemented and are shipped with MLlib which simplifies large scale machine learning pipelines, including:

- summary statistics, correlations, stratified
- sampling, hypothesis testing, random data generation
- classification and regression: support vector machines, logistic regression, linear regression, decision
- trees, naive Bayes classification
- collaborative filtering techniques including alternating least squares (ALS)
- cluster analysis methods including k-means, and Latent Dirichlet Allocation (LDA)
- dimensionality reduction techniques such as singular value decomposition (SVD), and principal component analysis (PCA)
- feature extraction and transformation functions
- optimization algorithms such as stochastic gradient descent, limited-memory BFGS (L-BFGS)

G. Unicodes for languages:

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. The Unicode Standard has been adopted by such industry leaders as Apple, HP, IBM, Just Systems, Microsoft, Oracle, SAP, Sun, Sybase, Unisys and many others. Unicode is required by modern standards and is the official way to implement ISO/IEC 10646. It is supported in many operating systems, all modern browsers, and many other products. The emergence of the Unicode Standard, and the availability of tools supporting it, are among the most significant recent global software technology trends. Unicode enables a single software product or a single website to be targeted across multiple platforms, languages and countries without re-engineering. It allows data to be transported through many different systems without corruption.

We observed that the languages have a unique range of unique codes.

Here is a small excerpt of the unicode ranges for some of the languages:

0600	06FF	Arabic
0530	058F	Armenian
0590	05FF	Hebrew
0600	06FF	Arabic
0700	074F	Syriac
0E00	0E7F	Thai
0E80	0EFF	Lao
0F00	0FFF	Tibetan
1000	109F	Myanmar
10A0	10FF	Georgian
1100	11FF	Hangul Jamo
1200	137F	Ethiopic

V. Approach & Architecture

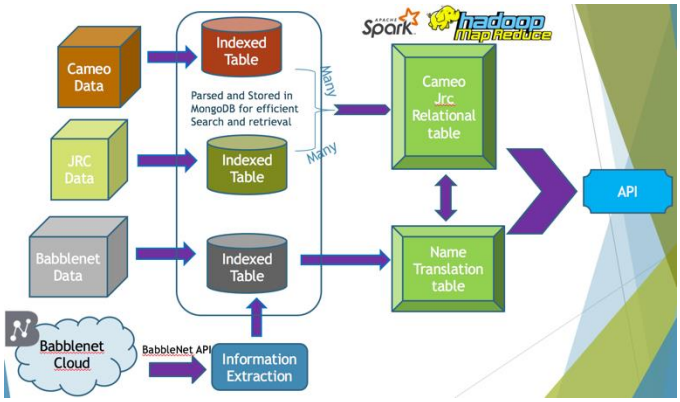


Figure 1. Model architecture

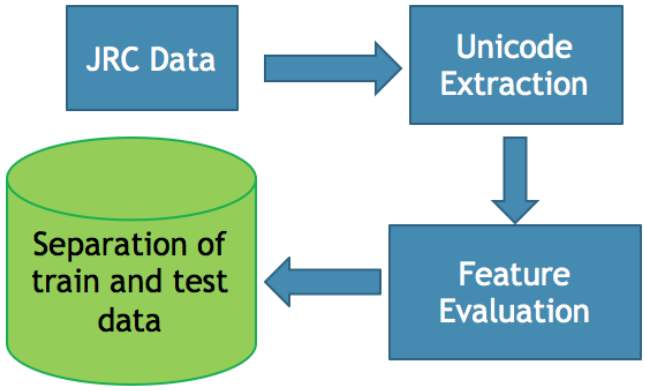


Figure 2.

VI. Implementation Details

Total records: 659895
No of training records: 71650
No of testing records: 588245

Missing values: The JRC Dataset is complete and had no missing values.

A. Feature selection for Machine learning classification:

The most important sub-tasks in pattern classification are feature extraction and selection; the three main criteria of good features are listed below:

It is intuitive and essential to construct a feature matrix to get good results. We need to think about how to best represent a text document as a feature vector. We need to identify two main types of features:

Salient. The features are important and meaningful with respect to the problem domain.

Discriminatory. The selected features bear enough information to distinguish well between patterns when used to train the classifier.

To find the measure of the spread of unicodes in each name we use metrics like Mean, Median, Highest, Lowest, Mode, Range. These statistics are commonly referred to as measures of central tendency.

Median - The median is the score that divides the distribution into halves; half of the scores are above the median and half are below it when the data are arranged in numerical order. The median is also referred to as the score at the 50th percentile in the distribution. The median location of N numbers can be found by the formula (N + 1) / 2. When N is an odd number, the formula yields a integer that represents the value in a numerically ordered distribution corresponding to the median location.

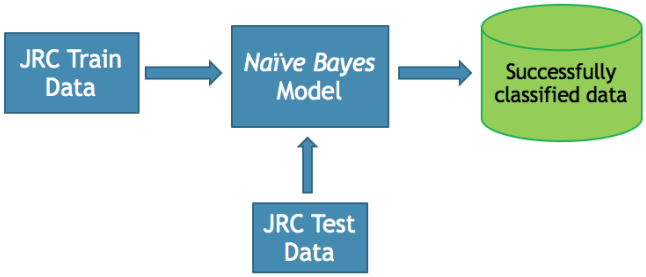


Figure 3.

Mean - The mean is the most common measure of central tendency and the one that can be mathematically manipulated. It is defined as the average of a distribution is equal to the SX / N. Simply, the mean is computed by summing all the scores in the distribution (SX) and dividing that sum by the total number of scores (N). The mean is the balance point in a distribution such that if you subtract each value in the distribution from the mean and sum these deviation scores, the result will be zero.

Highest: The highest unicode seen so far in the unicode set.

Lowest: The lowest unicode seen so far in the unicode set.

Range: The difference between the smallest value and the largest value in a dataset.

These are essentially chosen as the attributes of the feature vector.

B. CAMEO – JRC JOIN USING HADOOP MAP REDUCE

- Based on user search for a person or organization, the name in different languages especially Spanish and Arabic must be retrieved along with the data from CAMEO to get more details about that person. When a user searches for a person in a web interface, the retrieved data is to be visualized the data and expose APIs for other systems to consume.
- Cameo dataset contains the following fields:
  Id, record_type, cameo_title, compare_strings

- JRC dataset contains the following fields:
  Id, jrc_id, type, compare_strings
- The join operation on Cameo and JRC datasets is performed with compare_strings as the joining attribute.
- When the batch code is used to run the cameo and JRC datasets for performing the join operation, it consumes a lot of time. To simplify this, Hadoop Mapreduce is implemented to perform the join operation on the Cameo-JRC datasets. The direct join method on the joining key (which means for every compare_strings in JRC dataset, if there is a word in Cameo dataset) is performed. The outputs are taken by running the code in three different systems as the system configurations may vary.

System#	Time Taken in seconds
1	18.271
2	18.336
3	19.434

- The average time taken for the direct match of the Cameo-JRC join operation is 18.680 seconds.
- The edit distance algorithm is used to get the resulting join words with distance 0, 1 and 2. The edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other.
- Edit distances find applications in natural language processing, where automatic spelling correction can determine candidate corrections for a misspelled word by selecting words from a dictionary that have a low distance to the word in question. For each word in JRC dataset, the words with distance 0, 1 and 2 are printed in output along with the distances.

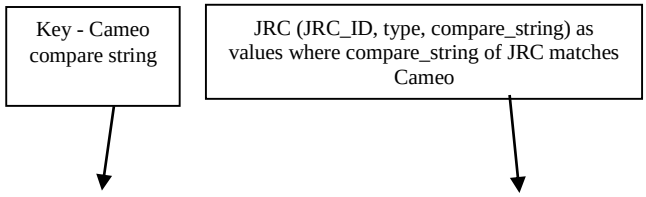
C. CAMEO – JRC JOIN USING SPARK

The join on JRC and Cameo datasets on the attribute "compare_string" provides a resultant dataset where a compare_string value in JRC correctly matches a compare_string value in Cameo.

A tuple in Cameo dataset contains the following fields:
(ID, record_type, cameo_title, compare_strings)

The number of records in Cameo dataset are:
A tuple in JRC dataset contains the following fields:
(ID, JRC_ID, type, compare_strings)
A sample tuple in the join RDD is as follows:

A sample tuple in the join RDD is as follows:




```
(homero+arellano,(58ce47e1fea250209c0a2d
51|Cameo.Phoenix.Countries.actors|ECUADO
R_|homero+arellano,58ce4dbffea2502114760
   b16|645673|PERSON|homero+arellano))
```

The outputs are taken by running the code in three different systems as the system configurations may vary.

System#	Time Taken in seconds
1	4.109709
2	4.157123
3	4.238234

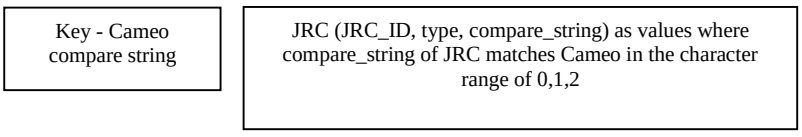
The direct join takes 4.16835 secs on an average when run in Spark cluster.

EDIT DISTANCE

The edit distance algorithm is used to get the resulting join words with distance 0, 1 and 2. The edit distance is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other.

For each compare string in the Cameo dataset, the values of the corresponding compare string with distance 0, 1 and 2 in the JRC dataset are printed in output.

A sample tuple in the join RDD is as follows:



(fisher,
<58ce4dbefea2502114743d24|176548|ORGANIZATION|it
er><58ce4dbefea2502114748513|196373|ORGANIZATIO
N|fnsea><58ce4dbefea2502114748513|196373|ORGANIZ

ATION|fnsea><58ce4dbefea250211474afe3|23150|ORGANIZATION|disney><58ce4dbefea250211474b45c|235557|PERSON|li+shen><58ce4dbefea250211474ea60|283990|ORGANIZATION|fide><58ce4dc0fea250211476c824|86269|ORGANIZATION|pfizer><58ce4dc0fea250211477175e|95258|PERSON|ali+sher>)

VII. EXPERIMENTS AND OBSERVATIONS

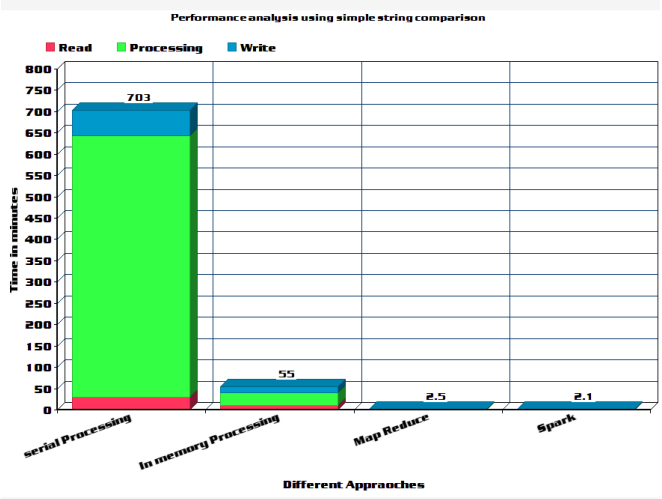


Figure 4. Performance analysis using simple string comparison

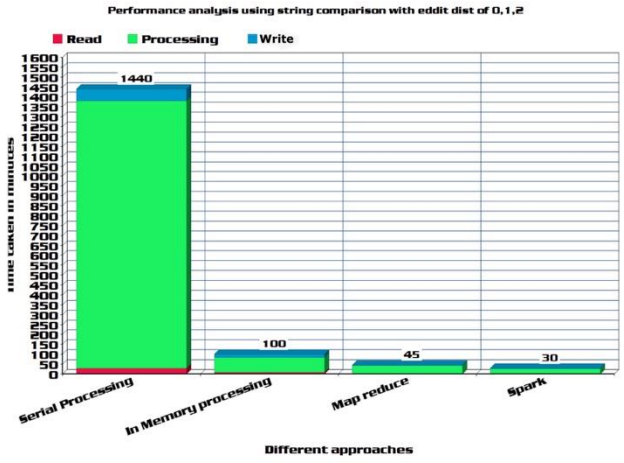


Figure 5. Performance analysis using string comparison with edit distance of 0, 1, 2

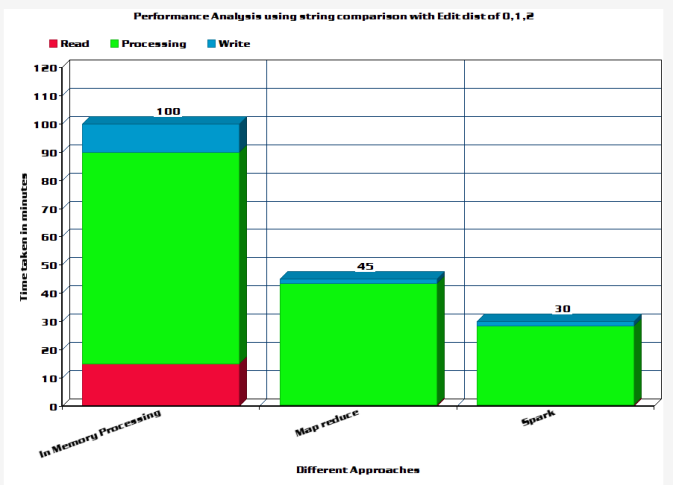


Figure 6. Performance analysis using string comparison with edit distance 0, 1, 2

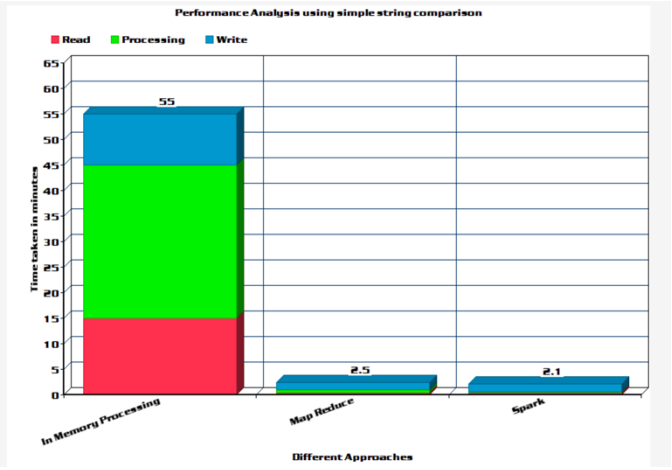


Figure 7. Performance analysis using simple string comparison

A. Experiments

Preprocessing of the data:

For each record in the dataset we build the feature vectors. As a first step, we calculate the unicodes of each character in the string name. We build a hashmap with name as the key and the unicode values of each character in the name string as the value set.

Secondly, we use this hashmap to calculate the features like mean, median, high, low and range. The output of this program is a hashmap with key as name string and value as a set of calculated features.

We generate another hashmap with name as key again and identified or unidentified language as the value.

Then we merge the two hashmaps and write the resultant to the file.

Based on whether the language is identified or not the resultant file is further divided into test, train and validation set. This file will be fed as input for the classification algorithms. We chose Naïve Bayes algorithm to learn the data, build the model and then predict.

Naive Bayes is a simple multiclass classification algorithm with the assumption of independence between every pair of features. Naive Bayes can be trained very efficiently. Within a single pass to the training data, it computes the conditional probability distribution of each feature given label, and then it applies Bayes' theorem to compute the conditional probability distribution of label given an observation and use it for prediction. We could run the algorithm for 250000 records.

B. Observations

Experiment #	Test Data Size	Time Taken in seconds	Accuracy
1	50000	132.06	75.10
2	50000	135.42	77.01
3	50000	133.97	73.38
4	50000	132.05	76.04
5	50000	134.37	73.24

As per the results obtained, we can see that the average accuracy is 74.97%.

Ways to improve the accuracy:

As we had relatively limited set of train data the model built may not very accurate. We observed that for some of the languages or classes the data does not have enough support or records. So, when records for such classes are encountered during testing there is a good chance of misclassifications. Also, the validation set size was relatively smaller in size which may have impacted the numbers.

Also, the features we settled for were mean median and range which gives a central distribution. There is a good chance that these features may not accurately describe the class. The other features we examined during the process and found promising were standard deviation, frequency and quantiles. This could give a good start to look at the Unicode distribution in a different perspective.

C. Conclusions

VIII. Shortcomings & Future Work

- Allow along with name, get more different attributes.
- Dynamically update the database with new updates.
- Use SVM, K-means for language detection in JRC dataset.
- Use Lucene indexed Babble-net dataset to extract more data.
- Using cross validation like k-fold method for better results. We can explore other features like statistic quantiles like standard deviation, variance, inter quantile range, quartiles for feature selection. We can try using more powerful machine learning techniques like ensemble methods for building the model.

Acknowledgment

We wish to acknowledge Professor for the Big Data Management and Analytics class, Dr. **Latifur Khan** for approving the project proposal and providing the guidelines for the project. Also, we acknowledge the teaching assistants for the class, **M Solaimani & Zhuoyi Wangfor** for resolution of queries and validating the demo run of the project.

References

[1] https://en.wikipedia.org/wiki/Apache_Spark

[2] https://en.wikipedia.org/wiki/Apache_Kafka

[3] https://zookeeper.apache.org/

[4] http://nverma-tech-blog.blogspot.com/2015/10/apache-kafka-quick-start-on-windows.html

[5] http://www.unicode.org/roadmaps/bmp/

[6] http://babelnet.org/about

[7] http://jrgraphix.net/research/unicode_blocks.php

[8] Discover New Actors In Politics: A Framework To Recommend Political Actors With Role In Real-time

[9] http://www.abs.gov.au/websitedbs/a3121120.nsf/home/statistical+language+-+measures+of+spread

[10] https://www.scala-lang.org/what-is-scala.html

[11] https://www.mongodb.com/what-is-mongodb/