Extraction of questions from the Internet using a Machine Learning approach

Alok Saw	B090924CS
Jerrin Shaji George	B090437CS
Shubhangam Agrawal	B090904CS
Stein Astor Fernandez	B090006CS

Guide : Dr. Priya Chandran

Contents

1. **Introduction**
2. Work Done in S7
3. Work Done in S8
4. Future Work

Introduction

There is a massive amount of data available on the internet.

Extracting only the **relevant sections** of data has become important.

This would result in massive savings of both time and effort.

For both faculty and students, obtaining a list of questions pertaining to a topic would be useful.

It is tedious to search for various query permutations and manually visit each page and then find such questions.

Would be highly useful if this could be automated.

Problem Statement

Given input a topic **t**,

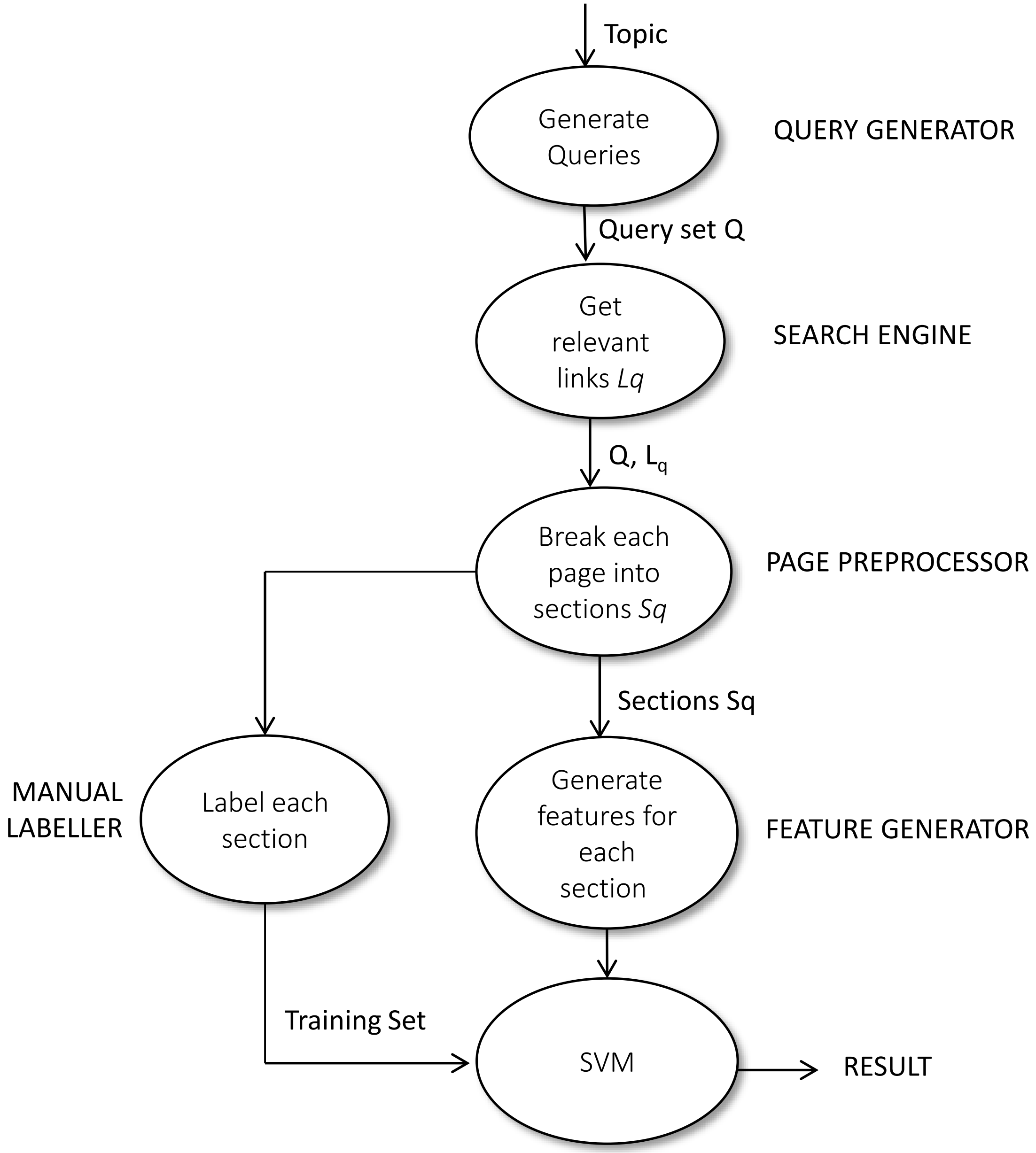
Search the internet for pages possibly containing questions related to **t**

Output a set **R** comprising of relevant questions extracted from these pages.

Contents

1. Introduction
2. **Work Done in S7**
3. Work Done in S8
4. Future Work

Architecture



Features

Features for relevance of text[1] -

I. Word Rank Based Features
II. Bigram Rank Based Features
III. Coverage of Top Ranked Tokens

Proposed features to determine if text is a question -

IV. Coverage of Interrogative Indicators
V. Absence of Specific Keywords

I. Word Rank Based Features

Rank – position in the list if the words were ordered by frequency of occurrence in S_q

Each word rank is a feature.

Value – Frequency of word in section

Dimensionality reduction by bucketing.

II. Bigram Rank Based Features

A ***bigram*** is defined to be two consecutive words occurring in a section[1].

Find the top *n* frequently occurring bigrams in $\boldsymbol{S_q}$.

The feature vector is computed in a manner similar to the previous one.

“Machine learning” may be more important than “machine” and “learning” separately.

III. Coverage of Top Ranked Tokens

Relevance may also be determined by the coverage of top ranked words in the section[1].

Words are ranked on the basis of frequency of occurrence.

For each section, find the coverage of the top ranked words per bucket.

Example –

- Top 5 tokens - `learning', `machine', `data', `access', `database'.
- Section contains `learning' and `data', feature value is 2

IV. Coverage of Interrogative Indicators

This feature indicates the likelihood of the section being a question.

The coverage of a predefined list of interrogative indicators is taken as the value for this feature.

Examples of interrogative indicators –

- “what”
- “why”
- “?”

V. Absence of Specific Keywords

Questions are generally devoid of specific keywords such as "because" and "yes".

The coverage of a predefined list of such keywords is taken as a negative value for this feature.

Contents

1. Introduction
2. Work Done in S7
3. **Work Done in S8**
4. Conclusion and Future Work

Work done in S8

Class Design

Data Design

- Data Structure Design
- Data File Design

Config Design

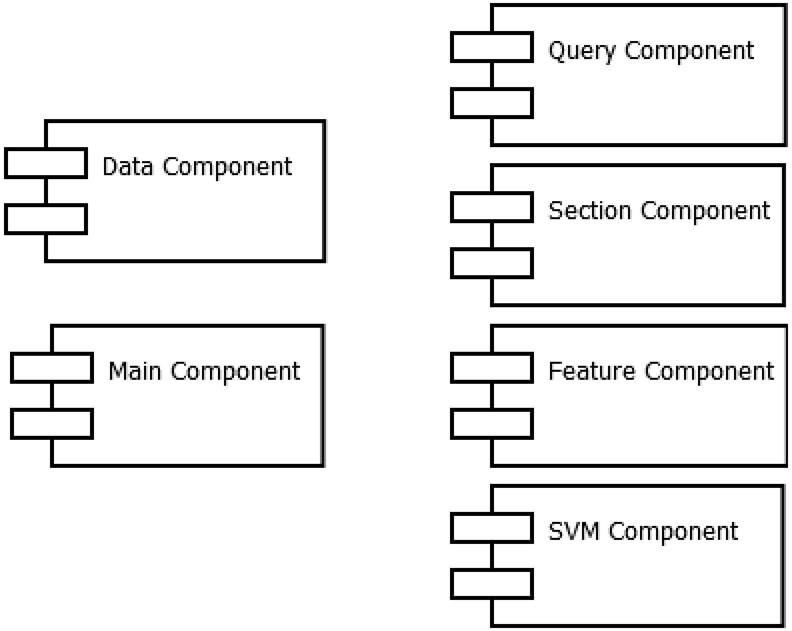
SVM Identification

Implementation

Analysis

Class Design

The overall design consists of six components designed for high cohesion and low coupling.



Main Component

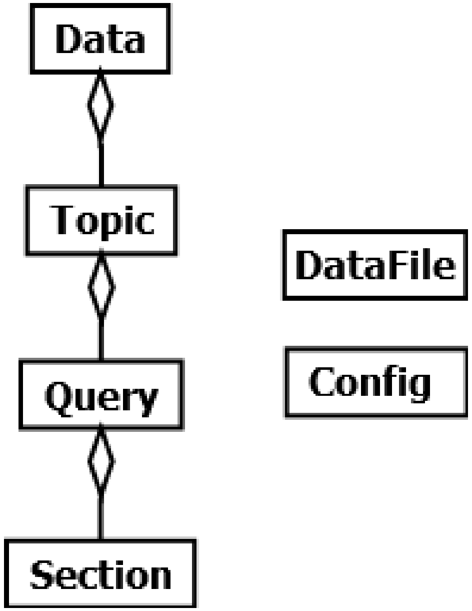
This component is responsible for calling all other components and finally generating the list of questions as output.

TrainingDataGenerator

RealRun

Data Component

The data component stores the overall data and its classes provided interfaces to access and modify this data.



Query Component

The query component is responsible for taking the input topic, then generating the requisite subtopics and queries related to the same.

TopicListGenerator

QueryGenerator

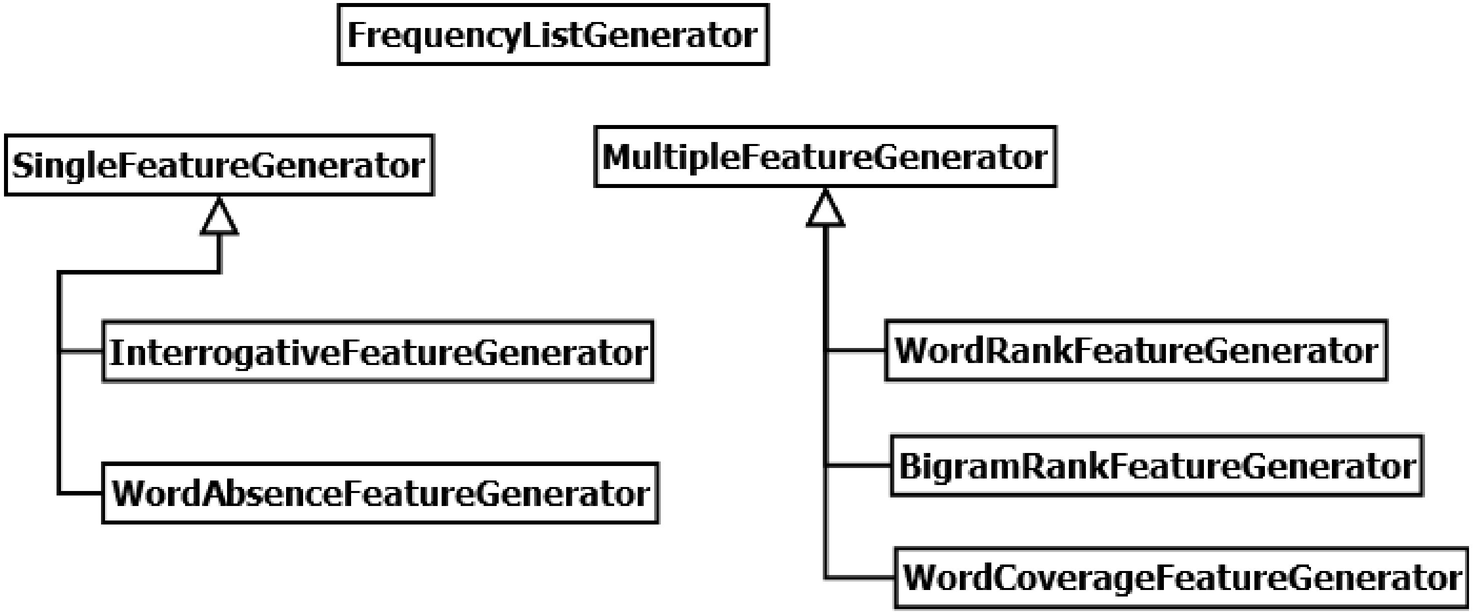
Section Component

The section component processes each query and generates a list of Section objects for each query.

LinkGenerator
HTMLParser
using JSoup
SectionGenerator

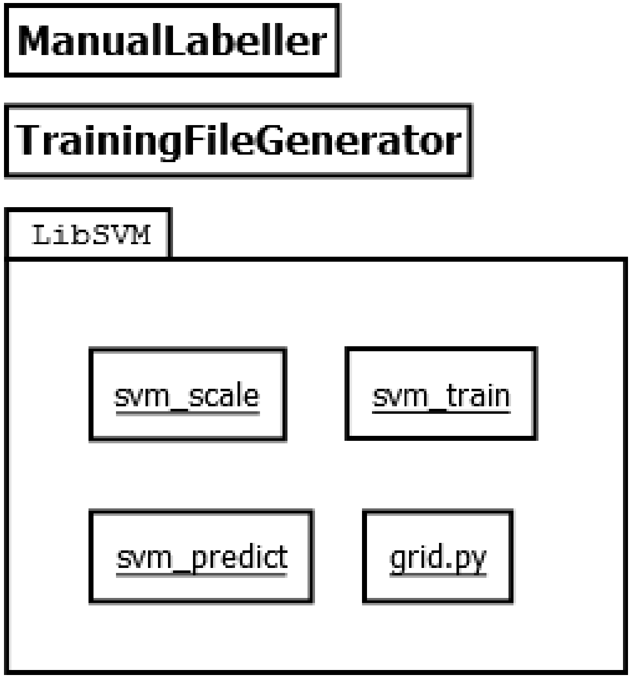
Feature Component

The feature component has classes to generate all the requisite features for the sections.


FrequencyListGenerator
SingleFeatureGenerator
MultipleFeatureGenerator
InterrogativeFeatureGenerator
WordAbsenceFeatureGenerator
WordRankFeatureGenerator
BigramRankFeatureGenerator
WordCoverageFeatureGenerator

SVM Component

Consists of all the classes which are related to the machine learning aspect of the program.



ManualLabeller

TrainingFileGenerator

LibSVM

svm_scale

svm_train

svm_predict

grid.py

Work done in S8

- Class Design
- **Data Design**
  - Data Structure Design
  - Data File Design
- Config Design
- SVM Identification
- Implementation
- Analysis

Data Structure Design

n–ary tree

Suitable for component-based architecture

Easy modeling of data

Extensible

Quick access

Group data in a functional manner

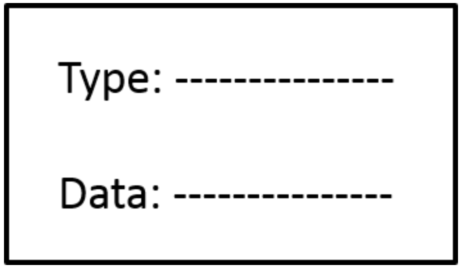
Node Design

The node will have the following attributes:

Type: The type of node

Data: The corresponding data

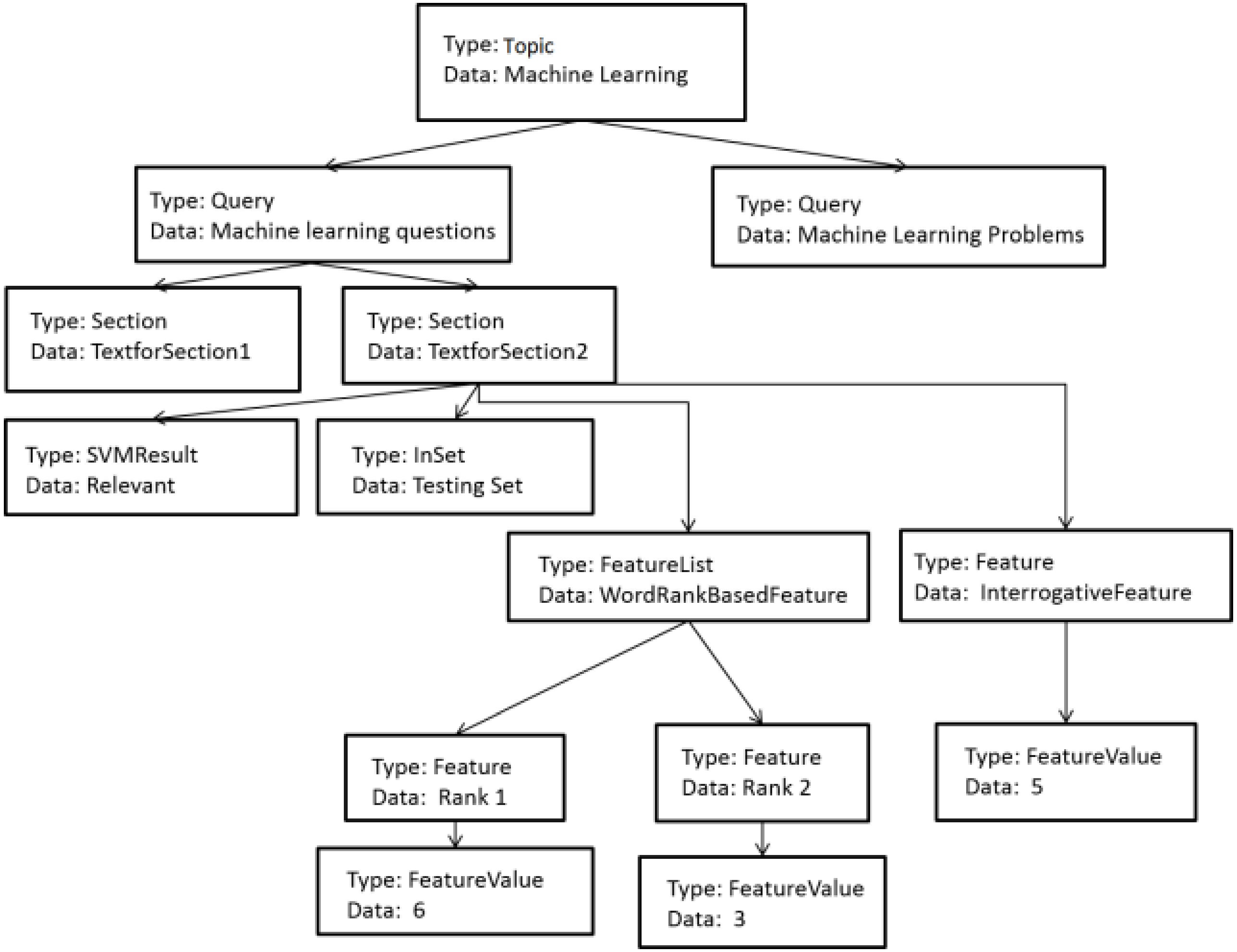
ChildList: List of child nodes



Type: ---------------

Data: ---------------

Tree Design



Type: Topic
Data: Machine Learning

Type: Query
Data: Machine learning questions

Type: Query
Data: Machine Learning Problems

Type: Section
Data: TextforSection1

Type: Section
Data: TextforSection2

Type: SVMResult
Data: Relevant

Type: InSet
Data: Testing Set

Type: FeatureList
Data: WordRankBasedFeature

Type: Feature
Data: InterrogativeFeature

Type: Feature
Data: Rank 1

Type: Feature
Data: Rank 2

Type: FeatureValue
Data: 5

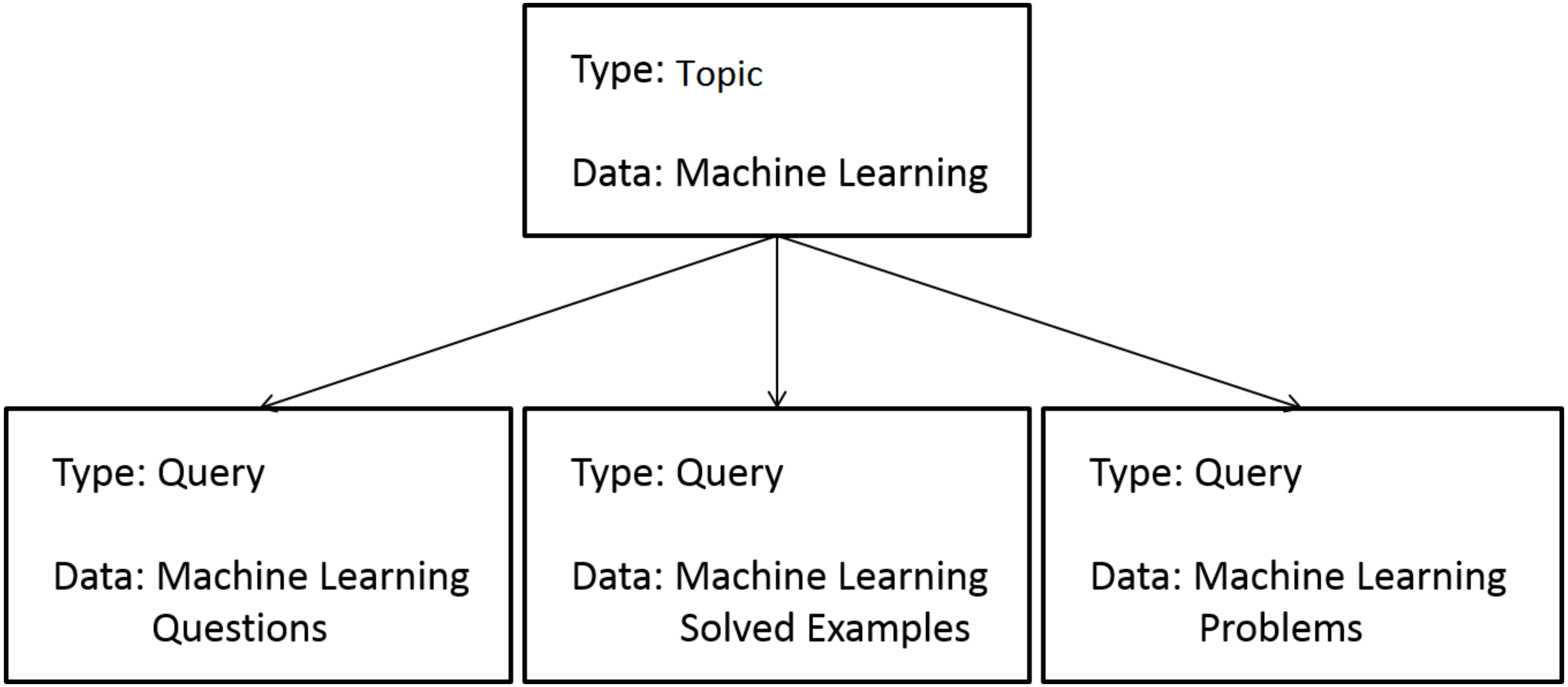
Type: FeatureValue
Data: 6

Type: FeatureValue
Data: 3

Data Tree Life-Cycle

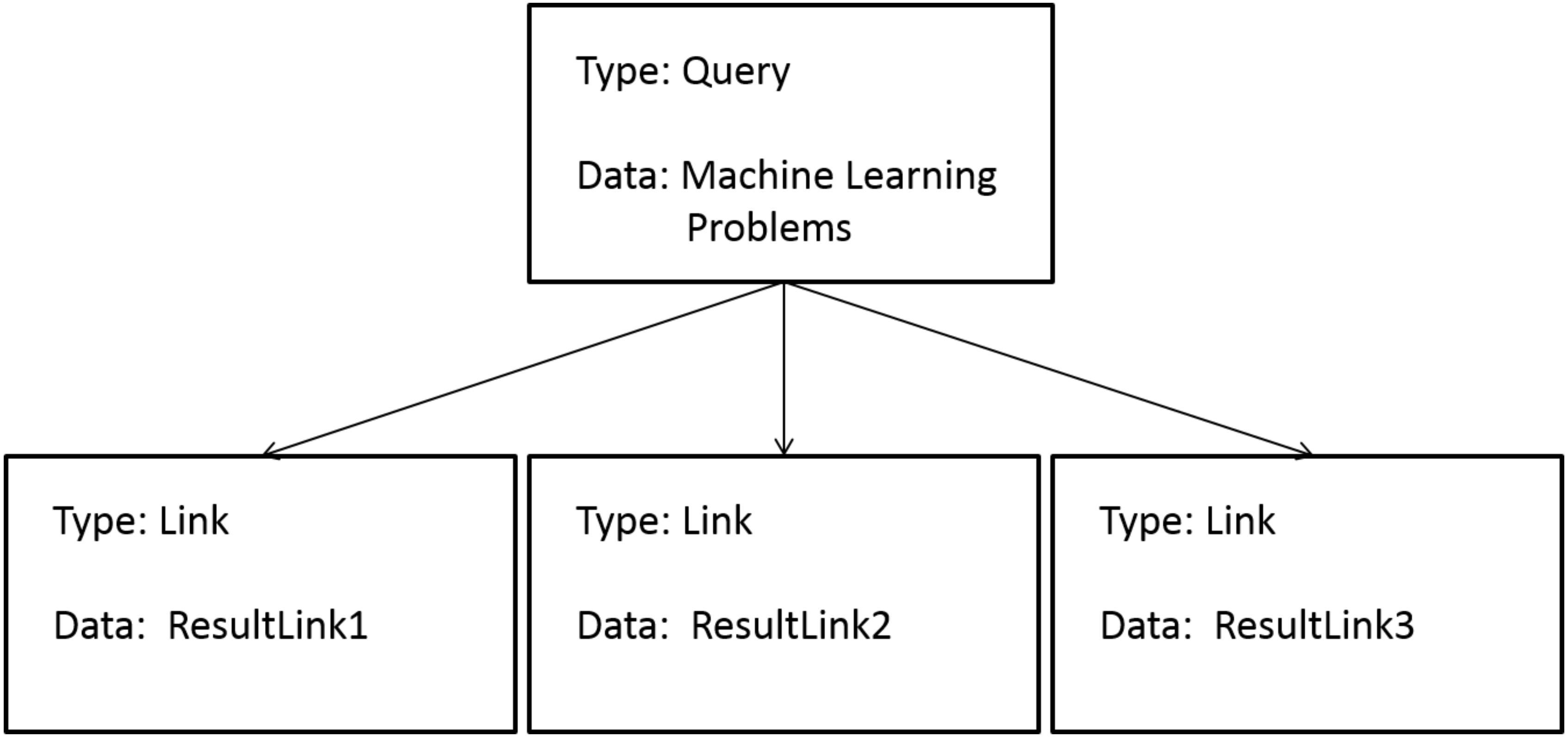
Query Generation

The Query Generator generates a set of queries to search for from the given search topic.


Type: Topic
Data: Machine Learning
Type: Query
Data: Machine Learning Questions
Type: Query
Data: Machine Learning Solved Examples
Type: Query
Data: Machine Learning Problems

Link Generation

For each of the queries, a set of links corresponding to the search results for that query is generated.



Type: Query

Data: Machine Learning Problems

Type: Link

Data: ResultLink1

Type: Link

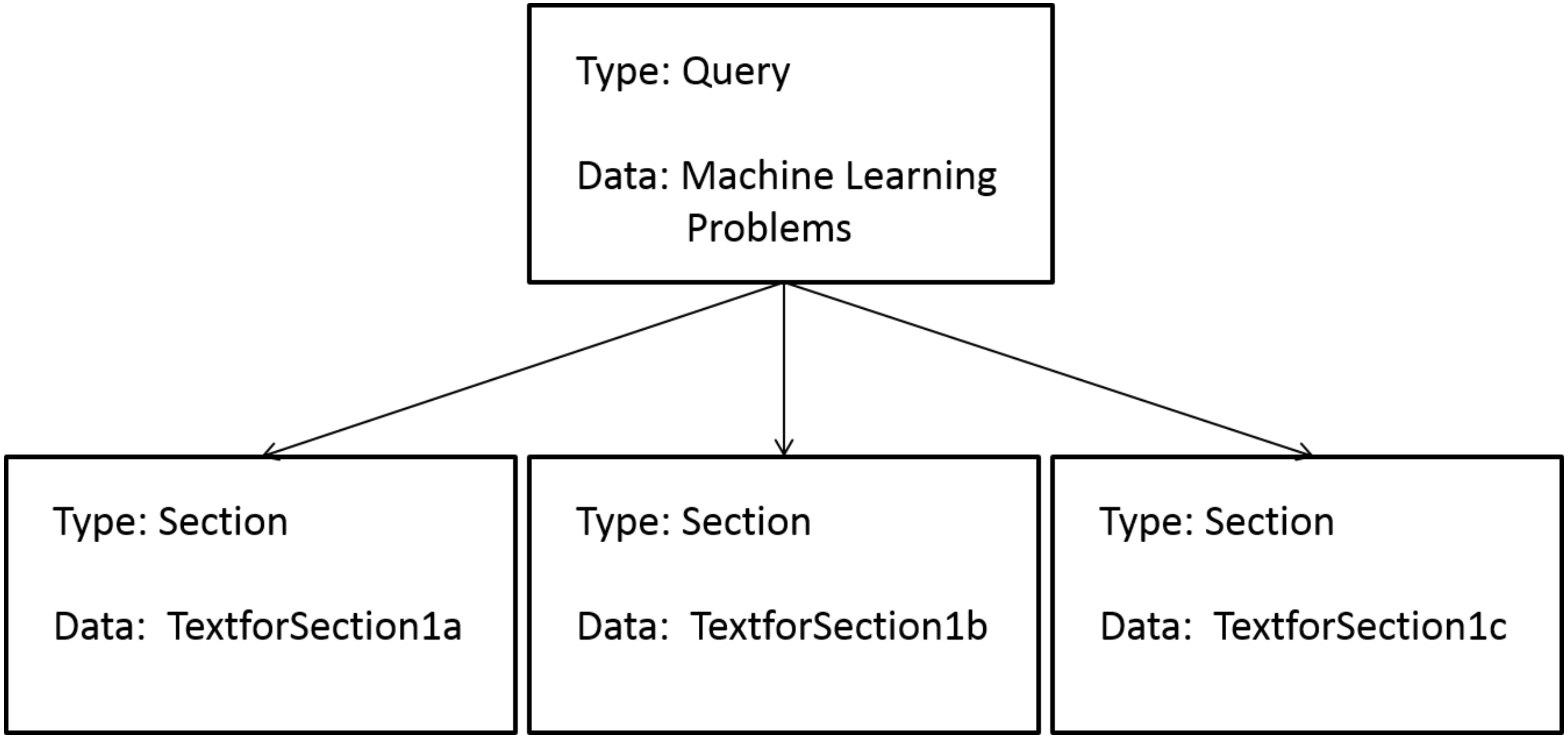
Data: ResultLink2

Type: Link

Data: ResultLink3

Section Generation

The Section Generator parses each of the links, removes unnecessary data and then divides each page into sections.



Type: Query

Data: Machine Learning Problems

Type: Section

Data: TextforSection1a

Type: Section

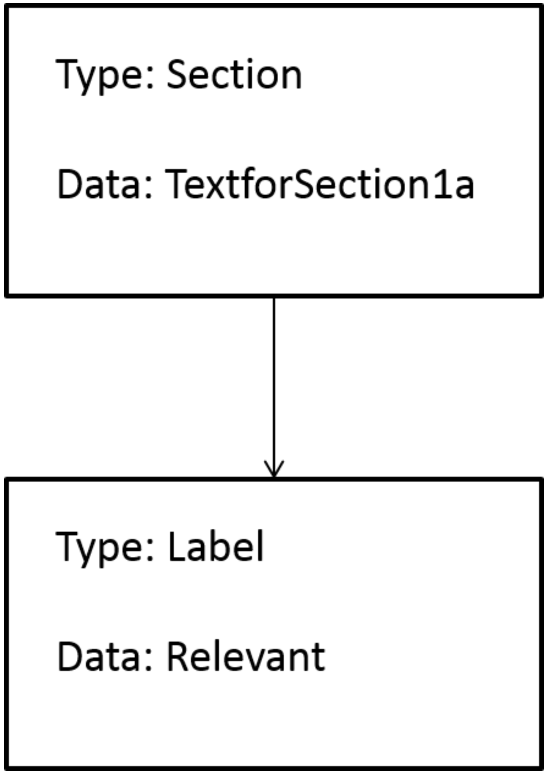
Data: TextforSection1b

Type: Section

Data: TextforSection1c

Labelling

Sections intended to be part of the training set will be labelled as relevant or not relevant.



Type: Section

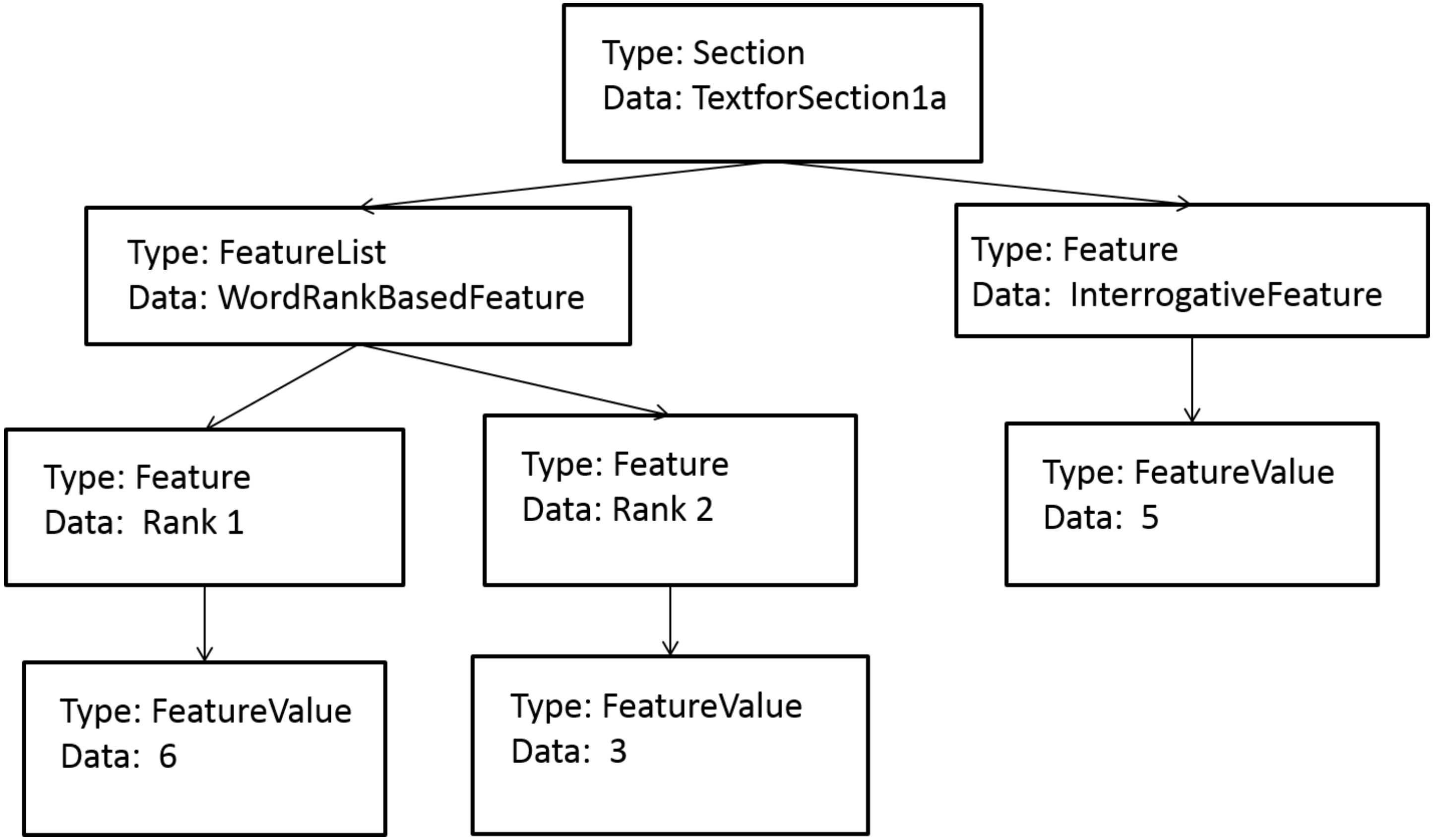
Data: TextforSection1a

Type: Label

Data: Relevant

Feature Generation

The features of each section are generated by the Feature Generator.



Type: Section
Data: TextforSection1a

Type: FeatureList
Data: WordRankBasedFeature

Type: Feature
Data: InterrogativeFeature

Type: Feature
Data: Rank 1

Type: Feature
Data: Rank 2

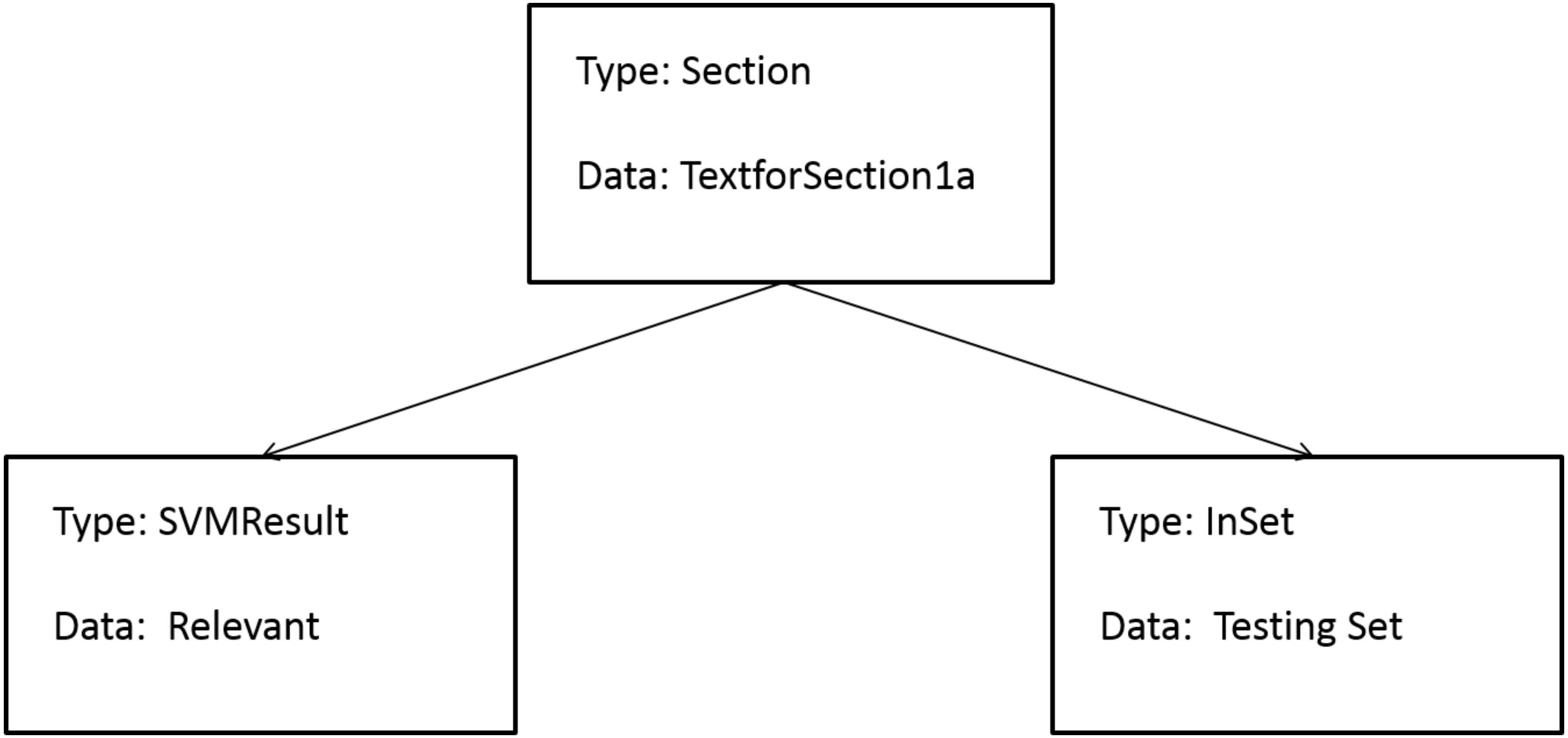
Type: FeatureValue
Data: 5

Type: FeatureValue
Data: 6

Type: FeatureValue
Data: 3

SVM Classification

The results of the Support Vector Machine classification and the set to which the result belongs are also represented as shown.



Type: Section

Data: TextforSection1a

Type: SVMResult

Data: Relevant

Type: InSet

Data: Testing Set

Data File Design

XML file format

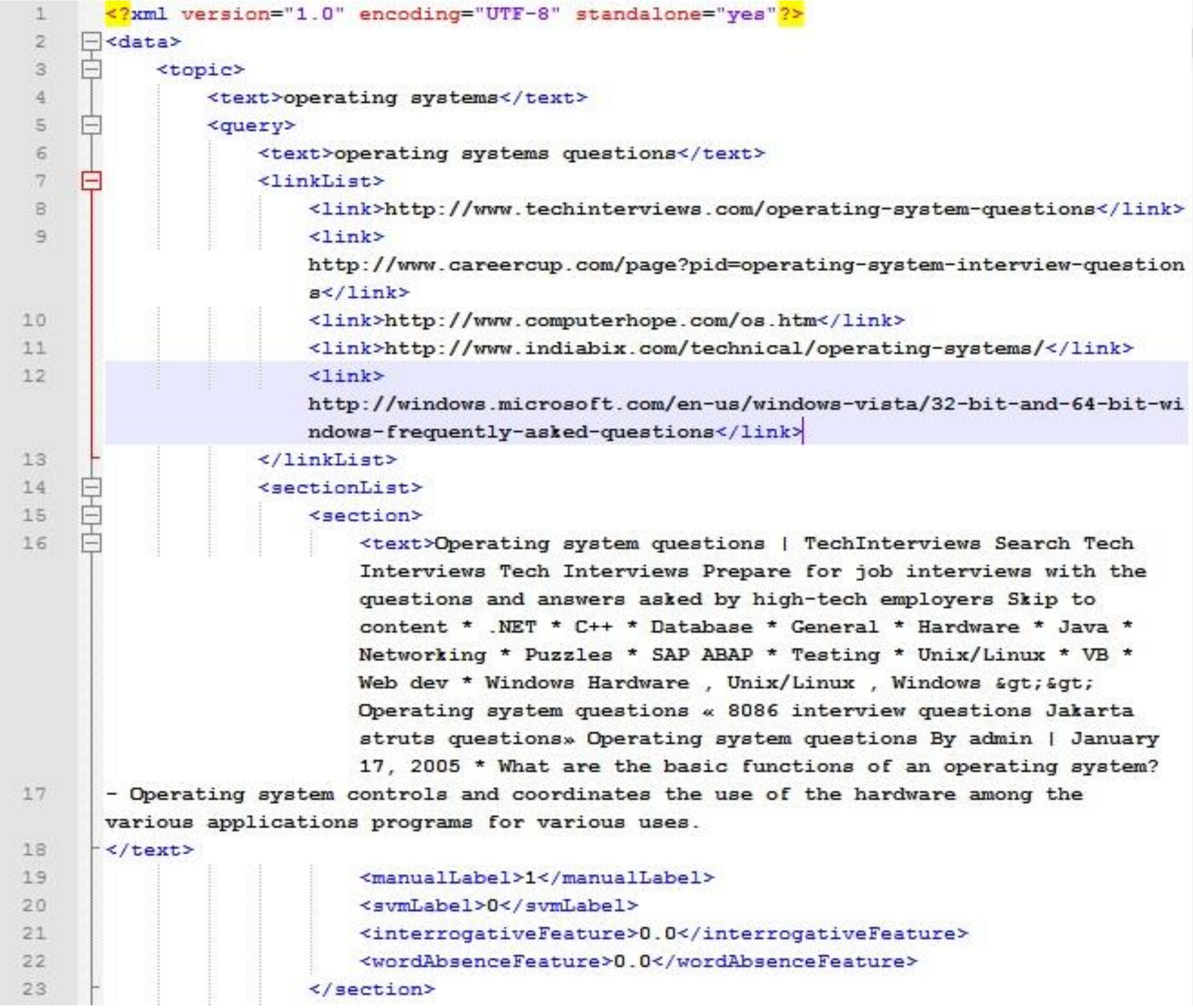
Suitable for a tree like data structure[2]

High Performance

Extensibility

Transparency - analyze or modify the file at any point

JAXB[5] (Java Architecture for XML Bindings)



```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<data>
    <topic>
        <text>operating systems</text>
        <query>
            <text>operating systems questions</text>
            <linkList>
                <link>http://www.techinterviews.com/operating-system-questions</link>
                <link>http://www.careercup.com/page?pid=operating-system-interview-questions</link>
                <link>http://www.computerhope.com/os.htm</link>
                <link>http://www.indiabix.com/technical/operating-systems/</link>
                <link>http://windows.microsoft.com/en-us/windows-vista/32-bit-and-64-bit-windows-frequently-asked-questions</link>
            </linkList>
            <sectionList>
                <section>
                    <text>Operating system questions | TechInterviews Search Tech Interviews Tech Interviews Prepare for job interviews with the questions and answers asked by high-tech employers Skip to content * .NET * C++ * Database * General * Hardware * Java * Networking * Puzzles * SAP ABAP * Testing * Unix/Linux * VB * Web dev * Windows Hardware , Unix/Linux , Windows &gt;&gt; Operating system questions « 8086 interview questions Jakarta struts questions» Operating system questions By admin | January 17, 2005 * What are the basic functions of an operating system?
- Operating system controls and coordinates the use of the hardware among the various applications programs for various uses.
</text>
                    <manualLabel>1</manualLabel>
                    <svmLabel>0</svmLabel>
                    <interrogativeFeature>0.0</interrogativeFeature>
                    <wordAbsenceFeature>0.0</wordAbsenceFeature>
                </section>
```

data.xml example

Work done in S8

- Class Design
- Data Design
  - Data Structure Design
  - Data File Design
- **Config Design**
- SVM Identification
- Implementation
- Analysis

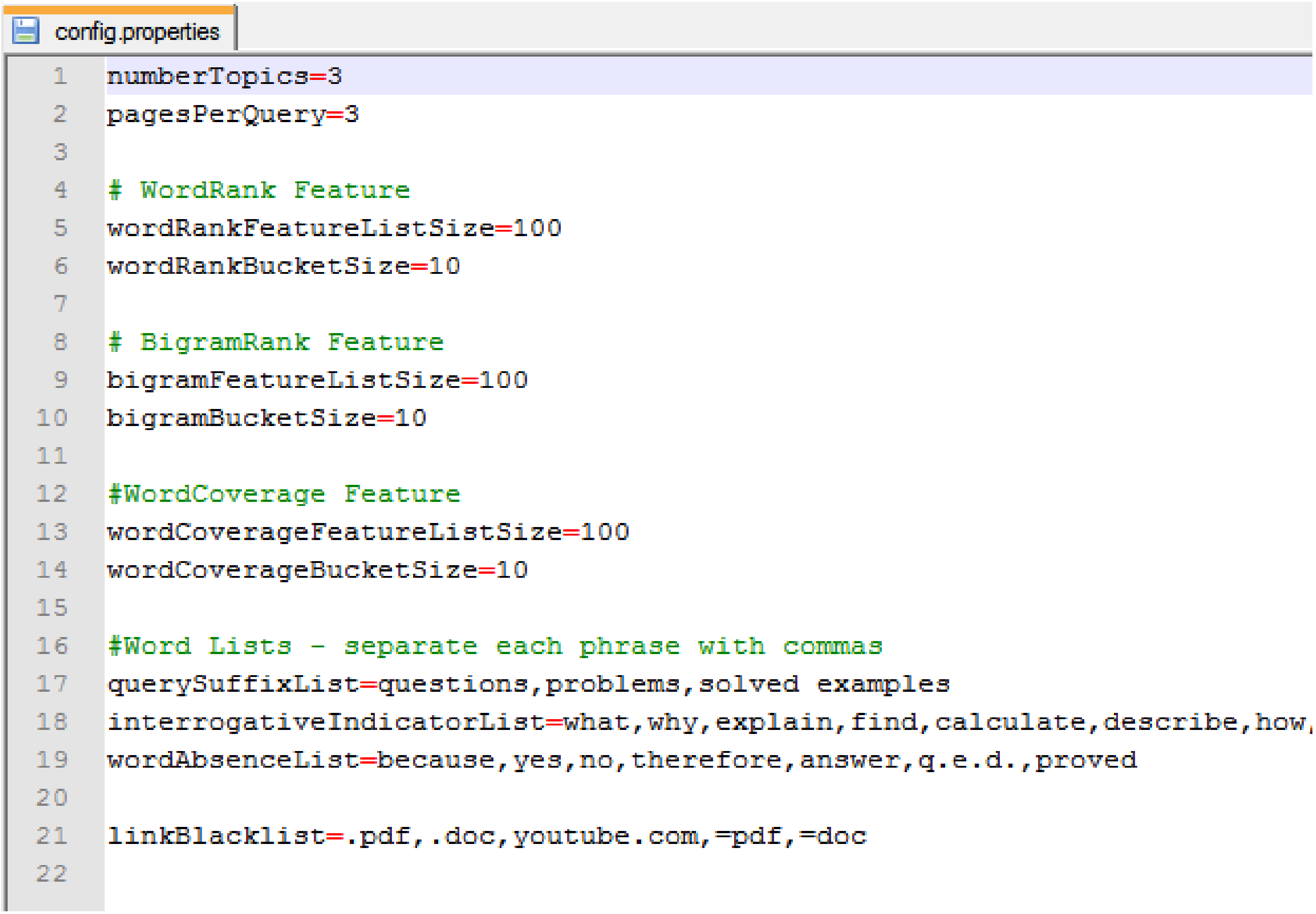
Config File Design

config.properties

Configurable parameters in the Java Properties format

The user is free to change parameters as per requirements

Reduce wastage of time during testing and analysis by eliminating needless recompilations



```
numberTopics=3
pagesPerQuery=3

# WordRank Feature
wordRankFeatureListSize=100
wordRankBucketSize=10

# BigramRank Feature
bigramFeatureListSize=100
bigramBucketSize=10

#WordCoverage Feature
wordCoverageFeatureListSize=100
wordCoverageBucketSize=10

#Word Lists - separate each phrase with commas
querySuffixList=questions,problems,solved examples
interrogativeIndicatorList=what,why,explain,find,calculate,describe,how,
wordAbsenceList=because,yes,no,therefore,answer,q.e.d.,proved

linkBlacklist=.pdf,.doc,youtube.com,=pdf,=doc

```

config.properties example

Work done in S8

Class Design

Data Design

- Data Structure Design
- Data File Design

Config Design

SVM Identification

Implementation

Analysis

SVM Package Identification

LIBSVM[3] library for the Support Vector Machine

Efficient implementation of Radial Basis Kernel[3]

Good performance[4]

High accuracy[4]

Work done in S8

- Class Design
- Data Design
  - Data Structure Design
  - Data File Design
- Config Design
- SVM Identification
- **Implementation**
- Analysis

Section Generation Heuristic

Paragraph and sentence boundary detection is done using the *SentParDetector* class from the *spiatools* library by Scott Piao[6]

Sentences with length > 400 characters further broken down

Every paragraph with sentences >= 6 split in half and the check is repeated

Frequency List Generation

Generated for Query objects and corresponding Section objects

Done for both unigrams and bigrams

Text content broken into tokens using the *StandardTokenizer* class from the Apache Lucene library[7]

The frequency of each token calculated and stored in a hashtable

- Key – token
- Value – frequency

The hashtable is sorted in order of decreasing frequency

Training Data Generation

The Training Data was generated for 3 topics –

- operating systems
- data structures and algorithms
- computer architecture

6467 sections

- 1243 relevant
- 5223 irrelevant

To avoid unbalanced data bias[8], SVM weightage –

- 4 for relevant sections
- 1 for irrelevant sections

Work done in S8

- Class Design
- Data Design
    - Data Structure Design
    - Data File Design
- Config Design
- SVM Identification
- Implementation
- **Analysis**

Performance Measure

$$t_i = \begin{cases} 1 & \text{if } f(x_i) = y_i \\ 0 & \text{if } f(x_i) \neq y_i \end{cases}$$

$$Accuracy = \frac{\sum t_i}{n} \times 100\%$$

$x_1, x_2, \ldots, x_n$ are the testing data

$f(x_1), f(x_2), \ldots, f(x_n)$ are the decision values by the SVM

$y_1, y_2, \ldots, y_n$ are the manual labels[4]

Performance Measure

5-fold Cross Validation Accuracy

In k-fold cross-validation, the sample is partitioned into k equal size subsamples and a single subsample is retained as the validation data, and the remaining k-1 subsamples are used as training data.

This is repeated k times and the results averaged to produce a single estimation.

Feature Analysis

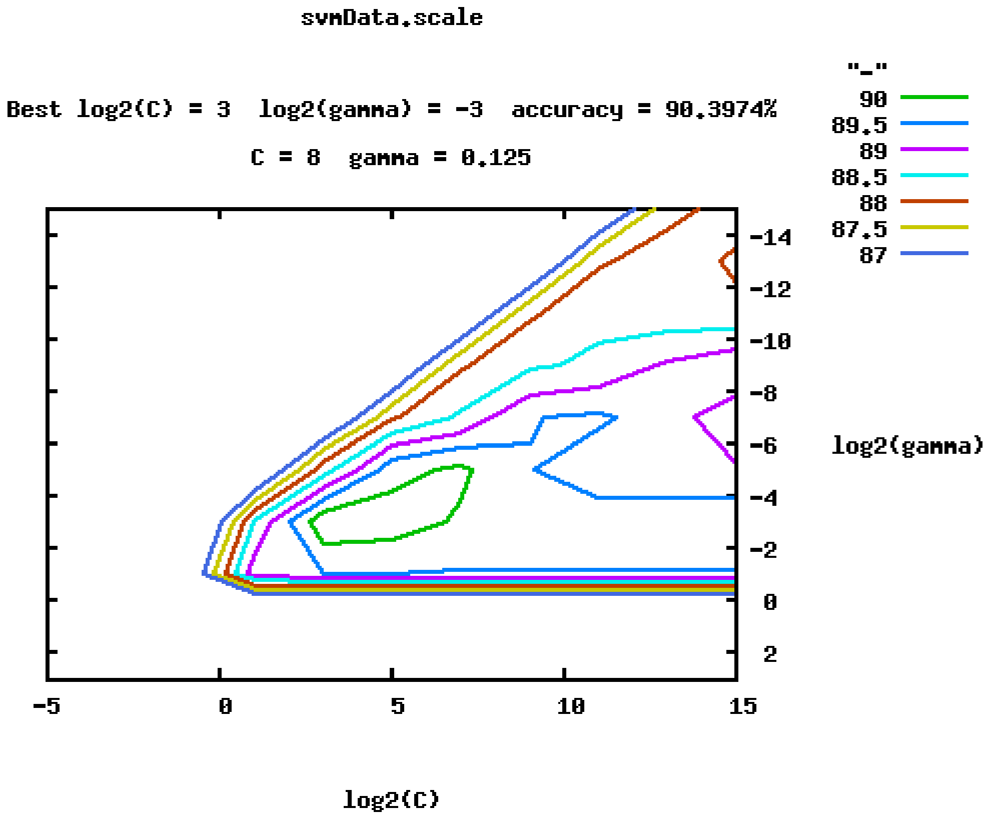
For each combination of features –

Features scaled[9] to [0,1] using *svm_scale*

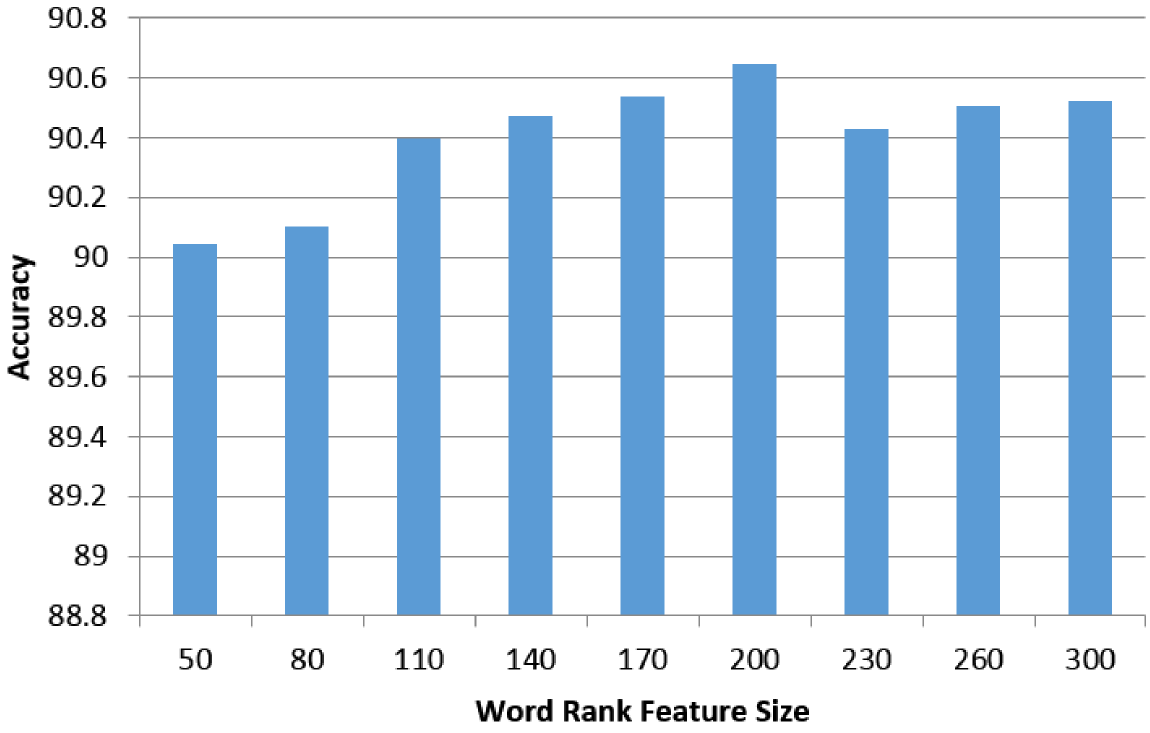
Optimal C and γ SVM parameters determined using *grid.py*

5-fold Cross Validation Accuracy determined

Feature Analysis



Word Rank Based Features



Word Rank Bucket Size = 1

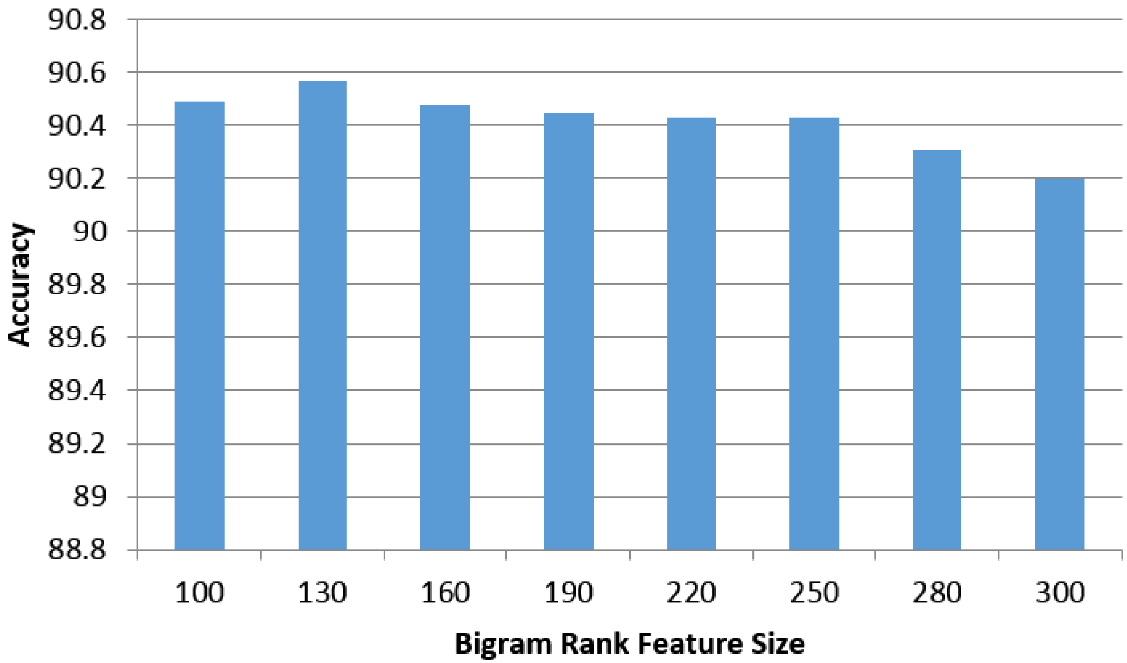
Bigram Rank Bucket Size = 1

Word Coverage Bucket Size = 5

Bigram Rank Features = 150

Word Coverage Based Features = 150

Bigram Rank Based Features



Word Rank Features = 150

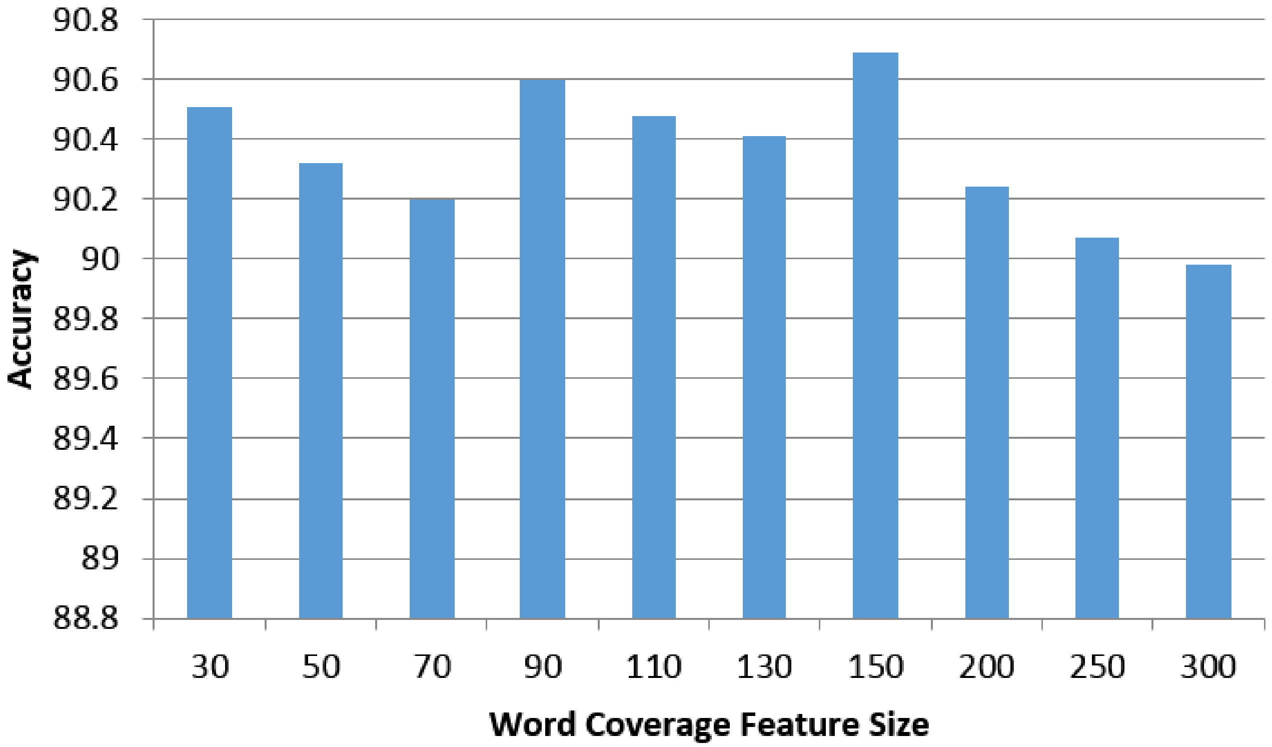
Bigram Rank Bucket Size = 1

Word Coverage Bucket Size = 5

Word Rank Bucket Size = 1

Word Coverage Based Features = 150

Word Coverage Based Features



Word Rank Features = 150

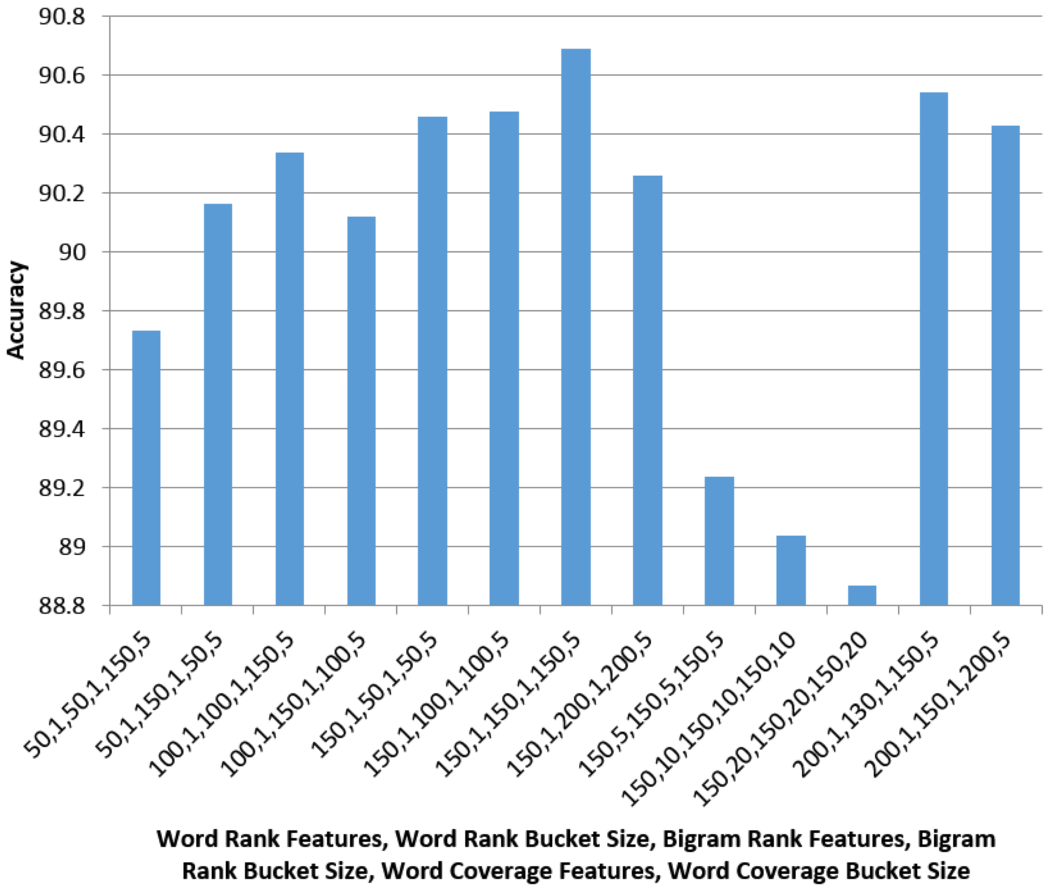
Bigram Rank Features = 150

Word Coverage Bucket Size = 5

Word Rank Bucket Size = 1

Bigram Rank Bucket Size = 1

Optimal Feature Configuration



Optimal Feature Configuration

Accuracy **90.69%**

Word Rank Features	= 150
Word Rank Bucket Size	= 1
Bigram Rank Features	= 150
Bigram Rank Bucket Size	= 1
Word Coverage Features	= 150
Word Coverage Bucket Size	= 5

Optimal SVM parameters

$C = 8.0$

$\gamma = 0.125$

Training Set Size Analysis

Procedure –

Training Set generated using stratified subsampling of Data Set

The features were scaled[9] to [0, 1] using *svm_scale*

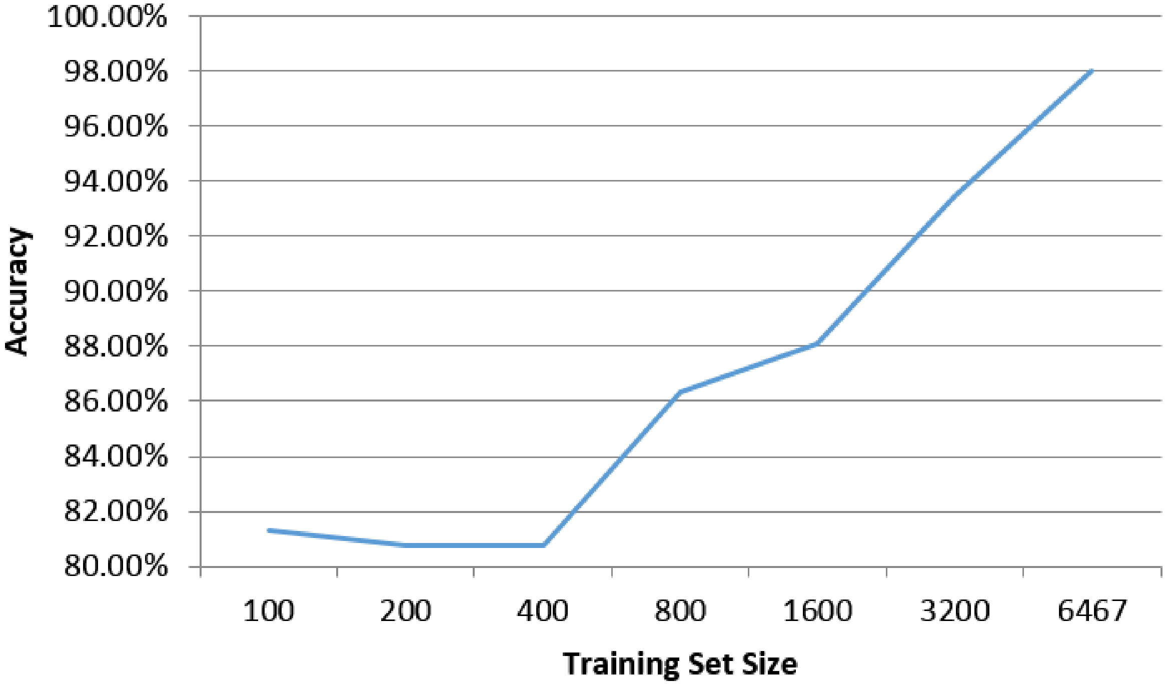
Optimal *C* and *γ* SVM parameters determined using *grid.py*

SVM Model generated using the Training Set

Accuracy calculated by predicting the Data Set

Observation

Accuracy increases exponentially with training set size.



Proposed Feature Analysis

The effect of our proposed features was studied by

Configurations of the Training File generated by selecting combinations of the 3 classes of features

Features scaled[9] to [0,1] using *svm_scale*

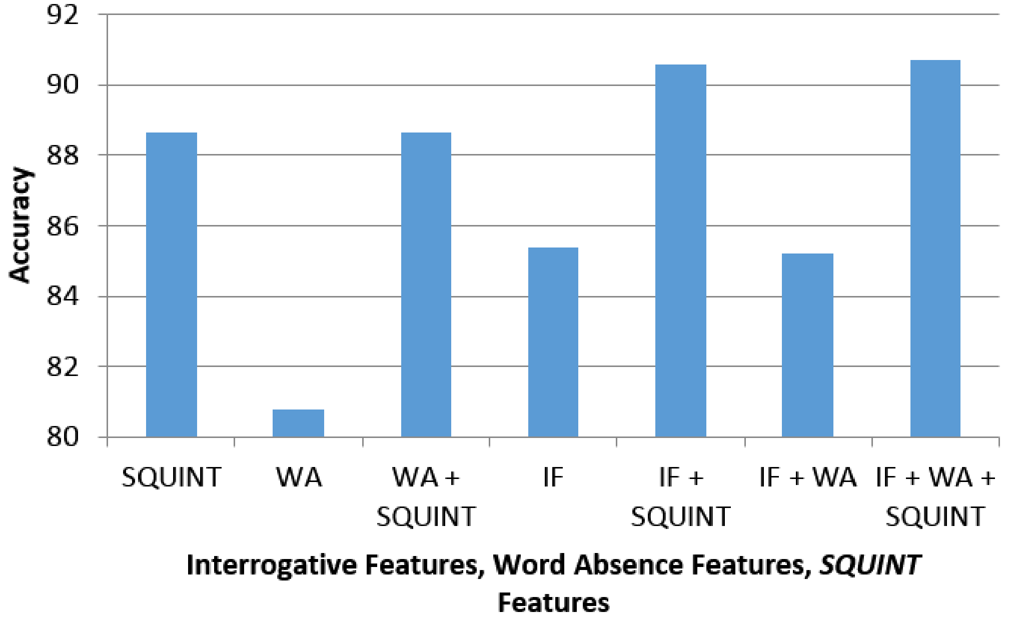
Optimal *C* and *γ* SVM parameters determined using *grid.py*

5-fold Cross Validation Accuracy determined

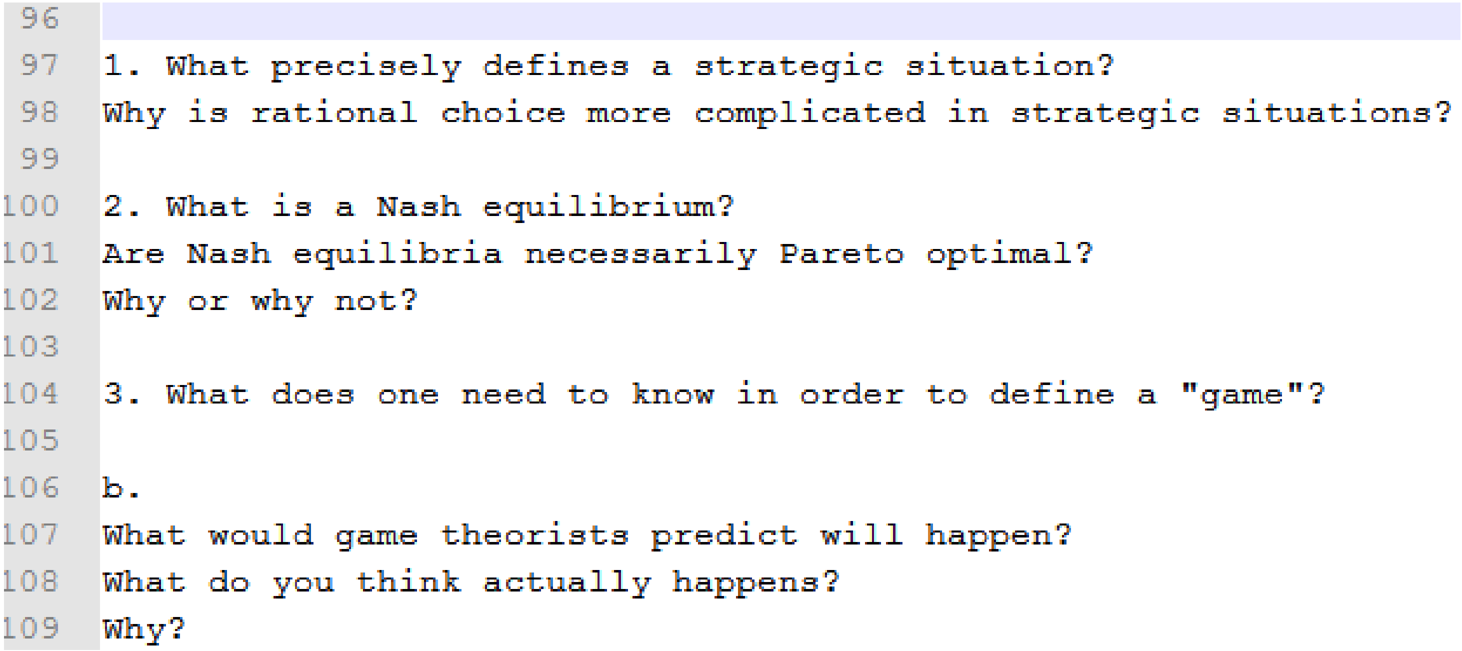
Observation

Interrogative feature + SQUINT features – 90.55% accuracy

All features – 90.69% accuracy



Real Run



```

1. What precisely defines a strategic situation?
Why is rational choice more complicated in strategic situations?

2. What is a Nash equilibrium?
Are Nash equilibria necessarily Pareto optimal?
Why or why not?

3. What does one need to know in order to define a "game"?

b.
What would game theorists predict will happen?
What do you think actually happens?
Why?
```

Sample topic – ‘Game Theory’

Contents

1. Introduction
2. Work Done in S7
3. Work Done in S8
4. **Conclusion and Future Work**

Conclusion and Future Work

Proposed method of using a SVM to classify relevant questions from the internet – successful

Future work –

Multi-class SVM classification

Other features

Using different types of classifiers eg. Neural Networks

References

1. SQUINT - SVM for Identification of Relevant Sections in Web Pages for Web Search, Riku Inoue, Siddharth Jonathan J.B., Jyotika Prasad, Department of Computer Science, Stanford University

2. XML, http://en.wikipedia.org/wiki/XML

3. LIBSVM, http://www.csie.ntu.edu.tw/~cjlin/libsvm/

4. LIBSVM, Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, April 15, 2010.

5. JAXB, Ort, Ed, and Bhakti Mehta. "Java architecture for xml binding (jaxb)." *Sun Developer Network* (2003).

6. A Highly Accurate Sentence and Paragraph Breaker, Scott Piao, 2008, http://text0.mib.man.ac.uk:8080/scottpiao/sent_detector

7. Apache Lucene - a high-performance, full-featured text search engine library, Jakarta, Apache, 2004.

8. An approach for classification of highly imbalanced data using weighting and undersampling, Ashish Anand, Ganesan Pugalenthi, Gary B.Fogel, P. N. Suganthan, *Springer Journal*.

9. A Practical Guide to Support Vector Classication, Chih-Jen Lin, Department of Computer Science, National Taiwan University, Talk at University of Freiburg, July 15, 2003.