

# Mini Project

## Topics In Operating Systems

Title: External Information Snippets Provider

Asi Lichtenstein 060963428

Benny Hansav 039157698

# Introduction

Saya is the new departmental robot receptionist, developed in order to simulate human like communication.

She achieves that by using Microsoft Speech SDK in order to synthesize speech and produce facial expressions in response to a precompiled dictionary.

The main goal of our project is: to provide the ability to responses to specifics user queries on variety subjects, such as: Sports , Weather , politics and etc , using data mining from the web , that is, retrieve relevant and updated information regarding a specific subject from the web.

# Motivation

Currently Saya's conversation abilities are poor.

Our main goal is to extend those abilities in order to Saya be able to conduct a more human like Conversations, that is, the content of Saya's answer will be logically related to what she has been asked for.

For example: if Saya has been asked question about Sports, her answer will be related to Sports.

We tend to cover more popular topics, such as: Sports, Politics, Weather and so on.

Therefore we managed to extend Saya's dictionary for each topic.

We also want that Saya's retrieved information to be up to date.

# Architectural description of the solution

In order to retrieve relevant and up to date information we used RSS reader.

Our RSS reader is based on an open source package which called ROME.

It is responsible of parsing, generating and publishing RSS and Atom feeds.

For each topic that we support (e.g. Sports, Politics and etc) we have a key words dictionary and the urls stack.

We support configuration file which defines for each topic the urls from which we are parsing the RSS feeds and the key words.

We have a class (SAYA\_RSS\_READER) which populates the data structures we use which are: for each topic we have the urls stack (from which we are reading and parsing the RSS feeds)

And the key words dictionary (the key words used to match the input string against the relevant topic).

In addition we have another class (RSS) which reads the relevant RSS feeds and returns the feeds as string.

Now we will describe the RSS reader algorithm:

We have a pool of RSS records from which we are returning records to the user.

Each request decreases 1 record from the record pool which is a vector of RSS records, when this pool reaches 0 we will populate our pool with new records from a new url which we pop from the urls stack.

From our experiments we have learned that when the size of the urls stack is  $\geq 5$

There are no duplicate records returned within a single conversation.

# User Guide

The first item you should provide is the configuration file which defines the data base.

It contains right now only 3 topics: Sports , Politics and Financial.

The user can add new urls and new key words to each topic which extend the Topic dictionary and the urls stack.

Please read the configuration file sample in the next page.

Here is a code example for using our program:

```
SAYA_RSS_READER saya = new SAYA_RSS_READER("conf.txt");
```

```
String ans;
```

```
ans = saya.HandleInputString("sport");
```

We create a new SAYA\_RSS\_READER object , which the argument for the constructor is the configuration file name , and by executing the method HandleInputString we get the relevant answer (if there is one) as a string. In the above example the input string is “sport”.

\*for running the program you should include the 2 jar files: jdom and rome-1.0RC1

# Sample Conf File

sportsNumOfUrls = 1

sportsNumOfWords = 5

sportsUrl0 = <http://msn.foxsports.com/feedout/syndicatedContent?categoryId=235>

sportsWord0 = sport

sportsWord1 = sports

sportsWord2 = soccer

sportsWord3 = basketball

sportsWord4 = football

newsNumOfUrls = 1

newsNumOfWords = 1

newsUrl0 = <http://msn.foxsports.com/feedout/syndicatedContent?categoryId=235>

newsWord0 = news

financialNumOfUrls = 1

financialNumOfWords = 1

financialUrl0 = <http://msn.foxsports.com/feedout/syndicatedContent?categoryId=235>

financialWord0 = financial

# Future work

The fact that we are reading our data from a file gives us the ability to extend:

The number of urls and the number key words very easily without changing the source code at all.

In addition adding more topics can be achieved only by adding new attributes (*\_topicDictionary* , *\_topicurls* , *\_topicObj*) to *SAYA\_RSS\_READER* class.

*Those properties make this program flexible* and generic, so future changes can be done straight forward without be familiar with the source code.

Another improvement is to add a time mechanism for updating the RSS records.

A problematic situation can occur when no one have talked to Saya for a while which leads to a situation that the records in our data structures might be out of date.

The solution to this situation can be: adding a time element for keeping the records up to date.

This can be done by checking if the delta between the current time and the last time Saya

Was talked to is bigger than t – read new records from the next url.

# Bibliography

- <https://www.rome.dev.java.net>
- <http://www.cs.bgu.ac.il/~orlovm/teaching/saya>
- <http://www.jdom.org>
- [http://en.wikipedia.org/wiki/RSS\\_\(file\\_format\)](http://en.wikipedia.org/wiki/RSS_(file_format))
- <http://www.rssreader.com/>