Personalized RSS Aggregator using an Artificial Immune System

Adam Burkepile
Western Washington University
adam.burkepile@gmail.com
http://www.adamburkepile.com/

Abstract

In the universe of the web, information overload can be a very real problem for some people. The number of stories a person can read from a set group of site in a day can be vast and ever increasing as time goes by. RSS is a method that can help standardize and condense information, but that doesn't help when the number of stories is overwhelming. We present a method of filtering through a large number of RSS stories in the hopes of condensing a large set of stories of from sites the user frequents into a a smaller subset that includes only the stories the user would find most interesting. We do this by creating a group of cells, which are a group of significant words from that story, to try to identify stories that the user would find interesting based upon the past selection of the user. These "cells" are analogous to antibodies in the human immune system, except instead of destroying things they match, the matches identify the stories the user would be interested in.

Categories and Subject Descriptors I.5.4 [*Applications*]: Text processing

General Terms RSS, RSS Feed Aggregator, AIS, Artificial Immune System

Keywords RSS, Artificial Immune System

WWU '08 12/12/2008, Bellingham.
[Adam Burkepile].

1. Introduction

The intent of this research is to examine the viability of using an artificial immune system as a selection algorithm for a RSS news feed aggregator in order to personalize a set of news feeds into a single feed that would be customized to show only stories the system thinks the user would be interested in. The problem with any news feed is that the topics it can cover can be too broad. The personalized feed in our system would be based on past selections of the user.

1.1 RSS

RSS (1) has a few different meanings; Rich Site Summary (RSS 0.91), RDF Site Summary (RSS 0.9 and 1.0), or Really Simple Syndication (RSS 2.0). What they are, in their most basic form, is a an XML file that contains a title and/or a description for an item; In our case the item is the story with a title and body. For downloading and processing of the RSS feeds themselves, we used a freely available RSS feed aggregator from IBM. It automatically downloads the RSS feed and parses the XML and stores the information. The aggregator is written in PHP and allows for flexible and easy processing of the information.

1.2 Artificial Immune System (AIS)

Artificial Immune Systems are "computational systems inspired by the principles and processes of the vertebrate immune system." (2). Simply put, they are systems inspired by the vertebrate immune system, the way it adapts and learns new things it has to guard against and how to adapt this into a machine learning algorithm that can be used to solve certain computational problems.

2. Generation of Cells

In order to examine and compare the stories, we generate cells, which are groups of words from that story. When creating these cells, we want to attempt to select only the most meaningful words that best represent story. We use stemming and stop word elimination to filter out words that don't contribute much to the meaning of the story.

We generate multiple cells for each story for a number of reasons. Firstly, generating multiple cells increases the changes that we will generate a cell or multiple cells that best describes the story. Secondly, we can use the fact that we have multiple cells for each story in our comparison algorithm, which more detail will be given in the Comparison Algorithm section.

2.1 Stemming

In order to collect accurate and meaningful statistics when doing term analysis, we must use some sort of stemming algorithm to normalize the words. We chose to use the Porter Stemmer (4) for a few reasons. Although it is inferior to some other stemming algorithms, the Porter Stemmer is regarded as a frozen algorithm, meaning that there are no new changes coming to the algorithm. This makes the algorithm a good choice when doing research work where results need to be repeatable. Also, even though the Porter stemmer might stem a word in a strange way (days becomes dai), this does not interfere with the term analysis and is unseen by the user so is not considered a problem.

2.2 Stop Words

Words that appear in the majority of the stories can not contribute that much to the meaning of the story and do not do anything to describe the uniqueness of the story. Because of this, we want to identify these "common" words and add them to a stop list, which is a list of words that we will filter out and are not going to consider part of a story.

2.3 Document Frequency

Apart from using predefined stop list from our book in Information Retrieval (3), we discovered there are many other words that should be removed or not selected from the stories. We used the Inverse Document Frequency measurement to calculate how common a word was.

Inverse Document Frequency

$$idf_i = \log \frac{|D|}{|\{d_j : t_i \in d_j\}|}$$

D = Total number of documents

$d_j : t_i \in d_j$ = number of documents term t_j appears

2.4 Custom Stop List

We used the static stop list provided from the Information Retrieval book (3) as the basis for our stop list but we needed to add to that list to remove more words that were not very descriptive. The method we decided to use to generate a custom stop list was a mix of automatic generation and manual removal. First, we calculate the inverse document frequency for each term in the corpus. Once we have the values for each term, we group all the terms together with other terms that had the same idf. We then identify the groups that have the predefined stop list terms in them. The group the has the last predefined stop word is then identified and any group with a idf less than that group is added to the stop list. This is the automatic part of the list generation. Once we have this list, we found that words such as "iphone" or "apple" could make it into the stop list when they shouldn't. This is where we manually go in an remove words that we consider meaningful to the story. This leaves us with our custom static stop list that we use to remove words that contribute little to the meaning of an story.

3. Comparison Algorithm

Once we have calculated the Inverse Document Frequency, eliminated the stop words, and created the cells for the stories; we can finally compare the stories to each other. The way we are currently doing this depends on two factors, the percentage each cell matches and the number of cells that match.

3.1 Cell Matches

Since each story has N number of cells, when we compare one story to another, we compare every cell to every cell in the other story leading to potentially N^2 cell matches. We are typically setting N at around $5 - 100$ right now, although we are still experimenting with the right settings.

3.2 Cell Similarity

The other half of the measure involves comparing one cell to another. This involves some sort of similarity

measurement and finding a minimum similarity that represents a match. In order to measure the similarity between two cells we use the Jaccard similarity coefficient.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

This measurement shows similarity between two sets. The similarity in this measurement is the size of the intersection divided by the size of the union of the two sets. We felt this was a fair measurement to use since it captures the intersection of the two sets as well as the uniqueness.

4. Conclusion

Although we are somewhat early to midway through this project, we are already seeing some promising results when simply doing a many-to-many comparison with the corpus of stories. We believe that as we incorporate more and more of the features that was intend to, the results will only get better.

Acknowledgments

The author would like to thank his supervisor, Perry Fizzano, for all his help and support with this project.

References

[1] RSS
http://en.wikipedia.org/wiki/Rss

[2] Artificial Immune System
http://en.wikipedia.org/wiki/Artificial_Immune_System

[3] Introduction to Information Retrieval
http://www-csli.stanford.edu/∼hinrich/information-retrieval-book.html

[4] The Porter Stemming Algorithm
http://tartarus.org/∼martin/PorterStemmer/