News Summarization:
Building a Fusion (a Solr based system) special collection:
News articles template summarization and categorization

Souleiman Ayoub, Julia Freeman, Tarek Kanan, Edward Fox

Computer Science 4624, Spring 2015, Virginia Tech, Blacksburg, VA 24061
Email: {siayoub, juliaf, tarekk, fox}@vt.edu

March 15, 2015

Table of Contents

Cover Page 1
Table of Contents 2
Table of Tables 3
Table of Figures 3
1. Requirements 4
5.1. Abstract 4
5.2. Objective 4
5.3. User Roles 4
5.4. Intent 4
5.5. Approach 5
5.6. Milestones 5
2. User Manual 5
3. Developer's Manual 5
3.1. Prerequisite Knowledge 5
3.2. Collection 6
3.3. NER 6
3.4. Current Progress 6
3.5. Sketch of Application Process 7
3.6. Possibilities for Future Program Use 8
4. Design 8
4.1. Implementation 12
4..1. Programming Languages 12
4..2. Tools and Libraries Employed 13
5.1. Code Repository Plans 13
5.1. Phases 14
4..1. Text and Attribute Extraction 14
4..2. Summarization 14
4..3. Indexing Documents 14
4..4. Testing 14
5. Prototyping 15
5.1. Classification with Weka 15
5.2. Bringing it together 16

5.3. Fusion 17
6. Testing 18
7. Timeline 18
8. Lessons Learned 19
9. Conclusion 19
10. Acknowledgments 19
11. References 20

Table of Tables

Table 1 - Categories 15
Table 2 - Sample Header of Feature Set 15
Table 3 - Sample overview of features for Apple Review 15
Table 4 - Average F1 Measure per Model 15

Table of Figures

Figure 1 - Processing of the PDF news article through the application 7
Figure 2 - Developer's Data Flow 7
Figure 3 - Lucidworks Fusion capabilities and relations[1] 8
Figure 4 - Solr interface used for querying. 9
Figure 5 - Weka interface used for data mining. 10
Figure 6 - Article view with "invisible" backend tags 11
Figure 7 - From left to right this is the typical best run time speed of C#, Java, and Python 12
Figure 8 - Security is a major issue for any project 14
Figure 9 - Sample result of Summarized Article 16
Figure 10 - Sample result of result in Fusion 17
Figure 11 - Timeline of the implementation of the project 18

1. Requirements

5.1. Abstract

This project will attempt to take Arabic PDF news articles and end with results from our new program that index, categorize, and summarize them. We will fill out a template to summarize news articles with predetermined attributes. These values will be extracted using named entities recognizer (NER) which will recognize organizations and people, topic generation using an LDA [1] algorithm, and direct information extraction from news articles' authors and dates. We will use Fusion LucidWorks[4] (a Solr[5] based system) to help with the indexing of our data set and provide an interface for the user to search and browse the articles with their summaries. Solr[5] will be used for information retrieval. We hope to end with a program that enables end users to sift through news articles quickly.

5.2. Objective

The summarized articles need to be archived in such a way that it can be retrieved to allow us (and possibly future users) to use. With the use of Fusion, we can archive these information to allow us to search and view the summarized articles. However, in order to achieve this, we'll need to collect the information that exists in the article via tools such as an NER, LDA and a form of classification to determine subject (i.e. sport, politics, etc.) With these information, we can use a template to help us summarize each articles.

5.3. User Roles

Each individual has a different role on the team. The two students currently taking the Hypertext and Multimedia Capstone are Julia Freeman and Souleiman Ayoub. Julia Freeman will be a developer as well as a peer evaluator. Souleiman Ayoub will also be a developer. Tarek Kan'an will be a mentor and team leader.

5.4. Intent

By May 8, 2015 we hope to have an application that can:

1. Parse Arabic PDF[5] news sources and extract articles.
2. Obtain useful information from the parsed articles.
3. Use the extracted information to fill in empty templates, generating Arabic news article summaries[7].
4. Enable the user to browse articles along with their summaries.

We will be using Weka[3] machine learning, Solr[5] retrieval system, Fusion[4], LDA's [1], NER's [2], and Java to create, extract, and generate the final summaries and to provide the user the ability to see the the articles and the summaries all in one place.

5.5. Approach

5.6. Milestones

- By February we will work on extracting the articles' main attributes like categories, Named Entities, and Topics; using machine learning tools, NERs[2], etc.
- By March we will learn Solr[5] and Fusion[4] and implement and modify Fusion schemas to include extra fields.
- By April we will connect summarization results with Fusion[4] to enable automation. Then we will validate the results of the programs and prepare a final report of what we did, what we were successful with, and what we might not be able to complete.
- We reserve the entirety of May for final touchups, debugging, and user testing.

2. User Manual

There is currently no existing system for what we are attempting to do. We are piecing together a few existing algorithms and methods for topic generating like LDA [1] (Latent Dirichlet Allocation) and for named entity extraction like NER[2] (Named Entity Recognizer) but we have to alter them to fit our project needs like handling the Arabic language which can be very challenging. Hopefully in the future users will be able to see trends in news data which can help with security or data mining.

3. Developer's Manual

3.1. Prerequisite Knowledge

In order to use the software and modification needed to be made there are some prior knowledge that is required in order to understand the scope of the application. Developer should be well-versed in a programming language (preferably Java and Python) and have at least a basic understanding of natural language processing and machine learning in order to understand the underlying concept used by the tools to help us achieve a solution. The following tools we have used:

- Java (1.8 or greater) SDK from Oracle
- Python (3.x or greater) SDK from Python
- Weka (3.6.12 or greater) from University of Waikato
- RenA - Arabic NER (provided) from Souleiman Ayoub and Tarek Kanan
- ALDA - Arabic Latent Dirichlet Allocation (provided) from Souleiman Ayoub and Tarek Kanan
- Fusion from LucidWorks

3.2. Collection

We will also be providing the collection of roughly 120,000 articles which will can be used to alter, modify or append to if necessary depending on end-goal. These articles are encoded in UTF-8 and should be processed using UTF-8 Encoding/Decoding, most languages such as Java and Python provides support (BufferedReader[8] Java API and codecs[9] Python API for more info). The use of the following tools NER, LDA and classification will be used to help us generate a summary to provided in the fusion schema along with the article. Each of these articles will be classified using Weka, more explanation will be provided below.

3.3. NER

There are two ways to use the NER, we have provided a python script that will quickly generate named entity extraction if needed for testing purpose, the python script is called *ner.py* and can be used as follows:

> ner.py -F <text_file> -t=<[PERS,LOC,ORG]> > <output>

The command above will generate a text file which consists of named entities based on the given file. -t are the named entity given to extract.

However, for a more advanced extraction, such as n-gram solution and advanced structures. Please refer to the class ***arabic.ner.RenA*** which consists of the option to request more features.

3.4. Current Progress

We are currently trying to perfect a way to parse text documents into ARFFs (Attribute-Relation Files) that will be used as input to the machine learning program. This type of document (ARFF) is ideal for the project because we can more easily scan the document, categorize, and summarize it as opposed to creating a whole other program to parse text documents. The conversion is not perfected yet because some articles might only contain pictures which are of no use to the program. We also remove any stop words which are words which are placeholders like “the” or “a” but in Arabic. This means we will have to go through and remove any empty files after they have been converted. To ensure that we are creating ARFFs properly and they are categorized properly (ex., a soccer article is not put into the Art category) we will have to manually test a sample of the data to make sure that it works for the entirety of the data set.

3.5. Sketch of Application Process

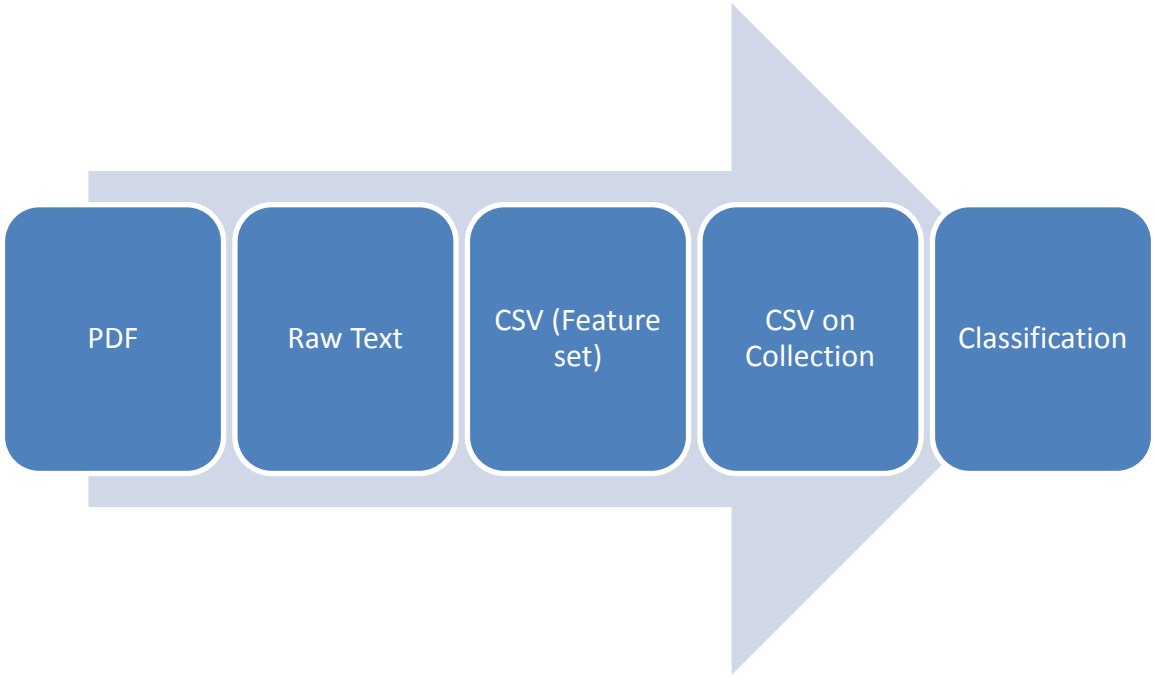

Figure 1 - Processing of the PDF news article through the application

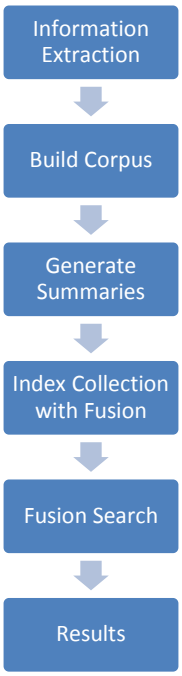

Figure 2 - Developer's Data Flow

3.6. Possibilities for Future Program Use

We are only planning to implement this program for the Arabic language. We hope that the work can be extended for use with more languages in the future. We are working with Arabic which means the code we write has to be language independent because some of the programmers do not speak or read Arabic. Optimistically countries like France or Australia that are trying to analyze news related issues could sort through news articles under a certain category and then use the information as meta data. It also helps that this is a scholastically created project so there are no monetary sponsors that could influence the creation of the project. News in America is notoriously bi-partisan and hopefully this will be a way to view news trends without trying to sway the end user to a particular viewpoint (more specifically the viewpoint of the sponsor). It is also beneficial that we are using Java as a language to create the program because it is one of the most widely used programming languages in the technical community.

4. Design

We are using Lucidworks Fusion for this program. It has a lot of capabilities that we are using, mainly for indexing.

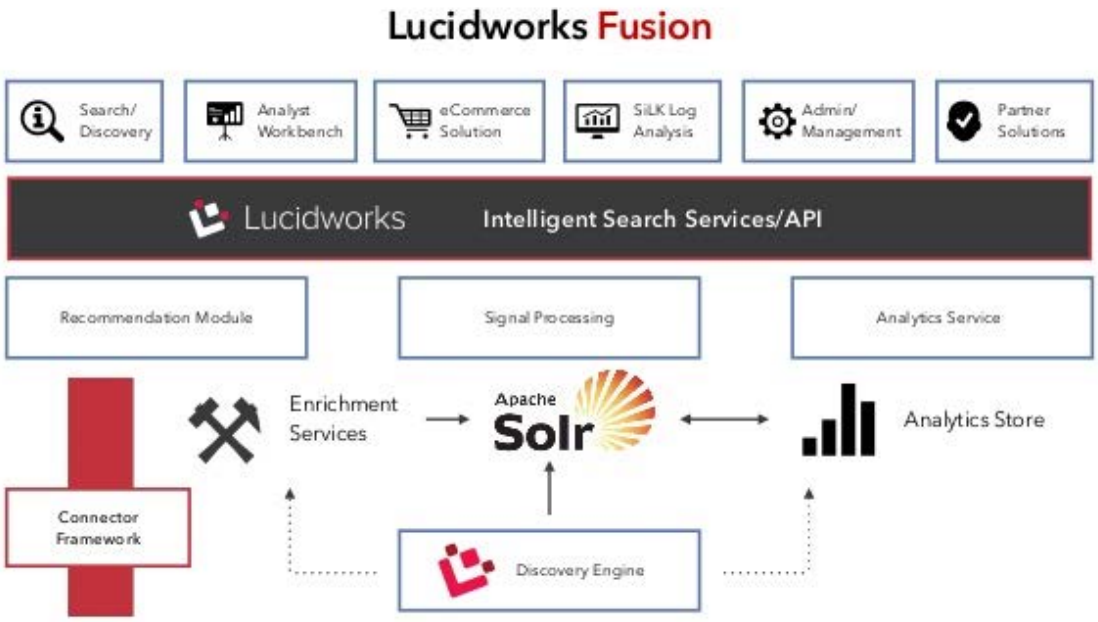

Figure 3 - Lucidworks Fusion capabilities and relations[1]

Fusion is built off of the Solr Apache system. We use Solr for querying after we have indexed the news items.

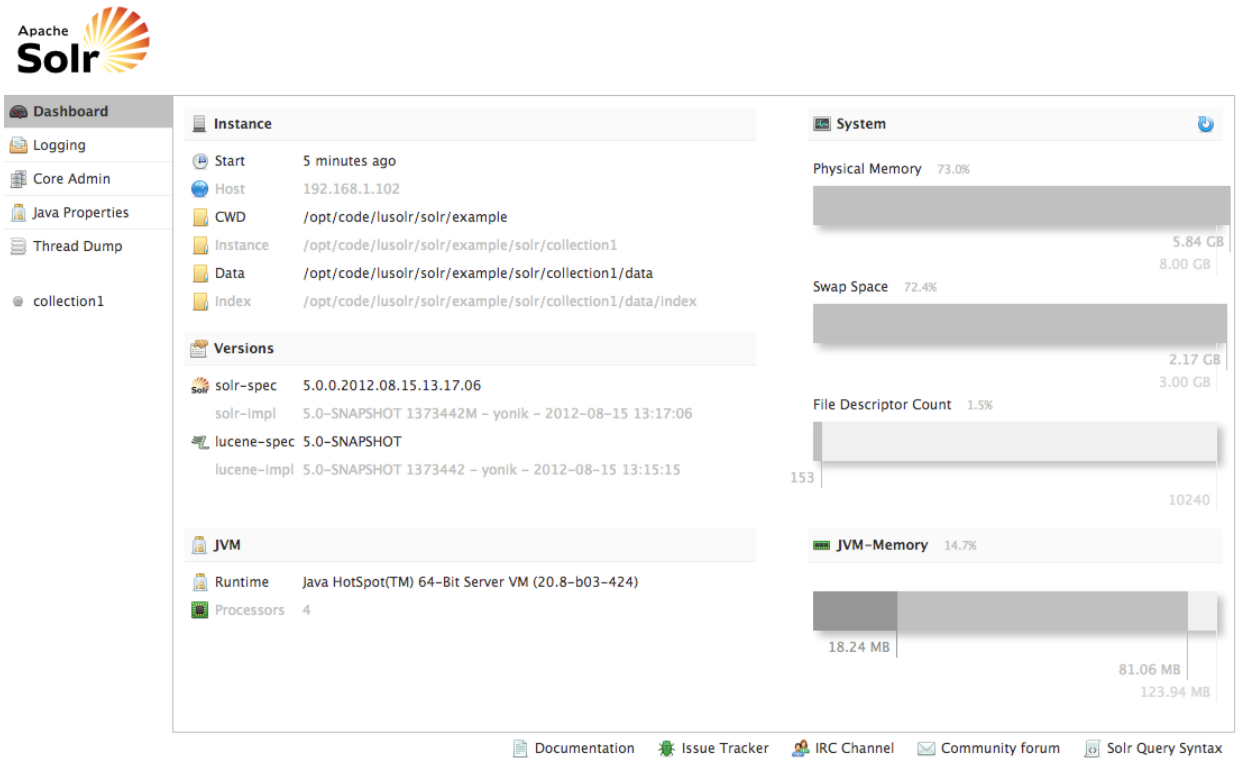

Figure 4 - Solr interface used for querying.

We are also use Weka for data classification. It was developed by the university of Waikato

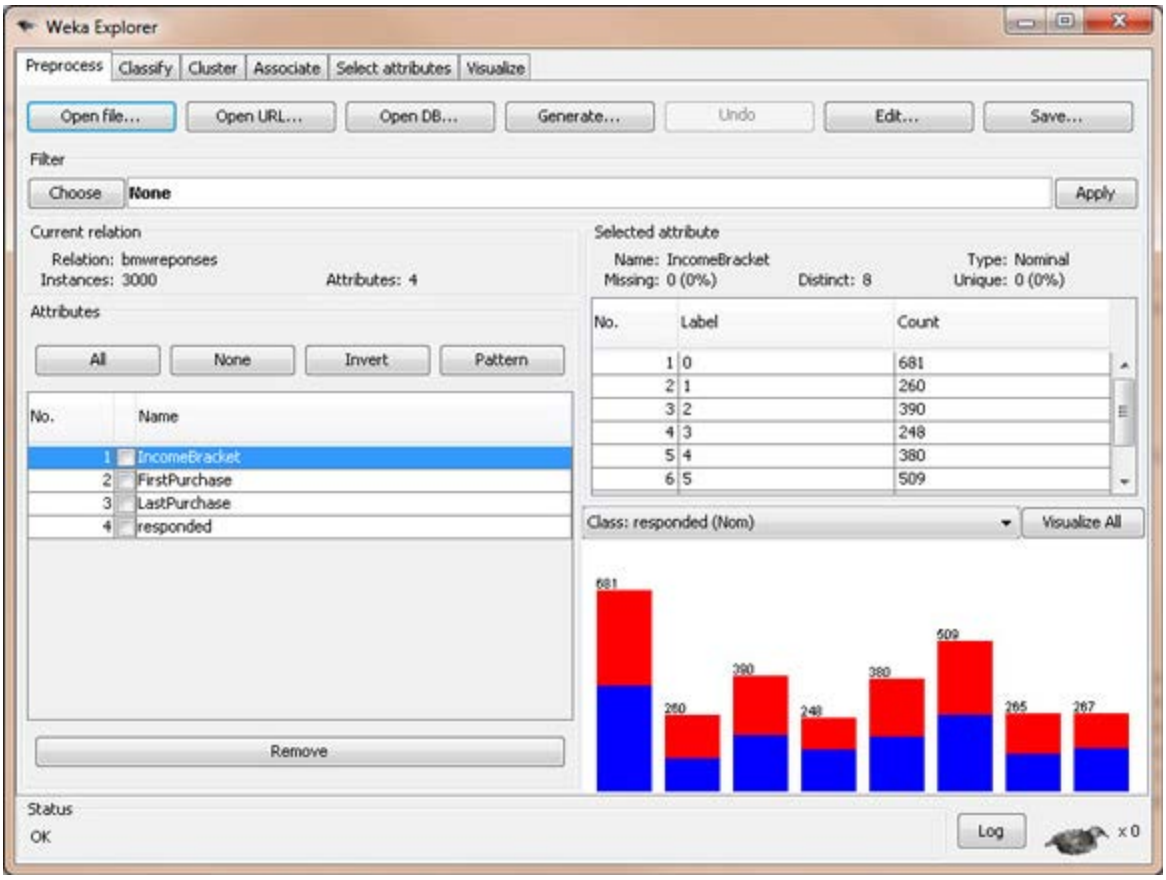

Figure 5 - Weka interface used for data mining.

The user will have an article view, and their will exist tags for every article that the user will not be able to see but will allow the article to be categorized.

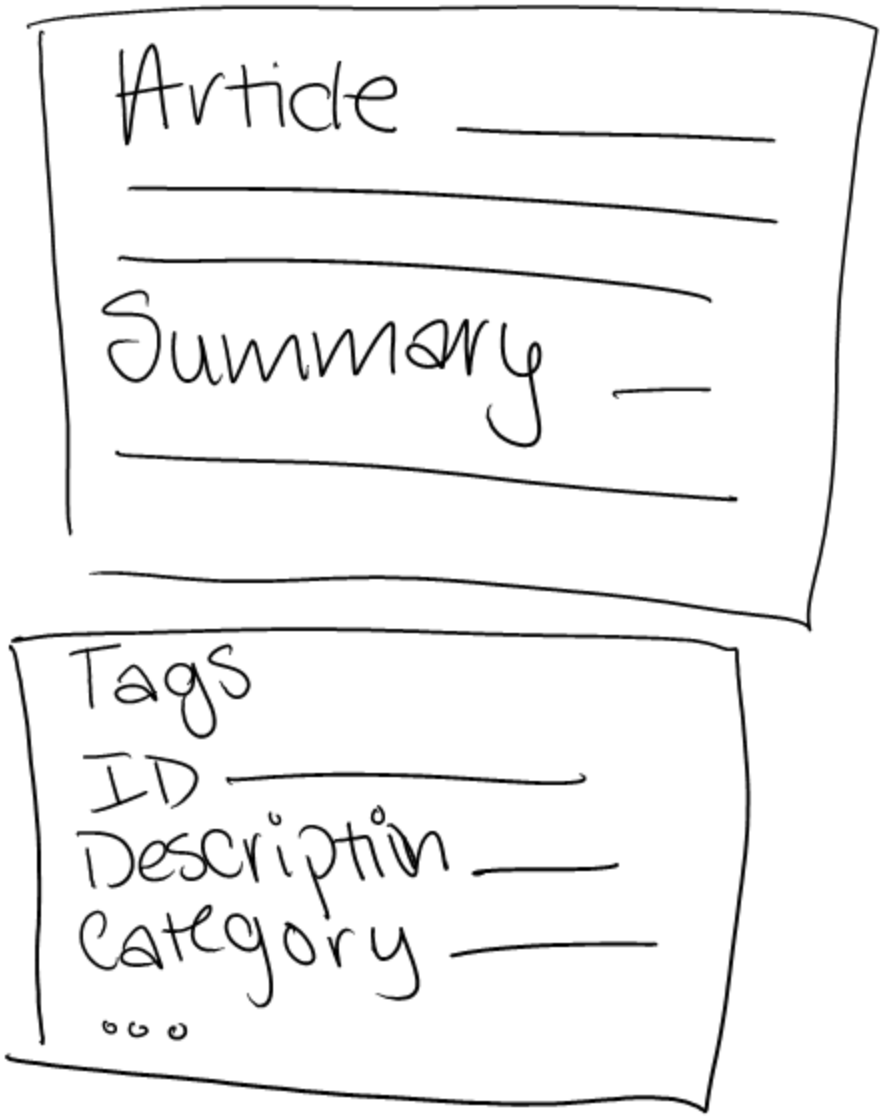

Figure 6 - Article view with “invisible” backend tags

4.1. Implementation

4..1. Programming Languages

Java is the only programming language used in this project. We chose this language over other prevalent languages like C or Python for a couple of reasons. C takes more time to type because the programmer must directly allocate any memory used for the project, but this means that it will hopefully be faster and more efficient because the user manages all the memory. A large potential problem using C is memory leaks, if the developer did not program the application correctly. This means that the application will not reuse allocated memory and can eventually run out of usable memory. Python is typically easier to write than Java, but this tradeoff means that will it will most likely run slower than its Java counterpart [10]. Java seemed to be a good middle ground for ease of writing the code and the speed which it will run. It also helped that the developers have many years of experience writing in Java compared to any other language. The running speed of a program might not be an issue for smaller projects because there is less of a time difference, but if someone chooses to expand upon this project in the future we would like to enable them to make significant changes.

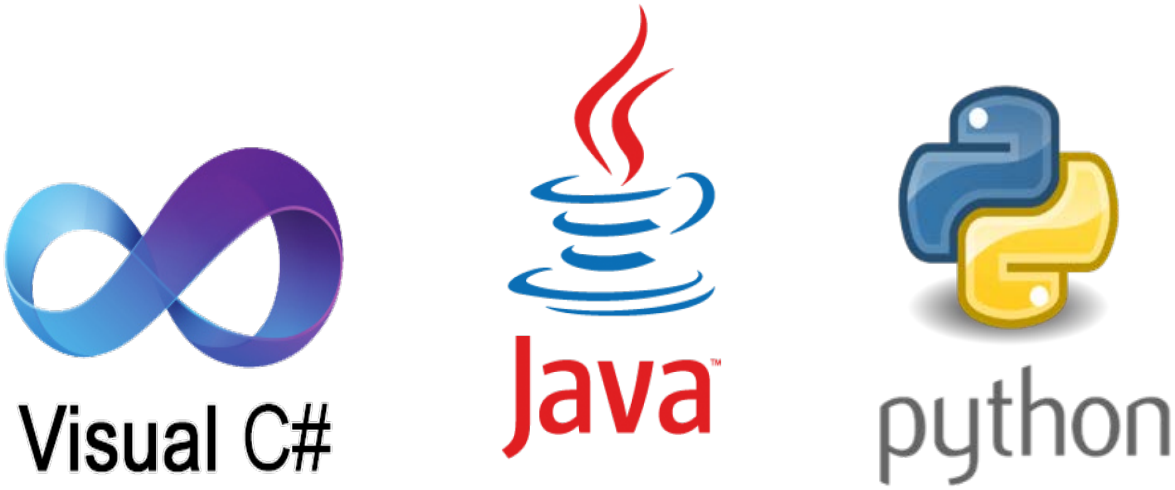

Figure 7 - From left to right this is the typical best run time speed of C#, Java, and Python

Java is platform independent which can be useful for others using or programming this project. Unfortunately, compared to other languages, programmers are encouraged to use Object Oriented Programming when writing in Java which can take more time to write. However, it will be much easier for future developers to understand what is going on in the code and to work on it immediately.

4..2. Tools and Libraries Employed

We have already introduced the tools we will be using, Weka, Solr, and Fusion. They are all Java based which is complementary to us programming in Java. Tools that are Java based are were written and created using Java.
Since we are using Java we will limit ourselves to using the built in libraries Java provides.

5.1. Code Repository Plans

We will not be using a program to manage commits. There are a limited number of personnel working on this project so there is little likelihood that multiple people will attempt to write code at the same time. Every person is working on different parts of the project so even if people are working on the project at the same time, there is no chance that someone will overwrite another member's work, or that their code will be impacted by code updates. Since this is not a massive project in regards to the number of people working on it we will only host on local machines, and every individual will have their own local copy of code. The final results of the program will be stored on a separate server.
We do not need to worry about many security issues for the project. The largest problem we could encounter is a user accidentally or intentionally interacting with the source code for how the program works. Since the project will be stored locally the user who changed the code will not impact any other users eliminating the need for login credentials. The server should handle any unauthorized accesses or changes, eliminating the responsibility of security for our program. Security is normally a major issue, but this program will not contain any sensitive data nor will it register users who use it so there is no need to worry about security.

Figure 8 - Security is a major issue for any project

5.1. Phases

4..1. Text and Attribute Extraction

We need to write algorithms to extract the text and any relevant attributes to be able to categorize articles. All other summarization goals are dependent on finishing this phase.

4..2. Summarization

We will provide a brief summary for each article so that if the title is not adequate for a user they will be able to read the summary as well. We will alter the Fusion template to allow the summary to appear on the same page as the article title and category.

4..3. Indexing Documents

We will create a way of sorting and identifying the documents so that we can access them in a manner of our choosing.

4..4. Testing

This phase will require many hours of manual testing to ensure that the algorithms work correctly. We will also step through the program to ensure that it is reacting correctly and meets all of our project specifications.

5. Prototyping

5.1. Classification with Weka

In order to fully utilize implementation of fusion, we need to begin by classifying the news articles for the purposes of generating summarization for each article based on its categories. Each file must consists on of the following category as show in the table below:

Table 1 - Categories

Art
Economy
Politics
Social/Society
Sports

In order to classify the collection of articles, we need to choose a random sample to build our feature set. We are given a random sample (Thanks to Tarek) of 2,000 articles, where each categories consists of 400 articles. In order to build our feature set, we will first need to collect featured words (or bag of words) from all of the articles, (unique words, and elimination of stopwords). An example can be seen in the table below:

Table 2 - Sample Header of Feature Set

id	label	Apple	Car	Phone	Computers	Languages	Java

Suppose our training set consists of articles regarding to technology, we know that each article has a specific label, such that for each words in the article, if it triggers a word that exists in the feature said, a boolean will be placed in the cell in respect to the word. For example, in the table below, continueing from table 2, suppose we have an article about an Apple product review. It is expected that Apple, Phone and perhaps computer will have a boolean flag. (Suppose we have a technology category)

Table 3 - Sample overview of features for Apple Review

id	label	Apple	Car	Phone	Computers	Languages	Java
apple_review	technology	1	0	1	1	0	0

As we continue to do this for the 2,000 articles with it's appropriate label, we can begin to train using various classification models, including SMO (SVM), NaïveBayes, and Random Forest, each using 10-fold cross-validation.

Table 4 - Average F1 Measure per Model

SMO	NaïveBayes	Random Forest
84.38%	79.31%	77.17%

We have opted to use SMO as it provided higher results for classifying labels correctly. After confirming the selection of our model. We can now begin to classify our dataset of ~120,000 articles using SMO. However, as previously stated, we need to extract out bag of words from each articles and flag the booleans to begin classification. Once all the articles have been labeled, we'll begin to put together the results to form a summary of the article.

5.2. Bringing it together

For each, article we are given a CSV file that contains the category, as well as other information that has been extracted using NER to collect entities, LDA to collect articles topics, titles, and author. Below is a screenshot of a sample file meeting these criteria.

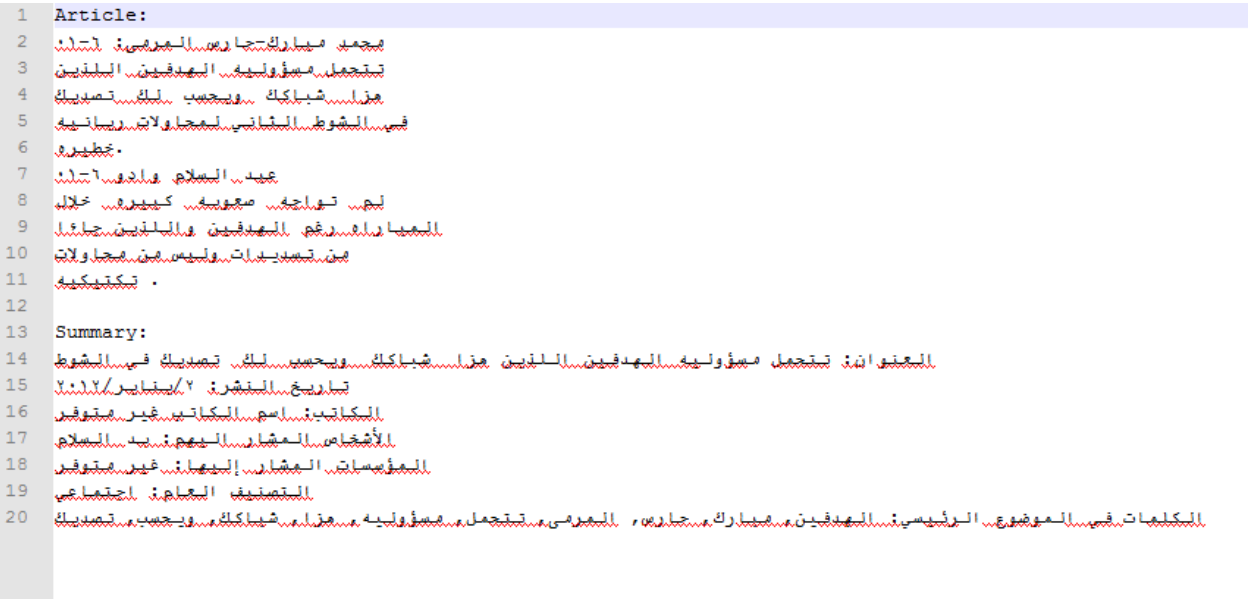

Figure 9 - Sample result of Summarized Article

5.3. Fusion

Once we have collected our summaries for our articles. We can beging importing them to Fusion. Fusion has a very simple UI that allows us to import a persistence and it will automatically index the article on it's own. After importing our local persistence to fusion, we can begin searching; below we can see some sample result:

Figure 10 - Sample result of result in Fusion

We have modified the schema to allow adding and removing some fields by adding an extra field of summary and removing unnecessary field such as source link. We have also helped test the new interface for Fusion.

6. Testing

We have done various form of testing to help ensure stability of our application. For functional testing, we have tested the schema file modification and extracting the text files to XML file. we also did a functional and unit testing for indexing to make sure that everything searches properly. We have done majority of our integration and usability testing on our interface to make sure that everything integrates well and is approachable.

7. Timeline

The team plans to work on the project consistently until the end of the semester in May 2015. We will meet every Wednesday afternoon for two hours in Torgeson to discussed reviewed work and to continue developing the project. We are using an Agile program development style where we continually change and update the project to fit any problems, design needs, or deadlines. This is necessary with our continual feedback from the professor and client. If we were to use a Waterfall style method we might not be able to use our client's and professor's feedback. This style is very sequential and once we finish a portion of the project we cannot make any changes later on. We have already outlined our proposed timeline on how we plan to complete this project in time.

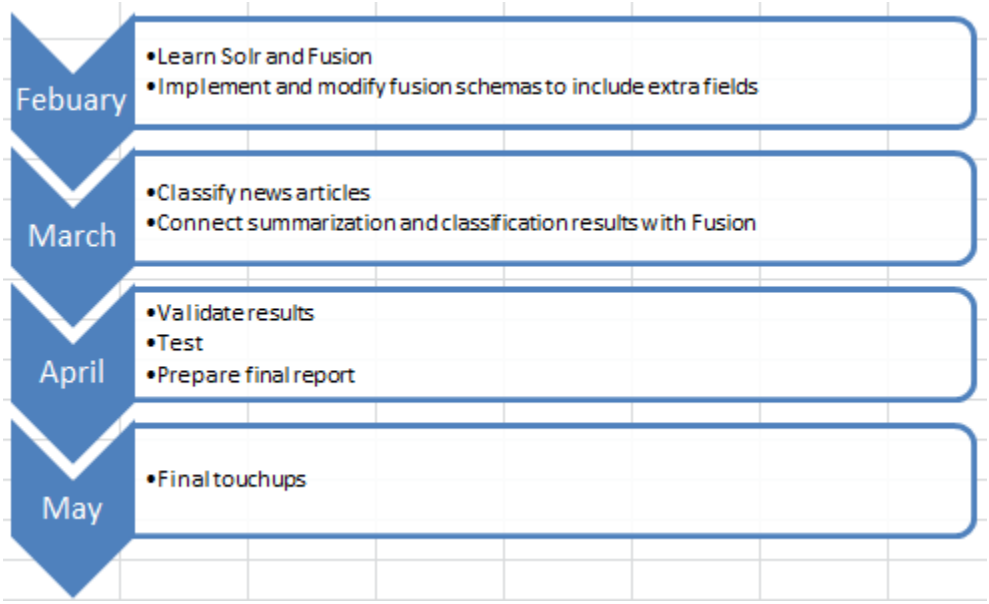

Figure 11 - Timeline of the implementation of the project

The developers will learn Solr and Fusion by reading the companies' websites. Afterwards, we will then try to make small programs using what we learned. At a point where we are comfortable using this technology we will implement these tools in our project. Fusion has a predefined template, which we will need to modify to allow us to include extra fields. To do this efficiently we will need to understand the underlying architecture driving this tool. March requires us to classify news articles so we can determine which algorithm will be most accurate

to minimize any categorization issues. We will be displaying a summary along with the classification results, which will require more Solr and Fusion manipulation. Result validation will take a large amount of time since we currently do not have any computer related automation. This means developers will manually sift through a sample of the data. It will be immediately apparent if the summarization and categorization categories display properly so there will be no need for further testing.

8. Lessons Learned

So far we have been able to keep to the timeline. All work that our contract stated which had to be finished by May is complete. We encountered various problems developing this system. A student who was supposed to help develop tags was unable to aid us. Team members also became sick which meant that some of the team meetings had to be held online. To keep to the timeline the team worked extra to cover any deficits in other people's work. Even though a student was unable to help us with some work we still kept to the schedule. The timeline states the work we have left, integrate Fusion summarizations and debug the project.

9. Conclusion

We have gotten everything together and working smoothly as per requested from the client. The result are as expected. The interface runs seamlessly and shows the results based on the search criteria. The application is up an running on the clients machine and has been tested. The documents that were parsed have been imported into Fusion and indexed, along with the modified schema file which should now show the results of the extra fields.

10. Acknowledgments

We would like to acknowledge Dr. Edward Fox, the class professor, for his guidance throughout the class. We also would like to thank our client and mentor Tarek Kanan for all his help the creation of this project. He can be reached at tarekk@vt.edu. A special thanks goes to Lucidworks, the Fusion creator company, for answering some of our questions and for guiding us through the Fusion part of this work. This work was made possible by NPRP grant # 4-029-1-007 from the Qatar National Research Fund (a member of Qatar Foundation).

11. References

[1] LDA http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
[2] NER http://en.wikipedia.org/wiki/Named-entity_recognition
[3] Weka http://www.cs.waikato.ac.nz/ml/weka/
[4] LucidWorks Fusion http://lucidworks.com/product/fusion/
[5] Solr https://wiki.apache.org/solr/
[6] PDF Parsing http://www.pdfparser.org/
[7] Text Summarization http://en.wikipedia.org/wiki/Automatic_summarization
[8] Java BufferedReader http://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html
[9] Python Codecs https://docs.python.org/3/library/codecs.html
[10] https://www.python.org/doc/essays/comparisons/

Resources used:
http://home.adelphi.edu/~siegfried/cs480/ReqsDoc.pdf
http://wwwis.win.tue.nl/2R690/projects/spingrid/srd.pdf

Figures:
[1]http://image.slidesharecdn.com/meetsolrforthefirsttimeagain-141013005741-conversion-gate02/95/meet-solr-for-the-tirst-again-18-638.jpg?cb=1413179929
[2]http://heliosearch.org/wp-content/uploads/2012/08/solr_admin.png
[3]http://www.ibm.com/developerworks/library/os-weka2/weka-data5.jpg
[8] https://users.soe.ucsc.edu/~kunqian/logos/csharp.png
https://lh4.googleusercontent.com/-0fbEaUy0zgs/Uycq_aNlvHI/AAAAAAAAG5s/vvq4LmANws0/s0/java-logo.png
http://www.blancocuaresma.com/s/static/images/python-logo.png
[9]http://www.valiantsolutions.com/images/infosec.jpg