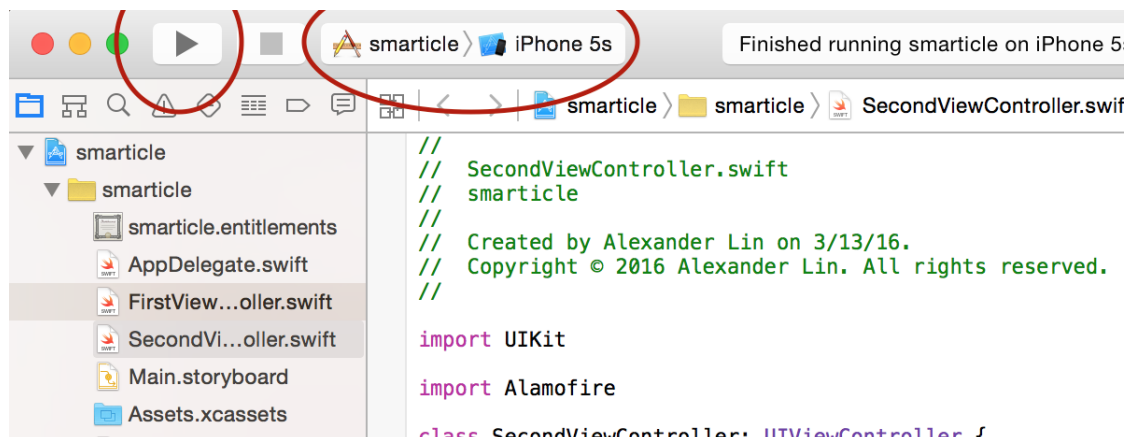# sm@rticle

Alex Lin

April 3, 2016

Overview: sm@rticle is a Swift-based iPhone application developed by Alex Lin that smartly generates short summaries of news articles. It mainly utilizes an unsupervised machine-learning algorithm similar to Google's PageRank.

1 Setup Instructions

1. Pull the sm@rticle repository from the following link: https://github.com/al5250/smarticle. You should find the following files:

2. Open the file "smarticle.xcworkspace" with Xcode. Make sure the application is set to "smarticle" in the top left corner. Run the project with the big "Play" button.

3. The application will now load. Notice that there are two main functions - "Search" and "Browse". "Search" will be the default tab selected.
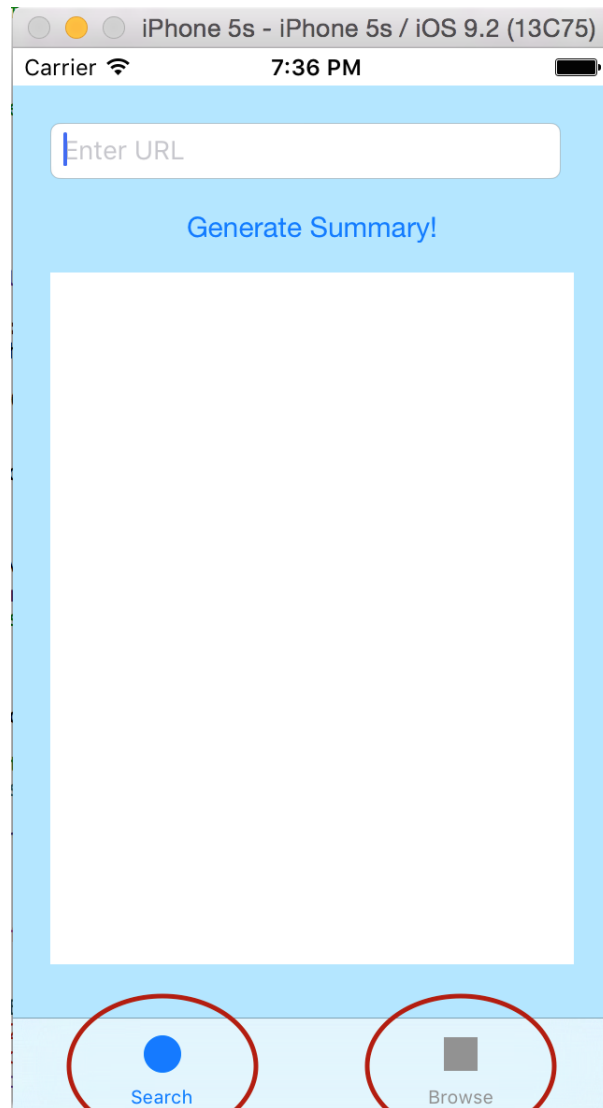
2   Main Features and Functions

There are three main components of sm@rticle - 1) "Search", which generates summaries for specific articles; 2) "Browse", which allows you to search for articles by keyword, and 3) "ShareURL", which is a share extension that allows you to pass in an article's URL from a web browser such as Safari.

2.1   Search

"Search" works in a very self-explanatory fashion. Simply enter a URL in the indicated text box and hit the "Generate Summary!" button. If you enter in a valid URL, then the app will construct a summary of the article consisting of four sentences and display the title and this summary in the text field below. Here is an example:
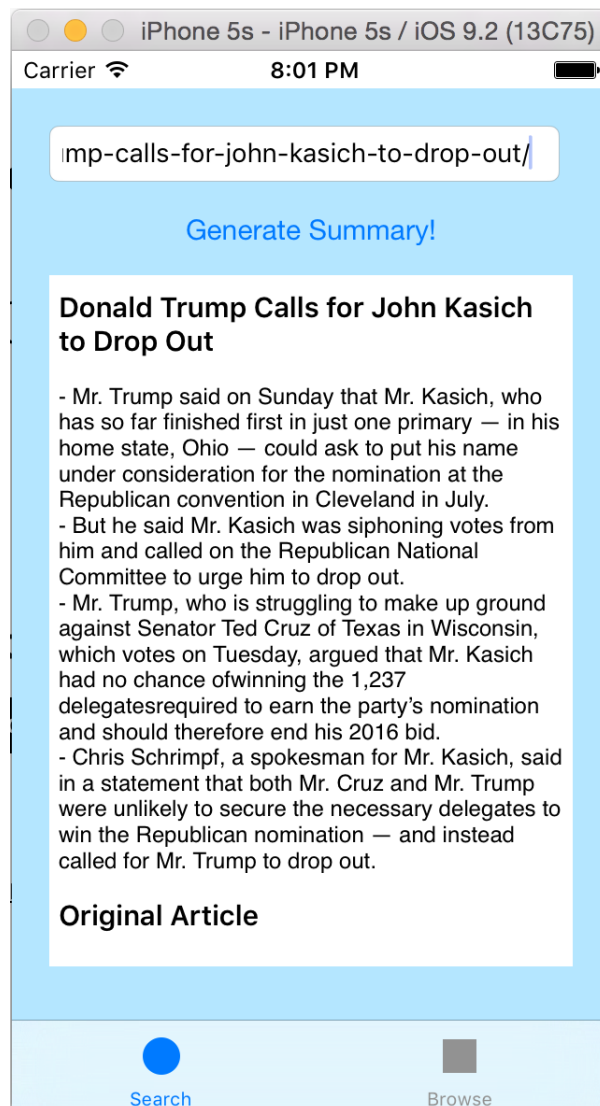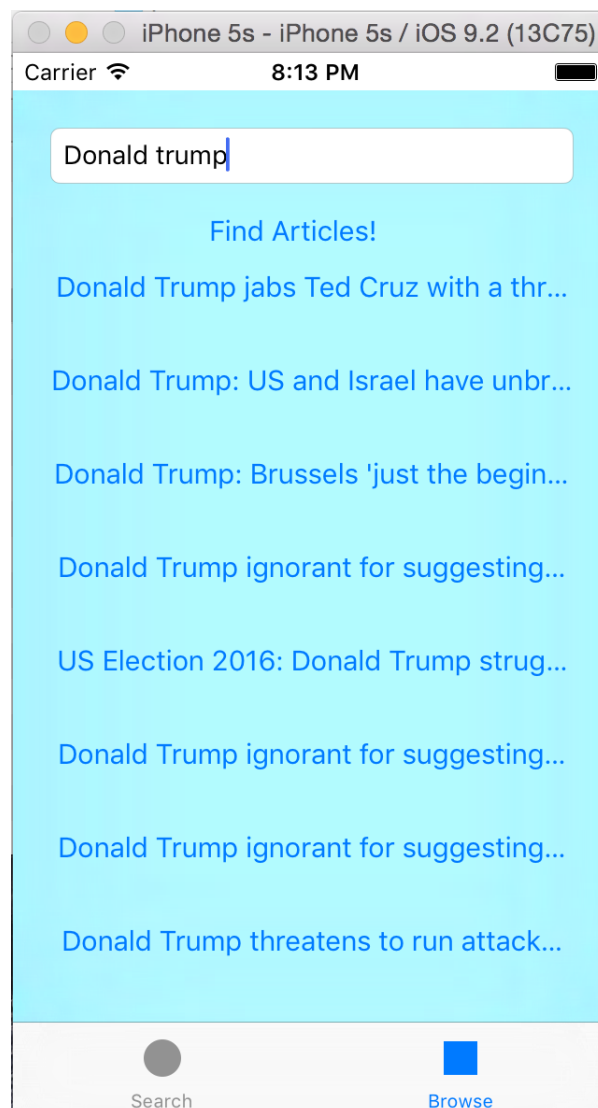

If you want to read the original article, simply scroll down and you will find the exact wording of the website. Conversely, if the text you entered in the text box is not a valid URL, then you will be prompted with the following message:

Article Not Found

Please enter in a different URL.

OK
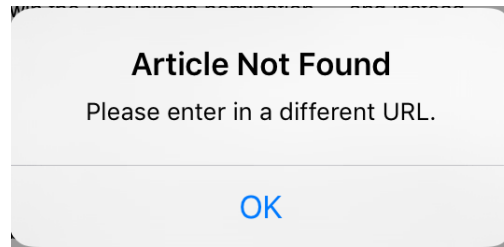
2.2   Browse

"Browse" is the second tab of sm@rticle, and it allows you to browse for articles by their title. I implemented "Browse" as an additional bonus objective. Simply enter in a key word or phrase into the textbox, and hit the button "Find Articles!" The names of up to 8 articles will appear as buttons on the screen.

iPhone 5s - iPhone 5s / iOS 9.2 (13C75)

Carrier   8:13 PM

Donald trump

Find Articles!

Donald Trump jabs Ted Cruz with a thr...

Donald Trump: US and Israel have unbr...

Donald Trump: Brussels 'just the begin...

Donald Trump ignorant for suggesting...

US Election 2016: Donald Trump strug...

Donald Trump ignorant for suggesting...

Donald Trump ignorant for suggesting...

Donald Trump threatens to run attack...

Search   Browse

Click on any one of these buttons and you will be automatically redirected to "Search", with the URL for the corresponding article filled in the text box. Hit "Generate Summary!" to get the summary of this article.

If your initial query fails, you will be prompted with the following message:
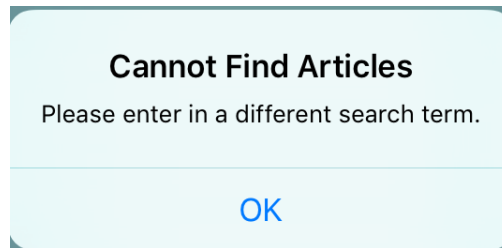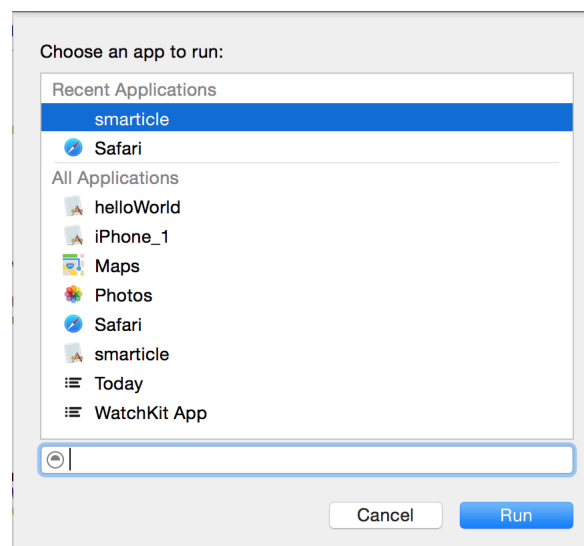

2.3   ShareURL

ShareURL is a share extension that is meant to get the URL of an article from a browser (such as Safari) and pass that URL into sm@rticle. Unfortunately, in order to allow data flow between ShareURL and sm@rticle, I needed to create an app group. This is only possible with a full developer account from Apple, which I could not afford to purchase. Thus, the current implementation of ShareURL only copies the URL of the page to the clipboard. The user can then paste the URL to the textbox under "Search" in sm@rticle.

To use ShareURL, go to the top left corner of Xcode and switch the current application from "smarticle" to "shareURL".
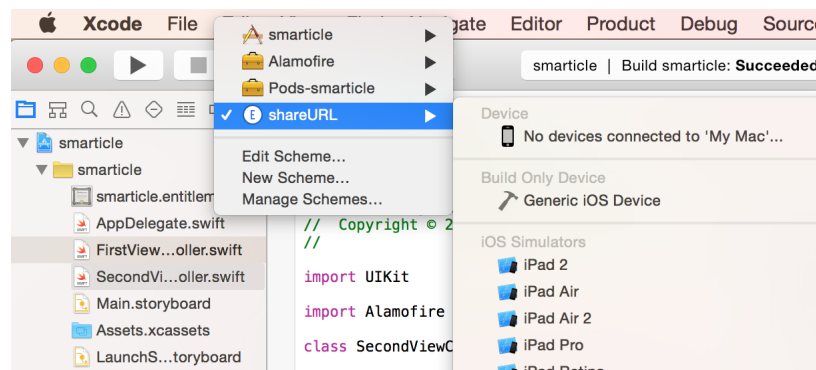
Then, hit the "Play" button. You will be prompted with the following options list. Choose "Safari" and hit "Run".

Using Safari, find an article that you want to summarize. Then, hit the "Share" button at the bottom of the screen. Scroll through the top row until you find "shareURL", and click on it.
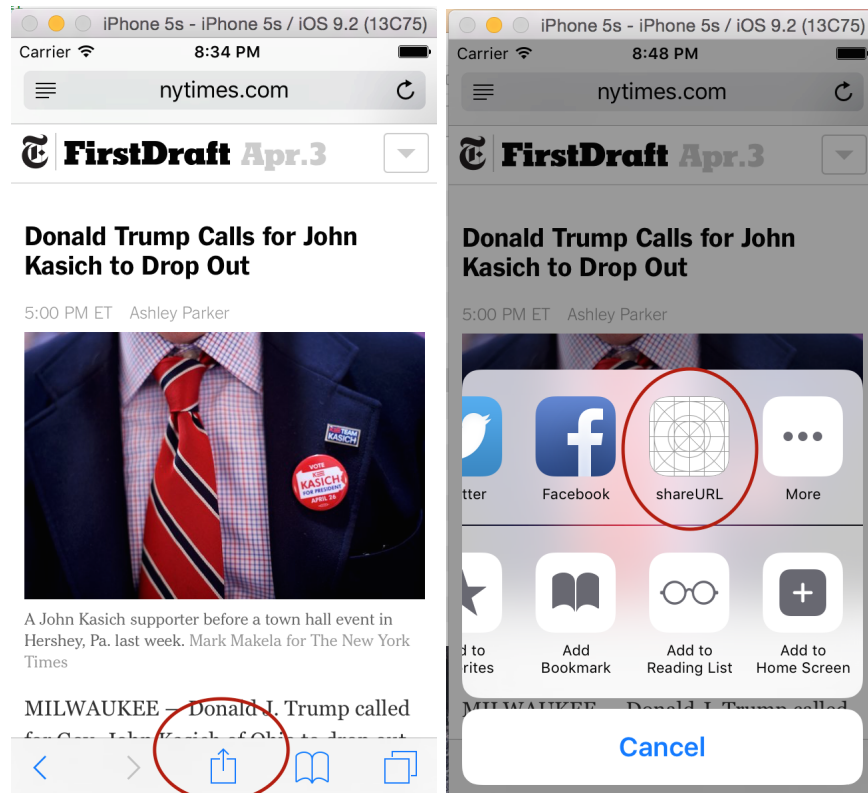
You will be prompted with the following screen. Hit "Post" to post the URL of the article to clipboard.
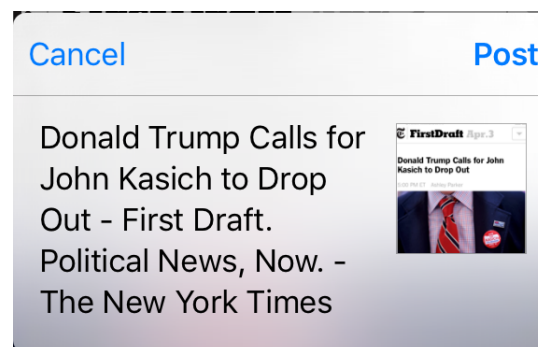
Now, hit the home button on the simulator. This is equivalent to pressing the keyboard shortcut "Command-Shift-H". Launch the sm@rticle app and paste the URL in the textbox of "Search". Next, hit "Generate Summary!" to get the summary of your desired article.
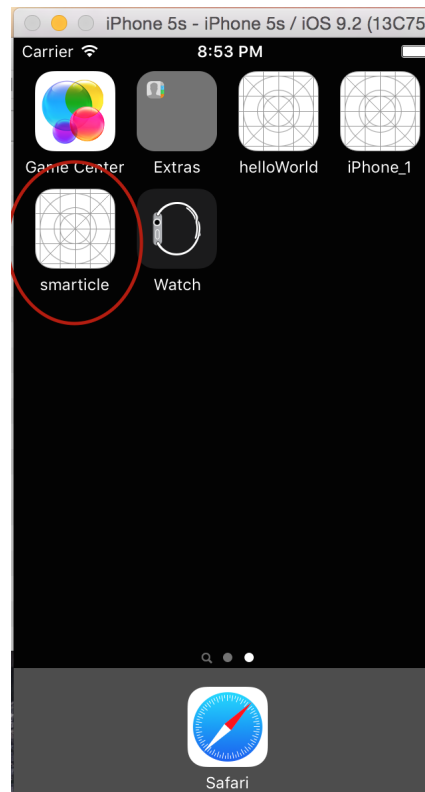
3   Solution Explained

3.1   FirstViewController

FirstViewController.swift contains the code for the "Search" tab. A good chunk of this code is dedicated towards implementing the PageRank-based machine learning algorithm for constructing the summary. I used this research paper (`https://web.eecs.umich.edu~mihalcea/papers/mihalcea.emnlp04.pdf`) as the main source of ideas for the algorithm. See Section 3.3 - Theory Behind the ML Algorithm for more information on what I did.

The rest of the code is dedicated towards processing/displaying the text of a news article. First, I used CocoaPods to integrate an open sourced networking library called AlamoFire into my project so that I could perform GET requests. I could then feed a user-entered URL into the Dandelion API (`https://dandelion.eu/`) to extract the source text of the article. I then cleaned the source text to get the sentences into a format that is friendly for my machine learning algorithm. Next, the algorithm spits out a ranking of sentences according to how well they represent the content of the article. The top four sentences are chosen as a summary and printed on the screen, along with the original source text.

3.2   SecondViewController

SecondViewController.swift contains the code for the "Browse" tab. When a user enters some text to be queried, I submit a GET request with AlchemyAPI to return the titles and URLs of articles containing the search term(s). I then dynamically generate buttons on the screen with the article titles. I keep a global array of URLs so that when the user selects an article, the corresponding URL is fed into FirstViewController. The user can then generate a summary/view the original text of the article.

3.3   Theory Behind the ML Algorithm

The algorithm I implemented for determining the best sentences to represent the article is based on Google's PageRank, which uses a very similar, unsupervised concept. We can picture the article as a weighted complete graph, with each sentence being a vertex and edges between every pair of vertices. The initial weight of a vertex is not important, so we standardize them all to be equal. However, the final weight of a vertex will tell us how well the corresponding sentence represents the ideas of the article. The edges also have weights. The weight w_e of an edge $e = (u, v)$ is determined by a similarity function on the two corresponding vertices (or sentences) u, v. Let S be the set of words that are in sentence u and sentence v. Let sentence u have length n_u words and sentence v have n_v words. I then simply defined w_e (aka the similarity between u and v) as

$$w_e = \frac{|S|}{\log n_u + \log n_v}$$

Note that this does not have to be the similarity function. I also tried using other similarity functions that were suggested by the research paper, such as cosine similarity and longest common subsequence. Nevertheless, upon conducting a brief experiment on 50 articles with these three types of similarity functions, I found that the one I mentioned above seemed to generate the best summaries. I stored the similarities between each pair of edges in a matrix. I then used another function to calculate how representative each sentence was of the article by how significant that sentence's edge weights were in comparison to other sentences' edge weights. I repeatedly applied the function to these vertices weights' until the weights converged on values that were within a 0.001

precision of the weights of the previous iteration. The vertices/sentences with the top four weights were then selected to summarize the article. The intuition behind the algorithm is, as follows: Sentences that end up having high weights share a significant percentage of words with many other sentences. Thus, it is likely that these highly weighted sentences are the best representation of the content of most of the sentences in the article. Hence, they make a good summary. This is the key concept behind the sm@rticle software.