COS 435 (Spring 2012) Final Project
Tonify: The Tone-Conscious Search Engine!

Diego Perez-Botero
diegop@cs.princeton.edu

1. MOTIVATION

The Internet has become the main information source for all of our information needs. It is the place where academic research, casual browsing, marketing, and many other use cases come together in the search for useful content. One of the Internet's advantages over traditional mediums, the variety of its data, generates a challenge for web search engines such as Google and Bing: we are dealing with heterogeneous data sources (e.g. bulletin boards, blogs, encyclopedias, news stories), all of which may contain valuable information for a specific query. Even though prioritizing web pages by the amount of traffic that they handle and other connectivity properties has proven to be a viable solution (e.g. PageRank), we are not always necessarily interested in the most popular query result, but rather the result that matches a particular underlying tone. For example, search engine users might want to read news stories that match their own point of view about a topic (e.g. against Stop Online Piracy Act) or might value personal opinions more than impartial accounts of an event. The former would be better serviced by documents with negative or contrarian tones; the latter would be benefited by documents with personal (informal) tone.

Queries usually target a specific author tone. Currently, web search engines feed us documents that match the general public's interests. We then proceed to filter out the unwanted tones by looking through the search results ourselves. This approach does not only delegate part of the data processing to the user; it also leads to suboptimal result rankings, given that the user's target tone is never taken into account when ordering the result list. There has to be a better way to return smarter results without the need of extensive personalization data for each user.

2. OBJECTIVES

Our goal is to develop a search engine that *prioritizes* documents by their underlying tone (e.g. negative or positive, happy or upset, formal or informal). Instead of just *filtering by tones*, we want the target tone to affect a document's score with respect to each specific query. Last but not least, a document's *degree* (e.g. markedly positive, moderately positive) of each desired underlying tone must play a part in its ranking. That is, tones are to be treated as more than a boolean property; they must have different degrees to them.

Our objectives are as follows:

- Find an effective method to accurately detect a document's underlying tone.
- Define an enhanced scoring function that takes tone *degrees* into account.
- Implement a prototype that makes use of the proposed scoring function.
- Evaluate the quality of the results against a standard search engine solution.

This is an experimental project. Its main contribution is a proof-of-concept solution for a tone-conscious search engine that integrates current document scoring and tone classifying techniques.

3. DESIGN

The first step towards a tone-conscious search engine is to investigate the state-of-the-art of document tone detection. This is a theory-heavy topic in which a valuable contribution on our part is ruled out due to time constraints and lack of experience in the field. We limit ourselves to finding a currently viable solution and explaining the basic theory behind it.

3.1 Statistical Text Classification

Given a set of classes (e.g. negative, positive, formal, informal), we seek to determine which class(es) a given document belongs to. This problem is known as *text classification*. For this task, Machine Learning-based algorithms are a good alternative, especially when manual classification is infeasible and hand-crafted rules to decide a text's class are initially unknown. In machine learning, the set of rules or, more generally, the decision criterion of the text classifier, is learned automatically from learning data [1]. We require a set of pre-labeled training documents for each class, so the need for manual classification is not entirely eliminated, but this is arguably easier than writing the rules for each class. A classifier can then be evaluated after being trained by using other pre-labeled datasets, such as Reuters-21578.

The Naive Bayes (NB) learning method can be applied to many different learning problems. An NB classifier can be set up using two possible supervised learning models: the multinomial or the Bernouilli model. A comparison between

the aforementioned approaches is shown in Figure 1. As can be seen, the multinomial model is better suited for classifying web content, given that it can handle longer documents and can take more features into account than the Bernoulli model, which is important when classifying a practically infinite collection of documents.

	multinomial model	Bernoulli model
event model	generation of token	generation of document
random variable(s)	$X = t$ iff t occurs at given pos	$U_t = 1$ iff t occurs in doc
document representation	$d = \langle t_1, \ldots, t_k, \ldots, t_{n_d} \rangle, t_k \in V$	$d = \langle e_1, \ldots, e_i, \ldots, e_M \rangle$, $e_i \in \{0, 1\}$
parameter estimation	$\hat{P}(X = t\|c)$	$\hat{P}(U_i = e\|c)$
decision rule: maximize	$\hat{P}(c) \prod_{1 \leq k \leq n_d} \hat{P}(X = t_k\|c)$	$\hat{P}(c) \prod_{t_i \in V} \hat{P}(U_i = e_i\|c)$
multiple occurrences	taken into account	ignored
length of docs	can handle longer docs	works best for short docs
# features	can handle more	works best with fewer
estimate for term the	$\hat{P}(X = \text{the}\|c) \approx 0.05$	$\hat{P}(U_{\text{the}} = 1\|c) \approx 1.0$

Figure 1: Multinomial vs Bernoulli Model. Taken from [1].

3.2 Document Scoring Function

Let X be a set of document tones that are being prioritized. The enhanced scoring function that we propose is shown below:

$$enhanced\_score(q, d) = score(q, d) * \frac{1}{\|X\|} * \sum_{x_i \in X} tone(x_i, d)$$

Where
$score(q, d)$: original score of document d with query q
$tone(x_i, d)$: score of document d with respect to tone x_i
$score(q, d) \in [0, 1],\ tone(x_i, d) \in [0, 1],\ \|X\| > 0$

Basically, a document's *average tone score* with respect to a query is being multiplied by the document's *base score*. This way, the score differential generated by the document's tone properties is directly proportional to its traditional relevance score. Thus, a document will only have a high *enhanced_score* if it is both relevant to the query terms and relevant to the tones being prioritized. This makes sense, considering that a document that has nothing to do with the topic at hand should not show up as a top search result regardless of how well it matches the target tones. Similarly, a document with a great term-based score is irrelevant to the user if its tone properties are the opposite of those being prioritized.

4. PROTOTYPE

The prototype that was implemented integrates a great variety of technologies, most of which are based on the Java programming language:

- **WebSPHINX**[1]: an open source Java web crawling library.
- **jsoup**[2]: a Java library for HTML parsing.
- **Lucene**[3]: a powerful open source Java search engine library.
- **uClassify**[4]: a web-based text classifier API.
- **Netbeans**[5]**, JEE 6**[6]**, Glassfish Server**[7]: a complete web application framework (JEE 6 as programming language, Glassfish Server as web container, and Netbeans as the Integrated Development Environment).

We decided to use existing text classifiers from the uClassify web service that have already been trained as opposed to training our own classifier, since the training process does not add anything to this work's contributions. Three classifiers were employed:

1. **Sentiment Classifier**[8]: This classifier determines if a text is **positive or negative**. It is based on circa 40000 Amazon reviews from 25 different product genres. The expected accuracy is about 78% (macro-precision: 77% and macro-recall: 77%) running 10-fold cross validation.
2. **Mood Classifier**[9]: This classifier attempts to determine the state of mind of the writer - **upset or happy**. The measured accuracy is 96% (using 10-fold cross validation).
3. **Tonality Classifier**[10]: Determines the tonality of a text - **corporate (formal) or personal (informal)**. No data was found regarding its accuracy.

Figure 2 shows the component diagram of the Tonify prototype. It is divided into two main components: the Custom Web Crawler and the actual Tonify Web Application.

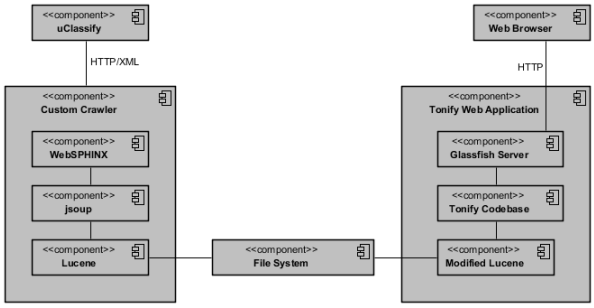


Figure 2: Tonify Prototype's Architecture

4.1 Custom Web Crawler

Some WebSPHINX source code was modified so that it could run extra processing tasks after fetching each web page. The basic workflow for this component is as follows:

1. WebSPHINX crawls the web using Breadth-First Search starting from a list of source nodes. A maximum depth is set, which is used as the termination criterion.

[1]http://www.cs.cmu.edu/ rcm/websphinx/
[2]http://jsoup.org/
[3]http://lucene.apache.org/core/
[4]http://www.uclassify.com/
[5]http://netbeans.org/
[6]http://www.oracle.com/technetwork/java/javaee/tech/index.html
[7]http://glassfish.java.net/
[8]http://blog.uclassify.com/sentiment-api/
[9]http://www.uclassify.com/browse/prfekt/Mood
[10]http://www.uclassify.com/browse/prfekt/Tonality

2. Every time a page is fetched, its content is parsed with the jsoup library, which gets rid of scripting code (e.g. javascript, dhtml) and removes HTML tags. That way, only the real human-readable content is preserved.

3. The human-readable content is sent to the uClassify web service via an HTTP/XML request. The HTTP-XML reply from the classifier API indicates the score of the sent text with respect to each of the tones (i.e. negative, positive, happy, upset, formal, informal), from 0 to 100. We also extended this functionality with an experimental Keywords API, which gives a list of the keywords that contribute the most to each of the tone scores. For example, a Positive tone score can be attributed to words like "good", "great", "excellent", etc.

4. Lucene is used to store each page's parsed content alongside with the tone score and tone keywords metadata. The Lucene index is stored in a traditional file system instead of inside a relational database, as Lucene provides all of the required data management capabilities (indexing, scoring, searching, etc).

4.2 Tonify Web Application

The Custom Web Crawler uses a normal Lucene binary to generate the web page index. On the other hand, the Tonify web application does require changes to the Lucene project's source code in order to support the enhanced scoring function when evaluating a query. The basic workflow for this component is as follows:

1. A Tonify user accesses the service through a web client. The user interface is shown in Figure 3. The user types in the query and indicates the tone that is to be prioritized (if any) from each of the classifiers (Sentiment, Mood and Tonality).

2. The Tonify web application receives an HTTP request with all of the query information. It then evaluates the query on the index that was generated by the Custom Crawler component, this time with a modified Lucene that uses the scoring function given in Section 3.2.

3. A result list is presented to the user showing every document's base score with respect to the query, as well as the appropriate tone scores and their respective keywords.

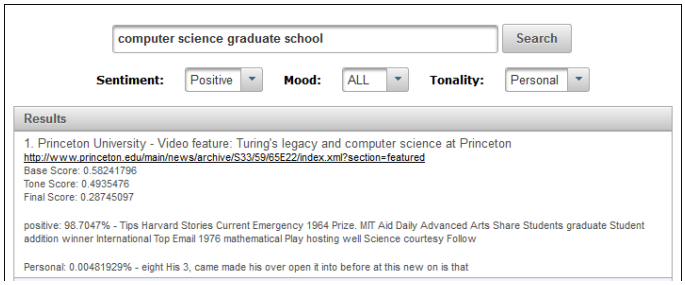


Figure 3: Tonify Prototype's User Interface

4.3 Source Code Repository

The prototype's source code is publicly available at: https://github.com/tonify

5. EVALUATION

Evaluating the Tonify prototype was no easy task, considering that no datasets were found combining sample query scores with classification scores. In addition to many hours of web browsing, all of the dataset sources listed in the Final Project Resources page[11] were taken into account without any success. Therefore, there was no way for us to test our system against pre-made correct result sets to specific queries.

As mentioned in Section 4, two of the three classifiers already have accuracy statistics, so there is no point in testing their accuracy as part of the prototype's evaluation. Furthermore, the classifiers being used are not a key part of this project and can be easily replaced in the future. Similarly, Lucene's default scoring function combines the Boolean Model and the Vector Space Model [2], which are well-studied traditional algorithms, so it does not warrant a separate evaluation of our own.

A meaningful evaluation of the Tonify prototype is to compare its results against those of unmodified Lucene with simple tone-aware queries on an acceptably large dataset. Hence, that will be our focus for the rest of this section. As will be described in Section 5.2, we drew inspiration from Problem Set 2[12] to design our own experiments.

5.1 Base Dataset Creation

Given that we do not have the resources to conduct a full web crawl on the internet, four different source nodes covering a wide spectrum of content types were used in a web crawl with a maximum depth of 2 to generate a sizeable index:

- **Princeton University Web Site**[13]: a good example of an institutional web site and covers a large variety of topics, given that it contains links to various departmental web sites.
- **Encyclopaedia Britannica**[14]: a formal, reliable source of information about almost any field of study.
- **CNN**[15]: a popular news web site containing stories about current events taking place all round the globe.
- **Cheezburger**[16]: a humorous informal web site that acts as a hub to many blogs, videos, images, bulletin boards, etc.

After several hours of web crawling, the base Lucene index was ready. It contains 6903 highly heterogeneous web pages.

[11] http://www.cs.princeton.edu/courses/archive/spring12/cos435/project_resource_list.html
[12] http://www.cs.princeton.edu/courses/archive/spring12/cos435/Probs/ps2.html
[13] http://www.princeton.edu/
[14] http://www.britannica.com/
[15] http://www.cnn.com/
[16] http://www.cheezburger.com/

5.2 Methodology

We will be evaluating the performance of Tonify by using unmodified Lucene as a point of reference. The three queries employed for this purpose are:

1. "French Presidential Election 2012" with Personal (Informal) Tonality as target tone. (Codename: **France**)
2. "iPad 3 Review" with Negative Sentiment as target tone. (Codename: **iPad 3**)
3. "United States Economy" with Upset Mood as target tone. (Codename: **USA**)

As can be noticed, each query deals with a different uClassify text classifier (Tonality, Sentiment and Mood - see Section 4). This is done to prevent a misbehaving classifier from sabotaging more than one result set. Mirroring Problem Set 2, we label each search result with one of three degrees of relevance: Irrelevant, Simply Relevant or Highly Relevant. Their definitions will be:

1. **Highly Relevant**: a search result that posses the target underlying tone AND talks about the topic being references by the query.
2. **Simply Relevant**: a search result that talks about the topic being referenced by the query, but carries a tone different to the target one.
3. **Irrelevant**: a search result that does not address the topic being referenced by the query, regardless of its tone.

The **Pooling** and **Scoring** rules remain almost the same as described in Problem Set 2's specification[17]:

- **Pooling:** The first 20 results from each search engine are taken. Duplicates are removes, and each result is visited to decide relevance.The size of the pool is recorded (number of unique results produced by the combined results 1 - 20 of each search engine). The number of "highly relevant" and "simply relevant" results in the pool are also recorded.
- **Scoring:** After constructing the pool, we go back and score each of the first 30 results returned by each search engine based on the scoring of the pool. If a result does not appear in the pool, it receives a score of irrelevant. If a document appears twice under different URLs in the list for one search engine, we count it only for its better ranking for that search engine and delete any additional appearances within the same list. In this case there will be less than 30 distinct results returned by the search engine. We do not go back to the search engine to get more results. We keep only what was returned in the first 30, with their original ranks.

[17] http://www.cs.princeton.edu/courses/archive/spring12/cos435/Probs/ps2.html

Nonetheless, this time around **we consider Simply Relevant and Irrelevant results to be *irrelevant* for all but discounted cumulative gain (DCG)**. This change is made because the queries themselves are simple, so Simply Relevant results are expected to be the norm. Furthermore, a real Tonify user would only be interested in results that comply with the Highly Relevant definition. As in Problem Set 2, the measures to be calculated are:

1. The reciprocal rank of the results.
2. Precision at rank 10.
3. The discounted cumulative gain at rank 10, DCG(10), using a gain of 0 for irrelevant documents, a gain of 1 for "simply relevant" documents and a gain of 5 for "highly relevant documents".
4. AVGP(20), the average over the precisions at each point that a relevant document appears in the first 20 (yes 20, not 30).
5. The recall of the relevant documents in the pool within the 30 results.

5.3 Extended Dataset Creation

Taking into account that the base dataset contained 6903 documents that may not have been related to the queries at hand, the initial Lucene index was extended with pages associated with the topics to be queried. To do this, each of the test queries (i.e. **France**, **iPad 3** and **USA**) were sent to Google and URLs were scraped from Google's result lists. Those URLs were then fetched by the Custom Web Crawler, leading to 586 additional pages about **France**, 596 related to **iPad 3** and 693 tied to **USA**.

Note that the results returned by Google were not necessarily relevant under our definitions. Their only role was to make the dataset more interesting. Meanwhile, the base dataset was preserved in order to generate noise for the two applications being tested.

5.4 Results

The document pools that resulted from the methodology used are very different from one another (see Table 1). While codename **France** has a negligible amount of Irrelevant documents and many Simply Relevant ones, **iPad 3** ended up having more Irrelevant documents than Highly Relevant ones. In the case of **USA**, Highly Relevant results are more numerous than Simply Relevant ones and the smaller pool size (29) indicates that both search engines shared many results in common.

Table 1: Web Document Pools Employed

	Query		
	France	iPad 3	USA
Pool Size (after duplicate removal)	34	35	29
# Highly Relevant in pool	13	7	10
# Simply Relevant in pool	20	18	6
# Irrelevant in pool	1	10	13

The "French Presidential Election 2012" query with Personal (Informal) Tonality as target tone returned three very similar measures for Lucene and Tonify in terms of "Precision at Rank 10", DCG(10) and AVGP(20). However, Tonify clearly outperforms Lucene when it comes to the Recall and the Reciprocal Rank. This means that Tonify's results list ends up containing more Highly Relevant documents overall, and it also shows that the first of those results appears earlier than in Lucene's result list. Thus, the enhanced scoring function partially improved the quality of the results for the **France** query.

Given that the French Presidential Election just took place, you would expect most of the coverage for that event to be coming from news sites, which usually do not contain articles with an informal tone. The Highly Relevant results were mostly blog posts and Twitter feeds, which tend to have a lower query term frequencies than news articles due to their shorter length. They also do not include links to other related news stories, which fictitiously increase a news site's traditional query-related score. For all these reasons, Tonify has the upper hand over Lucene.

Table 2: Measures for French Elections Query

	French Elections 2012	
	Lucene	Tonify
Reciprocal Rank	0.50	1.00
Precision at Rank 10	0.50	0.40
DCG(10)	13.30	13.60
AVGP(20)	0.56	0.56
Recall	0.46	0.77

The "iPad 3 Review" query with Negative Sentiment as target tone yields scores in which Tonify stands out as a clear winner against Lucene. There are reasons for this. The iPad 3 is a commercial success, which means that there are a lot of advertisements and giveaways related to the product, leading to irrelevant results scoring higher than usual with respect to the query. Fortunately for Tonify, a negative tone is not prevalent in web sites promoting a product, so a lot of noise from advertisements and giveaways is ruled out from their tone. While this might make the **iPad 3** query look like it was designed to give Tonify an unfair advantage, it is a very common use case for people to be looking for the downsides of a critically-acclaimed product (e.g. iPad). With products like the iPad 3, good reviews are everywhere to be found, so manually filtering out the unwanted results is much more of a hassle than with a normal query. Therefore, Tonify proves to be an effective tool for this sort of use case.

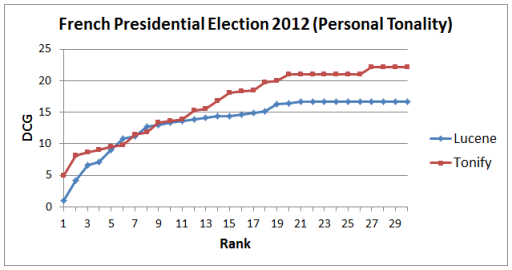
Table 3: Measures for iPad 3 Query

	iPad 3 Review	
	Lucene	Tonify
Reciprocal Rank	0.08	0.20
Precision at Rank 10	0.00	0.20
DCG(10)	3.46	5.13
AVGP(20)	0.13	0.38
Recall	0.43	0.57

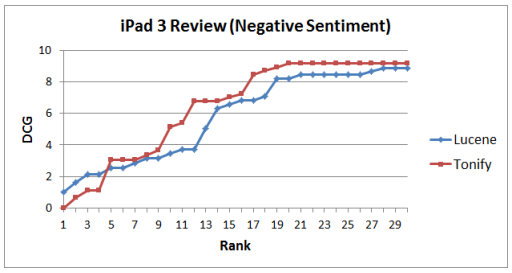
The last query tested ("United States Economy" with Upset Mood as target) gives almost inconclusive results. While Tonify beats Lucene in 3 of the 5 measures, it does not seem to justify the extra storage overhead and processing time in this case. Nonetheless, it must be noted that Tonify presented a better Recall and Reciprocal Rank than Lucene for the third time in a row, which leads us to believe that those two measures are its main strengths.

Table 4: Measures for USA Economy Query

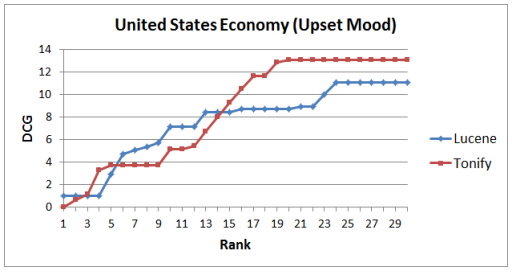
	USA Economy	
	Lucene	Tonify
Reciprocal Rank	0.20	0.25
Precision at Rank 10	0.30	0.20
DCG(10)	7.11	5.12
AVGP(20)	0.29	0.39
Recall	0.60	0.80



(a) "French Presidential Elections 2012" as query and "Personal" tonality as target



(b) "iPad 3 Review" as query and "Negative" sentiment as target



(c) "United States Economy" as query and "Upset" mood as target

Figure 4: Discounted Cumulative Gain Curves, Lucene vs Tonify

Another interesting set of results come from plotting DCG(i) versus i for both search engines (see Figure 4), giving us a

sense of progress along the rank list. In all three cases, both applications start off with similar trends. Nonetheless, Tonify always outperforms Lucene by the time the 20th result is reached. Its higher Recall also means that it maintains the lead from that point onwards.

6. RELATED WORK

Text classifiers have been widely used for Spam Filtering in projects like TrollGuard[18] and Layer-3 Firewalls in general. Meanwhile, projects that attempt to use text classifying technology on web pages have taken an on-demand approach, where users enter a specific URL to be analyzed for tone characteristics (see UrlAI[19], Typealyzer[20] and GenderAnalyzer[21]). Those services aim at satisfying user curiosity, but do not really address a specific problem in the way we interact with the Internet.

To the best of our knowledge, content tone has not been explored as a way to enhance text-based web search results. The most similar project that we found is Jinni[22], which is a mood-based search engine for video content (i.e. movies, TV shows, etc.) that appears to rely on user-aided metadata to label their objects. As such, it is safe to say that Jinni is not capable of addressing the user cases that we are concerned with, as it operates on a very specific non-text dataset without any substantial heterogeneity in it.

7. CONCLUSIONS AND FUTURE WORK

A tone-conscious search engine is an innovative idea that can improve web search results in a substantial way. Most importantly, it shapes the way people interact with a search engine by minimizing the need to *filter out* unwanted documents once the results are in.

Evaluation results show that the Tonify prototype is considerably better than unmodified Lucene in certain situations, demonstrating that the enhanced scoring function that we propose is a step in the right direction. Still, there is much to be done when it comes to adapting classifiers to analyze web pages. For example, our experiments made us realize that advertisements and user comments can have a huge effect on a news article's tone scores. Also, we have not yet integrated web-specific scoring techniques (e.g. PageRank) to the prototype, which would be necessary if the Tonify service were to stand a chance in an evaluation against search engines like Bing and Google.

8. REFERENCES

[1] P. R. Christopher D. Manning and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[2] A. Lucene. Similarity (lucene 3.0.2 api). `http://lucene.apache.org/core/old_versioned_docs/versions/3_0_2/api/all/org/apache/lucene/search/Similarity.html`.

[18] http://www.trollguard.com/
[19] http://www.urlai.com/
[20] http://www.typealyzer.com/
[21] http://www.genderanalyzer.com/
[22] http://www.jinni.com/