FreeSpeech: An anonymous short message platform

Eamonn Lawlor
B.A. (Mod.) Computer Science, Linguistics and German
Final Year Project, April 2014
Supervisor: Dr. Carl Vogel

School of Computer Science and Statistics

O'Reilly Institute, Trinity College, Dublin 2, Ireland
Declaration

I hereby declare that this project is entirely my own work and that it has not been submitted as an exercise for a degree at this or any other university.

______________________________  ________________________
Eamonn Lawlor                      Date
Permission to lend

I agree that the Library and other agents of the College may lend or copy this report upon request.

__________________________________________  __________________
Eamonn Lawlor                          Date
Abstract

The rise of social media as a primary means of communication and public discourse on the internet has led to concern over the amount of information that can be traced to an individual’s identity. However, anonymity in an online context is often associated with reckless, abusive and sometimes illegal behaviour. This paper proposes a system for online discourse that allows for anonymous communication while addressing some of the common concerns raised in relation to online anonymity. The system will use linguistic tools to meaningfully link related entries.
Acknowledgements

First and foremost, I would like to thank Dr. Carl Vogel, my project supervisor and mentor throughout my college years. His dedication to his field and to bringing out the best in his students is unrivalled.

I would like to thank the rest of the staff in the School of Computer Science and Statistics, the Centre for Language and Communication Studies and the Department of Germanic Studies for their support and tuition over the years.

Thank you to all my classmates and friends and above all my family for their unending support and encouragement.
# Contents

1 **Introduction** 1

2 **Background** 3
   2.1 The need for anonymity 3
   2.2 Is anonymity a good thing? 4
   2.3 Existing services 5
      2.3.1 4chan 5
      2.3.2 Kwikdesk 6

3 **Design** 8
   3.1 Core Concepts 8
   3.2 Technologies 9
      3.2.1 Server-side 9
      3.2.2 Client-side 11

4 **Implementation** 12
   4.1 Interface 12
   4.2 Server-side programs 16
      4.2.1 PHP 16
      4.2.2 Linguistic Tools 19
      4.2.3 MySQL 20
## Contents

### 5 Conclusion

5.1 Evaluation ................................................. 22
      5.1.1 Anonymity ........................................ 22
      5.1.2 Meaningfulness .................................. 23
      5.1.3 Grouping .......................................... 23

5.2 Further Work ............................................. 23

5.3 Final Remarks ............................................ 24

### A Source Code

A.1 functions.php ........................................... 29
A.2 login.php .................................................. 32
A.3 header.php ............................................... 32
A.4 index.php .................................................. 36
A.5 search.php ................................................ 38
A.6 speech.php ............................................... 40
A.7 footer.php ............................................... 41
A.8 default.css .............................................. 41
A.9 freespeechwn.java ...................................... 44

### B Source Code also on DVD


Chapter 1

Introduction

The concept of online anonymity predates the World Wide Web itself. In the late 1980s, users of newsgroups that discussed particularly sensitive topics began using ‘remailer’ services which routed their messages through servers that would strip away sender information, allowing for anonymous postings in newsgroups. (Rigby, 1995) These remailers would either send the message without any sender information, or with a pseudonym that could be used to identify the sender for the purposes of responding to the message. With the advent of the World Wide Web, anonymous and pseudonymous communication became more common through the use of message boards and e-mail services not provided by ISPs (and thus not attached to real identities) such as Hotmail.

It is only in recent years, with the meteoric rise of social media services such as Facebook and Twitter that online identities have come to be closely attached to real identities. Indeed, popular social networking services Facebook and Google+ both require users to provide, and be identified by, their real names. 12 On the other hand, anonymous and pseudonymous communication is still widespread on the internet and there are growing concerns about the privacy implications of sharing so much personal data online. This paper thus proposes a web service, titled FreeSpeech, that allows for totally anonymous short messages to be exchanged and grouped by linguistic properties, rather than the identity of...

This paper begins by examining the current state of online social interaction and the values of anonymity in online communication. The design of the proposed FreeSpeech system and the technologies required for it are then outlined. Following that, the implementation of the system is described. The concluding section of the paper assesses the effectiveness of the system and indicates future work that can be done in the area.
Chapter 2

Background

2.1 The need for anonymity

The question of identity associated with online communication has been a prevalent one since the birth of the internet and recent shifts in how people use the internet have accentuated this debate further. Online posts are increasingly being attached to a “persistent identity”. Whereas previously, this would usually refer to a pseudonym that would allow statements “to be public and persistent, but not attached to one’s real identity” (Madrigal, 2011), the ubiquity of social media websites such as Facebook and Twitter means that this persistent identity is often intrinsically linked to one’s real life identity. This can create a problem for those who wish to make comments online without an impact on their personal reputation. Those who wish to make comments anonymously online are often viewed as ‘trolls’ – an online slang term that refers to “a person who makes a deliberately offensive or provocative online posting”.¹ A common phrase among bloggers and journalists is that people should avoid reading “the bottom half of the internet”.² The argument here is that

CHAPTER 2. BACKGROUND

comments on news articles that make up the so-called “bottom half” do not contain anything of value and should be avoided due to the harmful content they may contain. This sentiment is one that has met significant opposition from other bloggers and journalists (Manuel, 2012; Fogg, 2013) and indeed, a blog was set up to highlight the quality content that can come from anonymous comments – the “bottom half”.

There are obviously many legitimate reasons one would wish to remain anonymous online – from fear of political oppression to whistleblowing to simply wanting to express feelings or ask questions that are outside of the societal expectations one would face by doing so in real life. This idea of being able to engage in a discussion without the burden of your comments being traced back to you is at the core of websites that allow anonymous posts such as 4chan, which will be discussed in §2.3.1. Social networking sites that encourage, or even require, use of one’s real identity run in opposition to this philosophy – they see anonymity as a “bug to be fixed”. (Dibbell, 2010) Mark Zuckerberg, founder and CEO of Facebook, has said that “having two identities for yourself is an example of a lack of integrity.” (Zimmer, 2010) However, sites such as Facebook have long been plagued with privacy concerns, and Zuckerberg himself has somewhat reversed his views on anonymity, admitting that “if you’re always under the pressure of real identity, I think that is somewhat of a burden.” (Stone and Frier, 2014) People have become disillusioned with the permanency of posts on sites like Facebook, and this has given rise to ephemeral services such as Snapchat, which allows users to send pictures that expire after a short period of time. (Wadhwa, 2013) Thus, it seems as if the notion of not having posts permanently attached to one’s identity is one that still holds value for a lot of people.

2.2 Is anonymity a good thing?

The merit of an anonymous system is certainly cause for debate. Impressions of anonymous online posters can often be negative – resulting in the aforementioned ‘troll’ moniker and “Don’t read the bottom half” meme. Vogel (2014) describes some studies that show varying

---

results as to the effect of anonymity on group behaviour. In some situations, it exerts a positive influence, in others negative, leading to the conclusion that the value of anonymity is context dependent. It is reasonable to assume that a totally anonymous online platform would have a lot of problems with content that is abusive, illegal or simply not of any interest, as a result of people being granted the freedom to post whatever messages they like without the consequences of these messages being traced back to them. The challenge for a new system would be to overcome the difficulty created by these posts in order to preserve this freedom.

2.3 Existing services

2.3.1 4chan

The most famous example of an anonymous online discussion forum is the imageboard 4chan.4 4chan allows users to upload an image which creates a discussion thread. There is no registration on the site; users have the option of identifying themselves with a moniker, but roughly 90 percent choose to remain anonymous. The site is organised into forums covering various topics such as Japanese animation and video games, though almost half of the site’s content comes from the ‘random’ section, known as /b/.5 (Dibbell, 2010) The site was founded in 2003 and has grown to 22.5 million monthly users as of September 2013.6

There is very little moderation on the site and as a result posts are often chaotic, nonsensical or obscene and include a large amount of pornographic content. Posts are also ephemeral as content is not archived and is instead deleted to make way for new content. Bernstein et al. (2011) estimate that the median thread on /b/ disappears off the site within five minutes. They also conclude that the anonymity of the site helps create a strong communal identity which can be seen through the numerous in-jokes, memes and style of

language adopted throughout the site.

4chan has been credited with the creation of a lot of internet memes and trends, for example “LOLcats”, pictures of cats accompanied with comically misspelled captions, as well as being the origin of the loose association of activists who operate under the Anonymous moniker. The most significant undertaking of Anonymous activists was Project Chanology, in which they organised simultaneous masked protests worldwide against the practices of the Church of Scientology. (Knuttila, 2011) The /b/ community has also drawn criticism for alleged invasions of other sites, replacing content with offensive and pornographic material, bomb threats, and hacking American politician Sarah Palin’s email account. (Fox News, 2009)

The site’s founder, Christopher Poole, known by the alias ‘moot’, believes anonymity is intrinsic to the success of the site. He disagrees with the notion of a persistent identity on the internet, claiming that “the cost of failure is really high when you’re contributing as yourself” and “people deserve a place to be wrong” (Dibbell, 2010) These ideals are at the core of the 4chan community, and have been successful in fostering a creative community with the power to affect change as a group, despite the users remaining anonymous. The complete anonymity and lack of moderation have led to a high density of graphic and nonsensical content, however, which is a possible drawback to the site as a platform for meaningful anonymous discourse.

2.3.2 Kwikdesk

Kwikdesk\(^7\) is an ephemeral, anonymous short message platform. It allows users to post a short message, known as a ‘KWIK’, and designate a period, either 24 hours or 10 days, in which the message will be available before it ‘self-destructs’. Messages are indexed by hashtags – words or phrases prefixed by a hash symbol (#) – a form of metadata tag used on other popular social networking services such as Twitter and Facebook. Users are also given the option of creating their own personal hashtag – in this case, a long series of hexadecimal digits – that they can share with their friends in order to communicate

with each other. Viewing previous KWIKs is accessible only through searching existing hashtags.

The site was created in 2013 by Kevin Abosch, who has said that the web application is a “proof of concept” of the underlying technology that telecoms companies would be keen to utilise. (Weckler, 2014) The site has been billed as a “Snapchat Meets Twitter”, emphasising the ephemerality of the service. Abosch reported 50,000 messages sent over 48 hours in November 2013. (Crook, 2013) As the site is relatively new, data on how users interact with the service is scarce. It is currently difficult to identify conversations taking place on the site without prior knowledge of what hashtag to search for.
Chapter 3

Design

3.1 Core Concepts

In response to the concerns surrounding anonymity described in §2, this paper proposes a new system, known as FreeSpeech, that will allow users to communicate by sending short messages, similar to Twitter or KwikDesk, in an anonymous capacity, and for the messages to be sorted and displayed to the user in a meaningful way. The core concepts of the system are as follows:

- The system should be totally anonymous. There should be no information linking a post to the identity of the user posting it. The site will have no registration system and not require users to submit a name or a pseudonym, and IP addresses of users should not be logged.

- Posts should be meaningful. Efforts should be made to avoid spam and obscene content. Detection tools should be put in place to either disallow such content, or make it less accessible to the user.

- Posts should be grouped together in interesting ways. In order to make the site more interesting to use, linguistic tools should be used to link related posts in a manner that is more meaningful than a simple tag search.
FreeSpeech is conceived as a web application that users would access through a web browser.

3.2 Technologies

3.2.1 Server-side

The system was built on a LAMP software stack, which in this case refers to Linux, Apache, MySQL and PHP.

Linux

The system was built on a server running the Debian distribution of the Linux operating system. Linux is an extremely popular open source operating system and Debian is the most popular distribution for web servers.\(^1\) Debian is seen as a stable Linux distribution with a wide range of software packages available for it.

Apache

The system was implemented on an Apache web server. Apache is an open-source HTTP server application that serves 52.4% of active websites.\(^2\)

MySQL

MySQL is an open-source relational database management system that is popular for use in web applications due to its integration with the LAMP stack. It is used to maintain a database of entries on the site.

\(^1\)http://w3techs.com/technologies/details/os-linux/all/all – retrieved April 2014.

CHAPTER 3. DESIGN

PHP

PHP is a server-side scripting language that allows for dynamic content to be displayed on webpages. It is used to process the information inputted by the user and to retrieve information to be displayed to them.

TextCat Language Guesser

TextCat\(^3\) is a free software implementation of the text categorization algorithm presented in Cavnar and Trenkle (1994) that can detect if a given text is in one of 69 natural languages.

TreeTagger

TreeTagger\(^4\) is a tool developed at the University of Stuttgart for annotating text with part-of-speech information.

WordNet

WordNet\(^5\) is a lexical database for English developed at Princeton University that groups nouns, verbs, adjectives and adverbs into sets of synonyms, or synsets. A third-party API – Java API for WordNet Searching (JAWS)\(^6\) – was used to interact with the database using Java.

Java

Java is a popular object-oriented programming language. It was used in order to interact with the WordNet database.

\(^3\)http://odur.let.rug.nl/~vannoord/TextCat/ – retrieved April 2014.
\(^4\)http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/ – retrieved April 2014.
3.2.2 Client-side

The user-interface is presented as a series of HTML documents with PHP commands embedded within. CSS is used to describe the formatting and layout of the pages for the user.
Chapter 4

Implementation

This chapter describes how the FreeSpeech web application functions. The source code is provided in Appendix A.

4.1 Interface

![FreeSpeech homepage, top.](image)

Figure 4.1: FreeSpeech homepage, top.
The first page the user sees upon entering the FreeSpeech website will be similar to that seen in Figure 4.1. The page begins with a logo and a simple explanation of the site’s functionality, along with some navigation options. The user is then presented with a text field in which they can type a new entry. This is followed by navigation options according to language of entries, and page navigation within the language views. By default, entries from all languages are shown. Figure 4.2 shows the bottom half of the page, which contains the remainder of the entries along with a search bar and some information about the project along with the Trinity College Dublin crest on the bottom of the page.

![Figure 4.2: FreeSpeech homepage, bottom.](image-url)

The search page (Figure 4.3), by default, shows the text field to type a new entry, followed by the search bar, without any entry information in between.
From either of these pages, the user has the option of creating a new entry, at which point it will be processed and displayed along with the other entries, and the user will see the updated homepage. The user can also use the search box to search entries in any of the languages that are available. (Figure 4.4) This list of languages, along with the languages on the navigation bar, update automatically as entries in new languages are added to the database. The results will then be displayed on the search page. (Figure 4.5)
CHAPTER 4. IMPLEMENTATION

You searched: "bayerm" in German.

Figure 4.5: Search results.

Each individual entry contains information on when it was posted, what language it was detected as being in and, if an entry has been made before, a counter of how many other times it has been entered. (Figure 4.6) Clicking on an entry will bring up a page headed by the entry and followed by related entries. (Figure 4.7)

Figure 4.6: Entry information.

Figure 4.7: Related entries.
4.2 Server-side programs

4.2.1 PHP

In this section, an outline is given of the PHP classes implemented in the web application.

functions.php

This file contains functions called upon by the other PHP files.

- **sanitize** accepts a string as a parameter and performs operations on it that will remove or escape characters that could cause an error if the string is inputted as part of a MySQL query. It also protects against MySQL injection.

- **printSpeech** accepts the unique ID number that corresponds to a speech entry in the database as a parameter, and prints out the information on it to HTML, as seen in Figure 4.6.

- **getSimilar** accepts arrays of adjectives, adverbs, nouns and verbs, and a string representing a language, as parameters. It compares each array to the array of that part of speech stored in the database for every other speech entry in that language. In the process, it compiles an array of matching ID numbers, each attributed scores based on how many items match as follows:
  - Adjective – 2 points
  - Adverb – 1 point
  - Noun – 10 points
  - Verb – 5 points

The five ID numbers with the highest scores after all the passes are returned in an array.
• **wordnet** accepts two strings representing a word and a part of speech as parameters. The Java program that interacts with the WordNet database is called and an array of strings representing the synset of the inputted word is returned.

**login.php**

This file contains the login information used to access the MySQL database. For security reasons, it was kept in a separate directory to the other files, not accessible to the web.

**header.php**

This file begins by calling both [functions.php](#) and [login.php](#). A connection with the MySQL database is established. The header information (logo, navigation, introductory text) is displayed. The form is then displayed to accept speech entries. Entries are passed back to the index page (which includes this file at the start of it) using the GET method. This file continues by checking if a speech variable is set, and displaying appropriate errors if the user did not input anything (Figure 4.8) or tried to input a URL. (Figure 4.9)

![Figure 4.8: No input error message.](#)
If the variable is set, and these error criteria have not been met, the program proceeds to pass the speech information to TextCat. A language variable is then set, and this is checked against the core languages (English, French, German, Italian and Spanish). If the language matches one of these, the speech information is sent to TreeTagger and arrays of the parts of speech (adjectives, adverbs, nouns and verbs) are acquired. If the language is English, these are arrays are further sent to the WordNet program (via the wordnet function in functions.php) and additional information is added. These arrays are then compared to the matching arrays of previous entries (using the getSimilar function in functions.php) and finally, all the information that has been acquired is sanitized (using the sanitize function in functions.php) and sent to the database.

index.php

This file begins by calling the header.php file. It then uses a series of PHP expressions to display the language and page navigation options. It then calls on the printSpeech function in functions.php to print out all the entries depending on what language and page has been selected. It then displays the search box and finally, footer.php is called.

search.php

This file begins by calling the header.php file. It then makes database queries relevant to the search and language queries passed to it by GET methods and displays the relevant
entries. If no search query is detected, no entries are displayed. It then displays the search box and finally, footer.php is called.

speech.php

This file begins by calling the header.php file. It then makes database queries relevant to the entry ID passed to it by a GET method. It displays the entry corresponding to the ID, and the entries corresponding to the five top matches for that entry. Finally, footer.php is called.

footer.php

This file displays the footer text and the Trinity College Dublin logo, before closing access to the MySQL database.

4.2.2 Linguistic Tools

TextCat

TextCat is called upon to detect the language of the user-inputted text from 69 languages. By default, TextCat returns multiple guesses at the language contained in an input text, but for the purpose of this application, only the closest match is returned.

TreeTagger

TreeTagger accepts a text file and a parameter file as input. For the purposes of this application, it was decided to add TreeTagger support for six core languages – English, French, German, Italian and Spanish. The inputted text was parsed to a temporary text file, and the relevant language parameter file was called.
WordNet

If an entry was detected as being in English, it’s categorised parts of speech were passed to a custom Java program based on a template by the author of JAWS. (see §3.2.1) This would return a string of synset matches for that word based on queries to the WordNet database, thus enriching the matching capabilities of English entries.

4.2.3 MySQL

A MySQL database was created with three tables as follows:

<table>
<thead>
<tr>
<th>Entries</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique entry ID</td>
</tr>
<tr>
<td>speechref</td>
<td>Text of entry</td>
</tr>
<tr>
<td>time</td>
<td>Timestamp of entry</td>
</tr>
<tr>
<td>lang</td>
<td>Detected language</td>
</tr>
<tr>
<td>adjs</td>
<td>Array of adjectives</td>
</tr>
<tr>
<td>advs</td>
<td>Array of adverbs</td>
</tr>
<tr>
<td>nouns</td>
<td>Array of nouns</td>
</tr>
<tr>
<td>verbs</td>
<td>Array of verbs</td>
</tr>
<tr>
<td>matches</td>
<td>Array of matching ID nos.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Speeches</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Unique speech ID</td>
</tr>
<tr>
<td>speech</td>
<td>Text of speech</td>
</tr>
<tr>
<td>count</td>
<td>No. of occurrences</td>
</tr>
</tbody>
</table>
Two triggers were also created so that:

- Whenever a row was added to the *entries* table, the *speechref* would be checked against the *speech* column in the *speeches* table, and a row added to the *speeches* table if there was no match, otherwise the corresponding match would increment its *count* column.

- Whenever a row was added to the *entries* table, the *lang* would be checked against the *lang* column in the *langs* table, and a row added to the *langs* table if there was no match, otherwise the corresponding match would increment its *count* column.
Chapter 5

Conclusion

5.1 Evaluation

The system outlined in this report gives a good indication of how an anonymous short messaging platform can be implemented. The system can be evaluated in terms of the three core concepts outlined in §3.1.

5.1.1 Anonymity

The system has succeeded in providing an anonymous platform. No user registration is required, and so no user can distinguish between who has inputted the other entries on the site. However, the Apache server software automatically logs IP addresses of anyone who interacts with the server, for example to submit an entry or perform a search. It is possible to program Apache to not log IP addresses, which would be necessary to hide user information from the site administrator as well as other users. It remains to be seen whether the proposed system would implement such a feature, as other anonymous sites such as 4chan still log IP addresses in order to ban those posting illegal content, and this may be a requirement in the future.
5.1.2 Meaningfulness

The system has some very basic spam filtering in place, in that it uses regular expressions to block users from posting URLs. This is not very effective, as a human user could easily circumvent this, for example by adding spaces between characters. The decision was made not to implement an obscene language filter, as it is far too difficult to do so using regular expressions.\(^1\) For both spam detection and obscene language filtering, a more sophisticated filter would need to be installed.

5.1.3 Grouping

The system succeeded in grouping entries in a number of ways, the most obvious being by language. However, the TextCat language detections were not always correct, especially for particularly short entries. (see Figure 4.1) The grouping according to number of part-of-speech matches, although primitive, provided some success. An entry could be linked to its five closest matches in any of the six chosen core languages.

5.2 Further Work

In each of the concepts mentioned above, the system can be improved. A more rigorous anonymity system can be put in place to ensure total anonymity of users, perhaps by re-configuring how the Apache server deals with IP logs. Care will have to be taken here, as some jurisdictions have minimum data retention periods in place for website owners.

More sophisticated spam and language filtering can be implemented using machine learning techniques, and eventually, it is hoped that a system could be set up to automatically detect how useful of a contribution an entry is based on its content, and give this precedence in how the user observes the site.

A lot can be done to improve how entries are grouped. User input can be used as a means of correcting false language detections, for instance. The system of assigning

\(^1\text{See, for example the Scunthorpe Problem. (McCullagh, 2004)}\)
arbitrary scores for different parts of speech is not particularly sophisticated, and machine learning can be introduced here also to create a more comprehensive matching algorithm. It is hoped that users will be eventually be able to interact with the site in a number of different ‘dimensions’, such as sorting entries by time submitted, meaningfulness, number of linked entries and so forth.

5.3 Final Remarks

In conclusion, the FreeSpeech system provides a rudimentary platform for anonymous speech. How real users would interact with such a platform remains to be seen, but the possibilities are limitless in the amount of classifications and linguistic filters that can be implemented to provide a more meaningful user experience.
Bibliography


Appendix A

Source Code

This appendix contains the source code for the FreeSpeech web application, along with relevant instructions on how to install it on a new server.

File structure is as follows:

```
/  
  |  
  +-- resources  
      |  
      +-- functions.php  
      |  
      +-- login.php  
      |  
      +-- var  
          |  
          +-- www  
              |  
              +-- header.php  
              |  
              +-- index.php  
              |  
              +-- search.php  
              |  
              +-- speech.php  
              |  
              +-- footer.php  
              |  
              +-- wordnet  
                  |  
                  +-- java  
                      |  
                      +-- freespeechwn.java  
```
A.1 functions.php

```php
<?php

/*
 * Remove characters from a string that might interfere with MySQL input.
 */

function sanitize($string) {
    if (get_magic_quotes_gpc()) $string = stripslashes($string);
    return mysql_real_escape_string($string);
}

/*
 * Prints a visual representation of a given speech entry.
 */

function printSpeech($id) {
    $query = "SELECT speechref, time, count, lang FROM entries LEFT JOIN speeches ON entries.speechref = speeches.speech WHERE entries.id LIKE "$id";"
    $result = mysql_query($query);
    if (!$result) die ("Database access failed: " . mysql_error());
    $a = mysql_fetch_row($result);
    $txt = $a[0];
    $time = date("D j M y, H:i", strtotime($a[1]));
    $cnt = $a[2] - 1;
    $lang = ucwords($a[3]);
    if ($cnt == 1) $cntTitle = "This has been said one other time.";
    else $cntTitle = "This has been said "$cnt" other times.";
    $langTitle = "This speech has been detected as being in "$lang".";
    echo "<div class='speechobj'>
    <a href="speech.php?id=$id">$txt</a><div class='speechtxtobj'>$txt</div>
    <span class='speechtimeobj'>$time</span>
    <span class='speechlangobj' title='$langTitle'>$lang</span>
</div>

    if ($cnt > 0) echo "<span class='speechcntobj' title="$cntTitle">$cnt</span>;";
    ";
}

/*
```
Compares given arrays of words to other entries in database and returns
two-dimensional array of top 5 similar entries.
*/

function getSimilar($adjs, $advs, $nouns, $verbs, $lang) {
    $matches = array();
    foreach ($adjs as $adj) {
        $adj = sanitize($adj);
        $query = "SELECT id FROM entries WHERE lang LIKE "'".$lang."' AND adjs LIKE '".%$adj"%';
        $result = mysql_query($query);
        if (!$result) die ("Database access failed: ". mysql_error());
        for ($i = 0; $i < mysql_num_rows($result); ++$i) {
            $item = mysql_fetch_row($result)[0];
            $found = false;
            foreach ($matches as &$match) {
                if ($item == $match[0]) {
                    $match[1] += 2;
                    $found = true;
                    break;
                }
            }
            if (!$found) $matches[] = array($item, 2);
        }
    }
    foreach ($advs as $adv) {
        $adv = sanitize($adv);
        $query = "SELECT id FROM entries WHERE lang LIKE "'".$lang."' AND advs LIKE '".%$adv"%';
        $result = mysql_query($query);
        if (!$result) die ("Database access failed: ". mysql_error());
        for ($i = 0; $i < mysql_num_rows($result); ++$i) {
            $item = mysql_fetch_row($result)[0];
            $found = false;
            foreach ($matches as &$match) {
                if ($item == $match[0]) {
                    $match[1] += 1;
                    $found = true;
                    break;
                }
            }
            if (!$found) $matches[] = array($item, 1);
        }
    }
    foreach ($nouns as $noun) {
        $noun = sanitize($noun);
        $query = "SELECT id FROM entries WHERE lang LIKE "'".$lang."' AND nouns LIKE '".%$noun"%';
        $result = mysql_query($query);
        if (!$result) die ("Database access failed: ". mysql_error());
        for ($i = 0; $i < mysql_num_rows($result); ++$i) {
            $item = mysql_fetch_row($result)[0];
            $found = false;
            foreach ($matches as &$match) {
                if ($item == $match[0]) {
                    $match[1] += 1;
                    $found = true;
                    break;
                }
            }
            if (!$found) $matches[] = array($item, 1);
        }
    }
}
$result = mysql_query($query);
if (!$result) die("Database access failed: " . mysql_error() . " Query was: " . $query);

for ($i = 0; $i < mysql_num_rows($result); ++$i) {
    $item = mysql_fetch_row($result)[0];
    $found = false;
    foreach ($matches as $match) {
        if ($item == $match[0]) {
            $match[1] += 10;
            $found = true;
            break;
        }
    }
    if (!$found) $matches[] = array($item, 10);
}

foreach ($verbs as $verb) {
    $verb = sanitize($verb);
    $query = "SELECT id FROM entries WHERE lang LIKE '\$lang' AND verbs LIKE '\$verb\%';";
    $result = mysql_query($query);
    if (!$result) die("Database access failed: " . mysql_error());

    for ($i = 0; $i < mysql_num_rows($result); ++$i) {
        $item = mysql_fetch_row($result)[0];
        $found = false;
        foreach ($matches as $match) {
            if ($item == $match[0]) {
                $match[1] += 5;
                $found = true;
                break;
            }
        }
        if (!$found) $matches[] = array($item, 5);
    }
}

function cmp($a, $b) {
    if ($a[1] == $b[1]) return 0;
    return ($a[1] < $b[1]) ? -1 : 1;
}

usort ($matches, "cmp");
$matches = array_reverse($matches);
foreach ($matches as $match) error_log(implode(",", $match));

$sim = array_slice($matches, 0, 5);
$r = array();
foreach ($sim as $s) $r[] = $s[0];
return $r;
APPENDIX A. SOURCE CODE

A.2 login.php

Host name and MySQL account details will have to be substituted in here.

```php
<?php
$db_hostname = 'XXXX';
$db_database = 'speeches';
$db_username = 'XXXX';
$db_password = 'XXXX';
?>
```

A.3 header.php

```php
<?php
require_once '/resources/login.php';
require_once '/resources/functions.php';

$db_server = mysql_connect($db_hostname, $db_username, $db_password);
if (!$db_server) die("Unable to connect to MySQL: " . mysql_error());
unset($db_username, $db_password);
mysql_select_db($db_database)
or die("Unable to select database: " . mysql_error());

echo <<<_END
<div>
<div id="introcont"
```
FreeSpeech is an anonymous short messaging platform. Try it out by saying something in the box below, or click on a speech to see other related entries.

```php
if (isset($_POST['speech'])) {
    $speech = sanitize($_POST['speech']);
    $speech = trim($speech);
    if ($speech == '') {
        header("Location: index.php?e=1"); // Error message triggered if user attempted to enter empty speech
    } else if (preg_match('/[^a-zA-Z\s]+:\s+[^a-zA-Z\s]+\/[0-9a-zA-Z\s;.:@\#&%~,+]*/', $speech)) {
        header("Location: index.php?e=2"); // Error message triggered if user attempted to enter URL
    } else {
        /*
         * TextCat is used on the inputted string to detect language.
         */

        $textcat = "\$textcat = "/usr/local/bin/text_cat -u l -1 \n" . $speech . "\n";
        ob_start();
passthru($textcat);
        $lang = ob_get_contents();
        ob_end_clean();
        $lang = sanitize($lang);
        $lang = trim($lang);
        $lang = str_replace("\n", ", $lang);
        $lang = preg_replace("/^[a-z0-9_.]*/", ", $lang);
        $lang = str_replace(", ", $lang);
        if ($lang == "I don’t know; Perhaps this is a language I haven’t seen before:")
            $lang = "unknown";
        /*
         * TreeTagger is used to detect POSs of entry.
         */
```
APPENDIX A. SOURCE CODE

```php
if(preg_match("/english|french|german|italian|spanish/", $lang)) {
    $tokenizedSpeech = str_replace(" ", "\n", $speech);
    $tokenFile = "/fstmp/tokenfile.txt";
    $tfStream = fopen($tokenFile, "w");
    fwrite($tfStream, $tokenizedSpeech);
    fclose($tfStream);
    $treetag = "/tt/bin/tree-tagger/\tt/lib/\$lang.par/fstmp/tokenfile.txt";
    ob_start();
    passthru($treetag);
    $cats = ob_get_contents();
    ob_end_clean();
    $cats = rtrim($cats);
    $catsArr = explode("\n", $cats);
    $tokenizedSpeech = preg_replace("#\[[[:punct:]]\]#", "", $tokenizedSpeech);
    $wordsArr = explode("\n", $tokenizedSpeech);
}

$adjs = array();
$advs = array();
$nouns = array();
$verbs = array();

if($lang == "english") {
    for($i = 0; $i < count($wordsArr) && $i < count($catsArr); ++$i) {
        if(preg_match("/^JJ/", $catsArr[$i]))
            $adjs[] = $wordsArr[$i];
        else if(preg_match("/^RB/", $catsArr[$i]))
            $advs[] = $wordsArr[$i];
        else if(preg_match("/^N/", $catsArr[$i]))
            $nouns[] = $wordsArr[$i];
        else if(preg_match("/^V/", $catsArr[$i]))
            $verbs[] = $wordsArr[$i];
    }
}
else if($lang == "french") {
    for($i = 0; $i < count($wordsArr) && $i < count($catsArr); ++$i) {
        if(preg_match("/^ADJ/", $catsArr[$i]))
            $adjs[] = $wordsArr[$i];
        else if(preg_match("/^ADV/", $catsArr[$i]))
            $advs[] = $wordsArr[$i];
        else if(preg_match("/^NOM/", $catsArr[$i]))
            $nouns[] = $wordsArr[$i];
        else if(preg_match("/^VER/", $catsArr[$i]))
            $verbs[] = $wordsArr[$i];
    }
}
else if($lang == "german") {
    for($i = 0; $i < count($wordsArr) && $i < count($catsArr); ++$i) {
        if(preg_match("/^ADJ/", $catsArr[$i]))
            $adjs[] = $wordsArr[$i];
        else if(preg_match("/^ADV/", $catsArr[$i]))
            $advs[] = $wordsArr[$i];
    }
}
```
$advs[] = $wordsArr[$i];
else if(preg_match('/^N/', $catsArr[$i]))
    $nouns[] = $wordsArr[$i];
else if(preg_match('/^V/', $catsArr[$i]))
    $verbs[] = $wordsArr[$i];
}

else if($lang == "italian") {
    for($i = 0; $i < count($wordsArr) && $i < count($catsArr); ++$i) {
        if(preg_match('/^ADJ/', $catsArr[$i]))
            $adjs[] = $wordsArr[$i];
        else if(preg_match('/^ADV/', $catsArr[$i]))
            $advs[] = $wordsArr[$i];
        else if(preg_match('/^N[^E]/', $catsArr[$i]))
            $nouns[] = $wordsArr[$i];
        else if(preg_match('/^VER/', $catsArr[$i]))
            $verbs[] = $wordsArr[$i];
    }
}

else if($lang == "spanish") {
    for($i = 0; $i < count($wordsArr) && $i < count($catsArr); ++$i) {
        if(preg_match('/^ADJ/', $catsArr[$i]))
            $adjs[] = $wordsArr[$i];
        else if(preg_match('/^ADV/', $catsArr[$i]))
            $advs[] = $wordsArr[$i];
        else if(preg_match('/^[N][E]/', $catsArr[$i]))
            $nouns[] = $wordsArr[$i];
        else if(preg_match('/^VER/', $catsArr[$i]))
            $verbs[] = $wordsArr[$i];
    }
}

if ($lang == "english") {

    $wnadjs = array();
    $wnadvs = array();
    $wnnouns = array();
    $wnverbs = array();

    foreach ($adjs as $a) $wnadjs[] = wordnet($a,"adj");
    foreach ($advs as $a) $wnadvs[] = wordnet($a,"adv");
    foreach ($nouns as $a) $wnnouns[] = wordnet($a,"noun");
    foreach ($verbs as $a) $wnverbs[] = wordnet($a,"verb");

    foreach ($wnadjs as $a) $adjs = array_merge($adjs,$a);
    foreach ($wnadvs as $a) $advs = array_merge($advs,$a);
    foreach ($wnnouns as $a) $nouns = array_merge($nouns,$a);
    foreach ($wnverbs as $a) $verbs = array_merge($verbs,$a);

    /*
       WordNet is run on English entries to add synonyms to tags.
    */

    if ($lang == "english") {

        $wnadjs = array();
        $wnadvs = array();
        $wnnouns = array();
        $wnverbs = array();

        foreach ($adjs as $a) $wnadjs[] = wordnet($a,"adj");
        foreach ($advs as $a) $wnadvs[] = wordnet($a,"adv");
        foreach ($nouns as $a) $wnnouns[] = wordnet($a,"noun");
        foreach ($verbs as $a) $wnverbs[] = wordnet($a,"verb");

        foreach ($wnadjs as $a) $adjs = array_merge($adjs,$a);
        foreach ($wnadvs as $a) $advs = array_merge($advs,$a);
        foreach ($wnnouns as $a) $nouns = array_merge($nouns,$a);
        foreach ($wnverbs as $a) $verbs = array_merge($verbs,$a);
    }
A.4 index.php

```php
foreach ($wnverbs as $a) $verbs = array_merge($verbs, $a);

foreach ($adjs as $a)
    $sqladjs = sanitize(serialize($a));
foreach ($advs as $a)
    $sqladvs = sanitize(serialize($a));
foreach ($nouns as $a)
    $sqlnouns = sanitize(serialize($a));
foreach ($verbs as $a)
    $sqlverbs = sanitize(serialize($a));
foreach ($sim as $a)
    $sqlsim = sanitize(serialize($a));

$query = "INSERT INTO entries VALUES (NULL, "$speech", NULL, "$lang", "$sqladjs", "$sqladvs", "$sqlnouns", "$sqlverbs", "$sqlsim")";
$result = mysql_query($query);
if (!$result)
    die("Database access failed: " . mysql_error() . " and query was " . $query);
header("Location: index.php");

if (isset($_GET['e'])) {
    if ($_GET['e'] == 1) echo "<div class="error">You didn't say anything!</div><br>";
    else if ($_GET['e'] == 2) echo "<div class="error">URLs are forbidden!</div><br>";
}
else echo "</div></div>";
?>
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>FreeSpeech</title>
    <style>@import url('default.css');</style>
    <link rel="icon" href="favicon.ico" type="image/x-icon">
    <link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
</head>
<body>
    <?php
    if (isset($_GET['language'])) echo "All";
    else echo "<a href="index.php">All</a>";
    if (isset($_GET['language'])) {
        $curlang = sanitize($_GET['language']);
```
APPENDIX A. SOURCE CODE

```php
$query = "SELECT entries.id FROM entries LEFT JOIN speeches ON entries.speechref=
speeches.speech WHERE lang LIKE '
$curlang' ORDER BY entries.id DESC";
else {
$query = "SELECT entries.id FROM entries LEFT JOIN speeches ON entries.speechref=
speeches.speech ORDER BY entries.id DESC";
$curlang = "";
}
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
$rows = mysql_num_rows($result);
$idarr = array();
for ($i = 0; $i < $rows; ++$i) {
$row = mysql_fetch_row($result);
$idarr[] = $row[0];
}
$pages = ceil($rows / 10);
$query = "SELECT lang FROM langs ORDER BY lang";
$result = mysql_query($query);
if (!$result) die ("Database access failed: " . mysql_error());
for ($i = 0; $i < mysql_num_rows($result); ++$i) {
$langchoice = mysql_fetch_row($result)[0];
if ($curlang == $langchoice) echo "<a href="index.php?language=$langchoice"">" .
ucwords($langchoice) . "</a>";
else echo ucwords($langchoice);
}
if (!isset($_GET['page'])) $curpage = 1;
else $curpage = $_GET['page'];
if ($pages > 1) {
    echo '<br />
';
    if ($curpage != 1 && !isset($_GET['language'])) echo "<a href="index.php"">1</a>";
    else if ($curpage != 1) echo "<a href="index.php?language=$curlang"">1</a>";
    else echo '1';
    for ($i = 2; $i <= $pages; ++$i) {
        echo ' | ';
        if ($curpage != $i && !isset($_GET['language'])) echo "<a href="index.php?page=$i
"">$i</a>";
        else if ($curpage != $i) echo "<a href="index.php?language=$curlang&page=$i
"">$i</a>";
        else echo $i;
```
\[\textbf{APPENDIX A. SOURCE CODE}\]

```php
for ($j = $curpage * 10 - 10; ($j < $curpage * 10) && ($j < $rows); ++$j) {
    printSpeech($idarr[$j]);
}
```

```html
<!DOCTYPE html>
<head>
<title>Search Results | FreeSpeech</title>
<style>@import url('default.css');</style>
</head>
<body>
<?php require_once 'header.php'; ?>
</body>
</html>```
APPENDIX A. SOURCE CODE

```php
$END;

$END;

$END;

$END;

$END;

$END;

$END;

if (!strcasecmp($langchoice, "all")) {
  $query = "SELECT lang FROM langs ORDER BY lang";
  $result = mysql_query($query);
  if (!$result) die ("Database access failed: " . mysql_error());
  for ($l = 0; $l < mysql_num_rows($result); ++$l) {
    $langchoice = mysql_fetch_row($result)[0];
    echo "<option value="" . $langchoice . "">" . ucwords($langchoice) . "</option>";
  }
}

$END;
</select>

<input type="submit" value="Search" />

</form>
<br />

if (!isset($_GET['q'])) {
  $q = sanitize($_GET['q']);
  if (isset($_GET['language']) && ($_GET['language'] != "")) {
    $lang = sanitize($_GET['language']);
    echo "You searched: \"" . $q . "\" in " . ucwords($lang) . ",";
    $query = "SELECT entries.id FROM entries LEFT JOIN speeches ON entries.speechref= speeches.speech WHERE speechref LIKE '%"$q"' AND lang LIKE '$lang'";
  } else {
    echo "You searched: \"" . $q . "\"";
    $query = "SELECT entries.id FROM entries LEFT JOIN speeches ON entries.speechref= speeches.speech WHERE speechref LIKE '%"$q"'";
  }
  $result = mysql_query($query);
  if (!$result) die ("Database access failed: " . mysql_error());
  $rows = mysql_num_rows($result);
  $idarr = array();
  if ($rows < 1) {
    echo "<p class="error">No results found!</p>";
  } else {
    for($j = 0; $j < $rows; ++$j) {
      $row = mysql_fetch_row($result);
      $idarr[] = $row[0];
    }
  }
```

APPENDIX A. SOURCE CODE

```php
$pages = ceil($rows / 10);

if(!isset($_GET['page'])) $curpage = 1;
else $curpage = $_GET['page'];

if ($pages > 1) {
    echo '<br />';
    if($curpage != 1 && isset($_GET['language'])) echo "<a href="search.php?q=$q
        language=$lang">1</a>";
    else if($curpage != 1) echo "<a href="search.php?q=$q">1</a>";
    else echo '1';
    for ($i = 2; $i <= $pages; ++$i) {
        echo ' | ';
        if($curpage != $i && isset($_GET['language'])) echo "<a href="search.php?q=$q
            language=$lang">$i</a>";
        else if ($curpage != $i) echo "<a href="search.php?q=$q&page=$i">$i</a>";
        else echo $i;
    }
    echo '</br>';
    for ($j = $curpage * 10 - 10; ($j < $curpage * 10) && ($j < $rows); ++$j) {
        printSpeech($idarr[$j]);
    }
}
require_once 'footer.php';
?>
</body>
</html>

A.6 speech.php

```
APPENDIX A. SOURCE CODE

if(isset($_GET['id'])) {
    $id = sanitize($_GET['id']);
    printSpeech($id);
    $query = "SELECT matches FROM entries WHERE entries.id LIKE '$id';"
    $result = mysql_query($query);
    if(!$result) die ("Database access failed: " . mysql_error());
    $row = mysql_fetch_row($result);
    $matches = unserialize($row[0]);
    foreach($matches as $match) printSpeech($match);
}
require_once 'footer.php';
?>
</body>
</html>

A.7 footer.php

<?php
    echo <<<_END
    <div id="footertext">
        <span>FreeSpeech is a project by Eamonn Lawlor under supervision from Dr. Carl Vogel in the School of Computer Science and Statistics, Trinity College Dublin</span>
    </div>
_END;

mysql_close($db_server);
?>

A.8 default.css

This file contains the CSS stylesheet for the website.

* { 
    font-family: Helvetica, Verdana, Arial, sans-serif;
}
textarea {
font-family: inherit;
font-size: inherit;
resize: none;
}

table, td {
  padding: 2px;
}

.error {
  color: red;
}

.speechobj {
  border: 1px solid gray;
  border-radius: 5px;
  margin-bottom: 5px;
  padding: 5px;
  position: relative;
  width: 600px;
  text-align: center;
  font-size: 10pt;
}

.speechtxtobj {
  text-align: left;
  padding-bottom: 5px;
  font-size: 12pt;
}

.speechtimeobj {
  float: left;
  color: gray;
}

.speechlangobj {
  display: inline-block;
  color: gray;
}

.speechcntobj {
  float: right;
  color: gray;
}

.speechtagobj {
  text-align: left;
  color: gray;
}

a.speechlink:link {
APPENDIX A. SOURCE CODE

    color: black;
    text-decoration: none;
}

a.speechlink:visited {
    color: black;
    text-decoration: none;
}

a.speechlink:hover {
    color: blue;
    text-decoration: underline;
}

a.speechlink:active {
    text-decoration: none;
}

#fslogo {
    height: 200px;
    width: 320px;
    float: left;
}

#topbar {
    font-size: large;
    font-weight: bold;
}

#footertext {
    font-size: small;
}

#tcdlogo {
    height: 77px;
    width: 62px;
    align: text-bottom;
    position: relative;
    left: 350px;
}

#intro {
    float: left;
    width: 300px;
}

#introcont {
    overflow: hidden;
}

#formcont {

import edu.smu.tspell.wordnet.*;

public class freespeechwn {

    /**
     * Main entry point. The command-line arguments are concatenated together
     * (separated by spaces) and used as the word form to look up.
     */
    public static void main(String[] args) {
        System.setProperty("wordnet.database.dir", "/wordnet/dict/");
        if (args.length > 1) {
            String wordForm = args[0];
            String type = args[1];
            // SynsetType stype = public static final SynsetType ADJECTIVE;
            /*
             * if (type.equals("adj")) stype = new SynsetType(ADJECTIVE);
             * else if (type.equals("adv")) stype = new SynsetType(ADVERB);
             * else if (type.equals("noun")) stype = new SynsetType(NOUN);
             * else if (type.equals("verb")) stype = new SynsetType(VERB);
             */
             // Get the synsets containing the word form
             Synset[] synsets;
             WordNetDatabase database = WordNetDatabase.getFileInstance();
             if (type.equals("adj")) {
                 synsets = database.getSynsets(wordForm, SynsetType.ADJECTIVE);
             } else if (type.equals("adv")) {
                 synsets = database.getSynsets(wordForm, SynsetType.ADVERB);
             } else if (type.equals("noun")) {
                 synsets = database.getSynsets(wordForm, SynsetType.NOUN);
             } else if (type.equals("verb")) {
                 synsets = database.getSynsets(wordForm, SynsetType.VERB);
             } else {
                 synsets = database.getSynsets(wordForm);
             }
    }

A.9 freessechwn.java

This file contains the source code for the Java program that interacted with WordNet.
// Display the word forms and definitions for synsets retrieved
if (synsets.length > 0) {
    // System.out.println("The following synsets contain "+
    // wordForm + "+" or a possible base form "+
    // "of that text:");
    for (int i = 0; i < synsets.length; i++)
        // System.out.println(";
        String[] wordForms = synsets[i].getWordForms();
        for (int j = 0; j < wordForms.length; j++)
            // System.out.println(wordForms[j]);
        // System.out.println(" "+ synsets[i].getDefinition());
}
else
    System.err.println("No synsets exist that contain "+
    "the word form "+ wordForm + "+");
else
    System.err.println("You must specify "+
    "a word form and type for which to retrieve synsets.");
}
Appendix B

Source Code also on DVD

The source code is also available on the DVD attached to this report.