# Text Mining Wikipedia for Misspelled Words

Jon M. Stacey

MNGT 852, Database Organization [Independent Study]

Dr. Olson

October 11, 2011

## Abstract

This paper employs the scientific method to develop knowledge of the Wikipedia corpus. The question asked is if the percentage of misspellings on Wikipedia, relative to total content, change through time. The hypothesis formed is that the percentage of misspellings actually remains steady through time. A test capable of falsifying the hypothesis is developed that makes use of primitive text mining techniques. Finally, the results are revealed and the findings interpreted and discussed.

## Table of Contents

## Introduction

Have you ever read an article that was written with crowd-sourcing techniques, such as an article on Wikipedia, and wondered if the article was more accurate or syntactically correct than if the article had been written by an individual or small team? I have, and this curiosity lead me to ask a question [many questions in reality]. This paper is a demonstration of both the scientific method and of applying text mining to answer the question and test the hypothesis. The layout of this paper is rather untraditional in that it will mirror the steps of the scientific method nearly one-for-one. The question asked is how does the percentage of misspellings on Wikipedia, relative to total content, change through time?

### *The Scientific Method*

The scientific method is a core part of the scientific curriculum from the early years of grade school. This is because science is the practice of the *scientific method* (Van den Berg 25-26).

The scientific method consists of four primary steps. First, make an observation. This usually occurs without explicitly thinking. For example, you might observe something that piques your curiosity. Second, ask a question. The question narrows down and helps define the problem in manageable terms. Third, form a hypothesis. A hypothesis is for the most parts, just an educated guess. The hypothesis

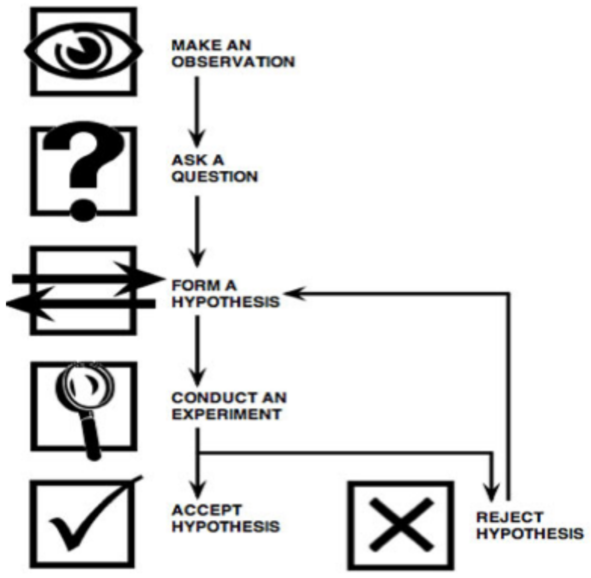


Figure 1. The Scientific Method, from "How the Scientific Method Works."

must be testable, so an experiment can be setup to test the hypothesis. The hypothesis must also be falsifiable, that is, it should be able to reveal if the idea is not true. Fourth, the experiment is conducted to disprove the hypothesis (Harris).

If the experimentation confirms the hypothesis, it can be accepted until proven wrong. Keep in mind that even if a hypothesis has not yet been proven false, that does not means that it is truth, law, or fact. If the experimentation shows the hypothesis to be wrong, a new hypothesis should be formed and the process repeated.

## The Question

When an article is published in a journal, or a book is published, it typically goes through several rounds of professional type editing. During this process, experts comb through the book to correct misspellings and grammar mistakes. Sometimes this step is performed by a single person. At other times it is performed by a small team. Ideally, the end product is a finely polished publication with no mistakes.

In the case of Wikipedia, an article can be edited by anyone. Hundreds and even thousands of people can edit a single page through its lifetime for both content and syntax reasons. Wikipedia is perhaps the best known example of crowd sourcing in that regard. How does this affect the percentage of misspellings on Wikipedia articles through time, relative to total content? Does the percentage of misspellings increase over time as new content is added? Does the percentage of misspellings decrease over time as more errors are caught and freely corrected by anonymous editors?

## Background Research

Before finalizing the hypothesis, research on the background of the underlying question was undertaken. This ensures first, that the question has not been answered before, and second, that any previously known knowledge is taken into account when forming the hypothesis. Think of it as a process of *educated guessing.* At this point, the most important feature of the hypothesis is that it is testable. This background research confirms that a test can be constructed for the hypothesis by using text mining.

### *Text Mining*

It is estimated that around 85% of total data in the world is unstructured (AlSumait et al. 183). Wikipedia is one of these sources of semi-structured data. It contains a large number of categorized and consistently formatted free text (193). It seems natural that a text mining solution would be the approach taken to test the hypothesis that has been hinted at in the introduction and will be formally stated in the next section.

The first step of text mining is to collect the data. The main issue after obtaining the data is cleaning or sterilizing the samples to make sure that they are of high quality. If the dataset is extremely large, then data-sampling techniques can be used to select manageable sets of relevant documents (Weiss, Indurkhya, and Zhang 13-14). These documents are composed of words in many different formats. These words can act as tokens which can then be added to a dictionary to measure the frequency of occurrence in the document (36).

Other research involving the Wikipedia dataset was also conducted. Quite recently, Adler et al. explored combining natural language, reputation, and metadata to detect vandalism on Wikipedia. For example, some of their detection was based on the particular language used within the text. Words such as “coolest” which are inappropriate for an encyclopedia could indicate vandalism when combined with other metrics (Adler et al. 12).

## Hypothesis

*I hypothesize that the percentage of misspellings on Wikipedia articles through time, relative to total content, remains steady.* The idea is that new misspellings are introduced through the introduction of entirely new and niche articles. New errors can also be introduced through the very act of correcting older errors. At the same time, existing content is refined and errors corrected.

Imagine this as a large machine with an input and an output. Going into the machine is new content with new [and existing] errors. Coming out of the machine is refined language. Both the input and output of the machine are the products of different individuals, but the machine itself consists of the masses. As a result, the machine does not act in a linear or progressive fashion. The input will frequently be disconnected from the resulting refined output. If the machine were frozen at various points in time, the inputs and output would rarely align. Finally, the very act of producing the refined content requires it to become input once again, and thus increase the probability of creating new errors.

## Experiment

To test the hypothesis, a simple text mining application was built to check the spelling of Wikipedia articles. The application performs 3 simple steps:

1. Randomly select and retrieve an article.
2. Check the spelling of the article contents at various points in time.
3. Record the results for further analysis and aggregation.

Retrieving a random article is easy, but how do we check the spelling of the content? This is a very difficult question to answer. Have you ever wondered how many words there are in the English language? Not even the Oxford dictionary knows the answer to that question.

> It is impossible to count the number of words in a language, because it's so hard to decide what actually counts as a word. . . . is *dog-tired* a word, or just two other words joined together? Is *hot dog* really two words, since it might also be written as *hot-dog* or even *hotdog* ("How Many Words . . .")?

The solution selected for this test was to generate a dictionary of "valid" English words. This dictionary is called the *valid word dictionary* throughout the rest of this paper. If a word on a Wikipedia page was not contained within the dictionary, then it was counted as a misspelled word.

*Compilation of the Valid Word Dictionary*

Initially, only the *12dicts* word list created by Kevin Atkinson was used (Atkinson). Creating the dictionary required processing multiple text files and weeding out duplicate words. The final dictionary contained just over 111,000 words. Initial samples were showing ~12% misspelling rates with most, if not all, false positives. This was in part because the 12dict word lists does not contain proper nouns. Additionally, the validity of some words could be discussed. For example, is "non-microsoft" a valid word? Perhaps the more correct form would be "not microsoft." Here, you see the complication of defining what is a valid word without context.

To bring the false positive rate down, two additional sources were used to create the final dictionary: SCOWL (Spell Checker Oriented Word Lists) and Wiktionary. The final dictionary consisted of just short of 500,000 words. The application used to generate the final dictionary can be found in appendix-B: The Dictionary Builder.

*Approach to Random Sampling*

The Wikipedia corpus was too large to process within a reasonable amount of time on the hardware available for this research. The entire Wikipedia database download consists of more than 5 terabytes of text ("Wikipedia:Database_download"). To overcome this problem, a random sampling approach was taken.

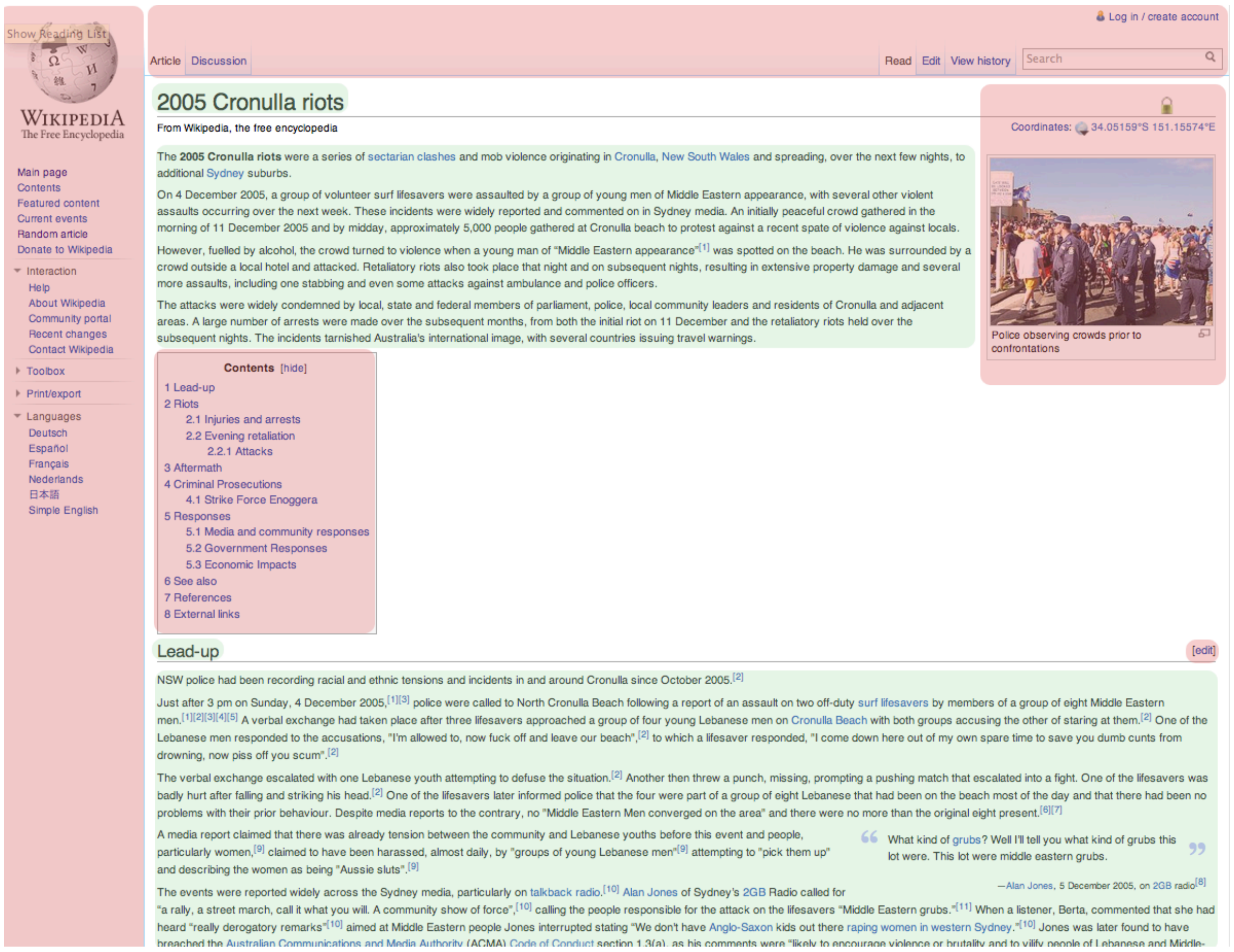
First, the Wikipedia API is used to request random article titles from the dataset. Next, a query is sent to Wikipedia requesting a listing of all revision metadata for each selected article. Then, the list of revisions for each article is sorted and grouped by year. A random revision within each year subgroup is then selected to serve as the sample. In this way, there is one random sample per year for an article over its lifetime.

To obtain a somewhat representative view of the Wikipedia dataset, 2,400 random articles were used. The number of revision samples obtained varies for each article for two reasons: (1) the article may not have existed yet, or (2) there were no revisions to the article during the year.

*Deconstructing a Wikipedia Page for Processing*

The typical Wikipedia page contains a lot of *junk*. Before an article can be fed to the spellchecker, it must be cleaned. After a sample is retrieved, it is passed to the content processor. The processor first cleans the data and then extracts a list of words to be spellchecked. All markup such as HTML must be removed. Additionally, some content is not relevant or is prone to unique syntax which could erroneously skew the test. As such, three content areas are explicitly excluded from being processed: references, gallery tags, and external links.

The colored transparencies in figure 2 demonstrate what content is included and what is excluded. Areas in red are thrown away. Areas in green are extracted and fed into the word separator which extracts each word that will eventually be spellchecked.



2005 Cronulla riots

From Wikipedia, the free encyclopedia

The **2005 Cronulla riots** were a series of sectarian clashes and mob violence originating in Cronulla, New South Wales and spreading, over the next few nights, to additional Sydney suburbs.

On 4 December 2005, a group of volunteer surf lifesavers were assaulted by a group of young men of Middle Eastern appearance, with several other violent assaults occurring over the next week. These incidents were widely reported and commented on in Sydney media. An initially peaceful crowd gathered in the morning of 11 December 2005 and by midday, approximately 5,000 people gathered at Cronulla beach to protest against a recent spate of violence against locals.

However, fuelled by alcohol, the crowd turned to violence when a young man of "Middle Eastern appearance"[1] was spotted on the beach. He was surrounded by a crowd outside a local hotel and attacked. Retaliatory riots also took place that night and on subsequent nights, resulting in extensive property damage and several more assaults, including one stabbing and even some attacks against ambulance and police officers.

The attacks were widely condemned by local, state and federal members of parliament, police, local community leaders and residents of Cronulla and adjacent areas. A large number of arrests were made over the subsequent months, from both the initial riot on 11 December and the retaliatory riots held over the subsequent nights. The incidents tarnished Australia's international image, with several countries issuing travel warnings.

**Contents** [hide]

1 Lead-up
2 Riots
2.1 Injuries and arrests
2.2 Evening retaliation
2.2.1 Attacks
3 Aftermath
4 Criminal Prosecutions
4.1 Strike Force Enoggera
5 Responses
5.1 Media and community responses
5.2 Government Responses
5.3 Economic Impacts
6 See also
7 References
8 External links

Lead-up

NSW police had been recording racial and ethnic tensions and incidents in and around Cronulla since October 2005.[2]

Just after 3 pm on Sunday, 4 December 2005,[1][3] police were called to North Cronulla Beach following a report of an assault on two off-duty surf lifesavers by members of a group of eight Middle Eastern men.[1][2][3][4][5] A verbal exchange had taken place after three lifesavers approached a group of four young Lebanese men on Cronulla Beach with both groups accusing the other of staring at them.[2] One of the Lebanese men responded to the accusations, "I'm allowed to, now fuck off and leave our beach",[2] to which a lifesaver responded, "I come down here out of my own spare time to save you dumb cunts from drowning, now piss off you scum".[2]

The verbal exchange escalated with one Lebanese youth attempting to defuse the situation.[2] Another then threw a punch, missing, prompting a pushing match that escalated into a fight. One of the lifesavers was badly hurt after falling and striking his head.[2] One of the lifesavers later informed police that the four were part of a group of eight Lebanese that had been on the beach most of the day and that there had been no problems with their prior behaviour. Despite media reports to the contrary, no "Middle Eastern Men converged on the area" and there were no more than the original eight present.[6][7]

A media report claimed that there was already tension between the community and Lebanese youths before this event and people, particularly women,[9] claimed to have been harassed, almost daily, by "groups of young Lebanese men"[9] attempting to "pick them up" and describing the women as being "Aussie sluts".[9]

“ What kind of grubs? Well I'll tell you what kind of grubs this lot were. This lot were middle eastern grubs. ”

—Alan Jones, 5 December 2005, on 2GB radio[8]

The events were reported widely across the Sydney media, particularly on talkback radio.[10] Alan Jones of Sydney's 2GB Radio called for "a rally, a street march, call it what you will. A community show of force",[10] calling the people responsible for the attack on the lifesavers "Middle Eastern grubs."[11] When a listener, Berta, commented that she had heard "really derogatory remarks"[10] aimed at Middle Eastern people Jones interrupted stating "We don't have Anglo-Saxon kids out there raping women in western Sydney."[10] Jones was later found to have breached the Australian Communications and Media Authority (ACMA) Code of Conduct section 1.3(a), as his comments were "likely to encourage violence or brutality and to vilify people of Lebanese and Middle-

Figure 2. What’s Included and What’s Excluded.

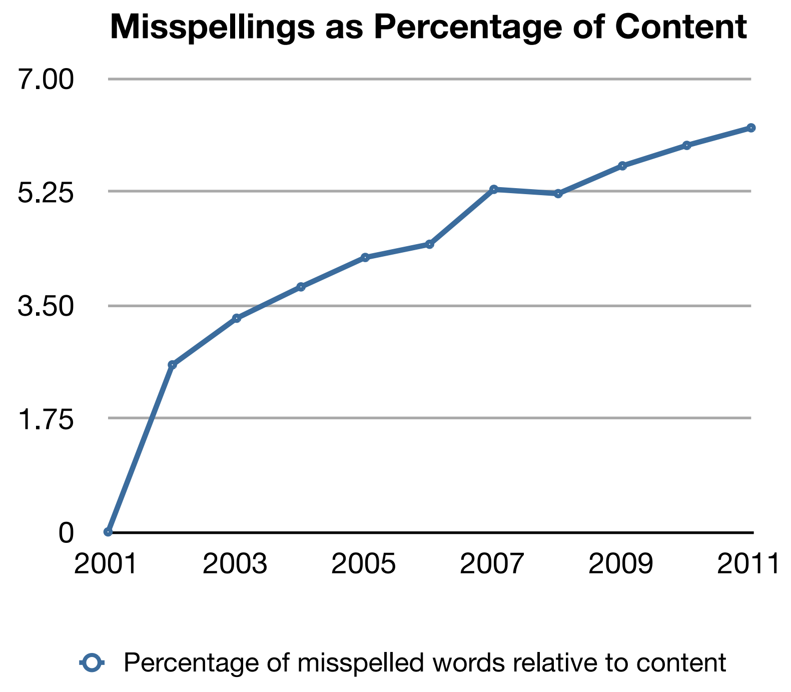
Further post processing is performed on the word list before it is finally spellchecked. This process include throwing away words that cannot be realistically spell checked using a static dictionary. For example, currencies cannot be realistically compared to a dictionary of possible valid values, so they are thrown out. Words which consist of just numbers, such as *1888* are tossed along with numbers that have the trailing letter *s* [e.g. 1940s].

## Results

The percentage of words relative to content that were categorized as misspelled increased year over year consistently. Table 1 and graph 1 show this increasing growth.

| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| - | 0.00 | 2.58 | 3.30 | 3.78 | 4.23 | 4.44 | 5.29 | 5.22 | 5.65 | 5.96 | 6.23 |

Table 1. Average misspellings as percentage of sampled content.

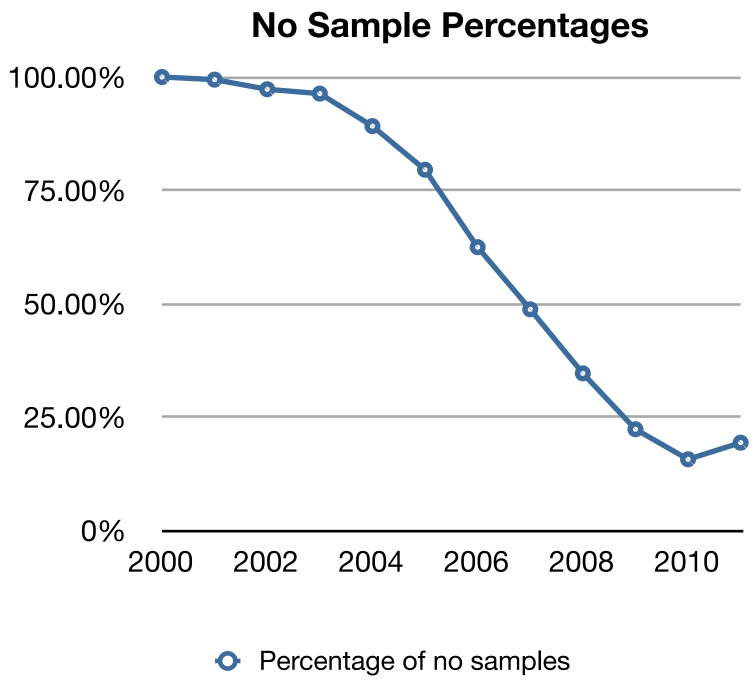


Graph 1. Misspellings as percentage of content.

A surprising discovery was that the majority of pages did not have any available samples until the year 2007. This data is shown in table 2 and graph 2. Wikipedia was launched in 2001, so the graph could perhaps indirectly demonstrate the inverse growth rate of content on the Web site. The increase in pages without samples in 2011 can be temporarily attributed to the fact that the test was conducted in September with three full months remaining in the year.

| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2,400 | 2,386 | 2,335 | 2,312 | 2,140 | 1,909 | 1,499 | 1,170 | 831 | 535 | 376 | 464 |

Table 2. Number of pages without samples by year.



Graph 2. Percentage of pages without samples.

An analysis of potential outliers was conducted to determine potential accuracy rates. Tables 3 through 7 show the number of samples that are above misspelling thresholds of 50%, 25%, 10%, 5%, and 1% respectively. For example, table 3 shows that in the years 2010 and 2011, 40 samples [or 1.67% of the samples] had more than 50% misspellings. The tests show that there are no particular years that are outliers at certain misspelling rate thresholds. The percentage of misspellings increase at every threshold consistently from year to year.

| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 2 | 3 | 6 | 13 | 22 | 31 | 36 | 40 | 40 |
| 0.00% | 0.00% | 0.00% | 0.08% | 0.13% | 0.25% | 0.54% | 0.92% | 1.29% | 1.50% | 1.67% | 1.67% |

Table 3. Outliers [samples with misspelling rates >50%].

| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 5 | 11 | 30 | 42 | 68 | 85 | 114 | 122 | 117 |
| 0.00% | 0.00% | 0.08% | 0.21% | 0.46% | 1.25% | 1.75% | 2.83% | 3.54% | 4.75% | 5.08% | 4.88% |

Table 4. Outliers [samples with misspelling rates >25%]

| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 5 | 26 | 59 | 108 | 178 | 231 | 292 | 344 | 345 |
| 0.00% | 0.00% | 0.08% | 0.21% | 1.08% | 2.46% | 4.50% | 7.42% | 9.63% | 12.17% | 14.33% | 14.38% |

Table 5. Outliers [samples with misspelling rates >10%]

| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 5 | 37 | 93 | 209 | 342 | 438 | 534 | 624 | 648 |
| 0.00% | 0.00% | 0.13% | 0.21% | 1.54% | 3.88% | 8.71% | 14.25% | 18.25% | 22.25% | 26.00% | 27.00% |

Table 6. Outliers [samples with misspelling rates >5%]

| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 30 | 17 | 81 | 162 | 373 | 582 | 750 | 941 | 1,086 | 1,130 |
| 0.00% | 0.00% | 1.25% | 0.71% | 3.38% | 6.75% | 15.54% | 24.25% | 31.25% | 39.21% | 45.25% | 47.08% |

Table 7. Outliers [samples with misspelling rates >1%]

## Findings

On the surface, the results of the test seem to indicate that the hypothesis is false. The percentage of misspellings relative to content have consistently increased year over year. However, it is unlikely that the rate of increase is likely to continue. I would expect to see the misspelling rate begin to level off at some point. However, there is not enough time lapse data to notice any such trend as of September 2011. The results of this test must be taken with a grain of salt. Understanding of the tests weaknesses will facilitate proper interpretation of the results.

### *Test Weaknesses*

The two major weaknesses of the test revolve around two attributes: the use of a limited static valid word dictionary and the fact that the text analyzer itself is very dumb.

Let us first discuss the use of a static valid word dictionary. The global Language Monitor estimates the number of English words to be around 1 million as of the year 2000 ("Number of Words . . ."). That is more than two times the number of words contained in the valid word dictionary. There are also valid words which may not be a part of the English language. For example, references to foreign subjects in which the word is of a different language could be met. On top of this, there are a multitude of syntactically correct forms in which a word may appear. For example, a word may be combined with another word through hyphenation. A word may be in possessive form.

Generation of the valid word dictionary may be more effective by building it from scratch rather than by joining preexisting dictionaries. For example, a larger and modern corpus of English text could be mined for word frequencies. These word frequencies would then determine their inclusion into the dictionary. There are various weaknesses even in this approach, but it may get us one step closer to more accurate results.

The second major weakness is that the text analyzer itself is very dumb. It does not deconstruct a sentence into basic elements. This leads to false positives. A common example of this error can be found with proper nouns which are often excluded from dictionaries. It is possible that a proper noun could be misspelled, but it is more likely to simply not be officially recognized as an English word. The use of frequency metrics when generating the dictionary on a more diverse corpus could be a good area for further research.

The test used in this experiment ignores context, and yet grammar could be a significant determinant of spelling *correctness*. Introducing more advanced natural language processing to help make smarter decisions, for example, given a word's grammatical context, could be employed.

*Conclusions*

The test does not satisfactorily or definitively answer the hypothesis. A new test should be developed to test the hypothesis again. Ideally, the new test should look at the entire Wikipedia corpus, and not a limited sample. Some articles have hundreds or thousands of revisions within the span of a year, and there is a large degree of variability in the content quality that is not captured in a sampled test.

In sum, the data seems to indicate that the hypothesis is false. The percentage of misspellings relative to content increases over time. It is possible that this conclusion is correct, but the probability of this being true is rather low. The probability is greater that the steadily increasing misspelling rates is the result of increasing language complexity used as the Wikipedia corpus grows and matures. The simple minded spellchecker may be unable to cope with this lexical environment. There could also be other outside and random factors that are contributing so consistently towards this result observed. It is my scientific opinion that another test be devised to incorporate a more holistic approach towards natural language processing.

*A View of the Future*

Research that builds on the concept of the spellchecker could lead to the development of more advanced spelling and grammar checking bots. These bots would be continually scraping the Wikipedia corpus to bring the high probability errors to the attention of editors in a central interface. This automated detection could lead to a drastic reduction in errors that may otherwise remain hidden. In this case, an error is considered hidden when a visitor might notice a mistake, but takes no action to correct the error.

If this functionality was combined with the ability to learn through some technique such as a neural network or a decision tree, then the entire Wikipedia corpus could become—to borrow a programming term—unit tested. Ultimately, the goals of the system would be to spur a consistent decline in error rates through time by learning what is considered correct by the masses, and what might be incorrect in new content.

The majority of the work has yet to be done to solve this problem. The stage is certainly set for impressive new developments that change the way we manage an extensive set of unstructured data such as found in Wikipedia.

## Appendix-A: The Wikipedia Sampler

```
#!/usr/bin/env ruby

# =Wikipedia Misspellings Sampler
#
# Version:: 0.2 | September 27, 2011
# Author:: Jon Stacey
# Email:: jon@jonsview.com
# Website:: http://jonsview.com
#
# ==Description
# This program samples Wikipedia via the API.
#
# Random pages are selected, then all revisions for each page collected.
# Revisions are grouped by year and then a random revision for each
# year is chosen. The selected revision is then spell checked against
# a valid word dictionary.
#
# ==Assumptions
# There are too many assumptions to discuss. It would be more
# beneficial to read the code and understand its severe limitations
# and mistakes, or read the corresponding research paper.
#
# ==Usage
# ruby ./sampler.rb
#
# ==License
# Copyright (c) 2011 Jon Stacey. All rights reserved.
#
# I grant the right of modification and redistribution of this application for
# non-profit use under the condition that the above Copyright and author
# information is retained.
#
# ==Disclaimer
# This script is provided "AS-IS" with no warranty or guarantees.
#
# ==Changelog
# 0.2 - 9/27/2011: Speed improvement; cleanup
# 0.1 - 9/22/2011: Initial creation

$:.unshift File.join(File.dirname(__FILE__), "mediawiki-gateway", "lib")

require 'media_wiki'
require 'pp'
require 'wikicloth'
require 'sanitize'
require 'date'
require 'csv'

SAMPLE_SIZE = 400
@dictionary =
File.open('dictionary.txt').readlines.map(&:chomp).map(&:strip).map(&:downcase)
```

```
@wikipedia = MediaWiki::Gateway.new('http://en.wikipedia.org/w/api.php', {:retry_count
=> 20, :maxlag => 30})
@results_file = '/Users/jon/Desktop/results.csv'
@record_years = [2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
2011]

# Write CSV header
CSV.open(@results_file, "a+b") do |csv|
  csv << ["page title"] + @record_years
end

def preprocess(wikitext)
  page = WikiCloth::Parser.new( { :data => wikitext } )
  page = page.to_html
  page = Sanitize.clean(page)

  # Remove galleries (ugly non regex way)
  page = page.split("&lt;/gallery&gt;").collect { |c| c.split("&lt;gallery&gt;")
[0] }.join

  # Remove "[edit]" remnants
  page = page.gsub('[edit]', '')

  # Remove footnotes
  page = page.gsub(/\[[0-9]+\]/, '')
end

def words(page)
  # Get words and downcase them
  words = page.split.map { |word| word.chomp.strip.downcase }

  # Remove some punctuation
  words.map! { |word| word.gsub(/[(\$,?!":;.)]+/, '') }

  words.delete_if do |word|
    # Remove words that consist of only numbers: [0-9]+
    # Remove words that consist of numbers and a trailing "s" or "'s": (('|)s|)
    word = word.gsub(/[0-9]+(('|)s|)/, '')

    # Try to catch percentages
    word = word.gsub(/[0-9.]+%/, '')

    # Remove special case date ranges: example: 1980-90
    word = word.gsub(/[0-9]+-[0-9]+/, '')

    # Forgive me for this disaster, oh computer science overlords.
    # Remove special case numbers such as 19th, 2nd, 1st...
    word = word.gsub(/[0-9]+th/, '')
    word = word.gsub(/[0-9]+nd/, '')
    word = word.gsub(/[0-9]+st/, '')
    word = word.gsub(/[0-9]+rd/, '')


    # Remove special case currency numbers such as $1.5billion
    word = word.gsub(/[0-9]+[a-zA-Z]+/, '')

    # Attempt to find currency
    word = word.gsub(/\(\d{1,3}(?:,?\d{3})*(?:\.\d+)?\)|-?\d{1,3}(?:,?\d{3})*(?:\.\d
+)?/, '')

    if word == '' || word.nil? || word.size == 1
      true
    else
      false
    end
  end

  words
end

def spellcheck(words)
  misspellings = words - @dictionary
  error_percentage = ((misspellings.size.to_f / words.size.to_f) * 100).round(2)

  # puts "Found #{misspellings.size} misspellings. That's roughly #{error_percentage}%
misspelled words."

  misspellings
end

def process_page(page_contents)
  skip_sections = ['references', 'further reading', 'external links']

  words = Array.new
  total_words = 0

  preprocessor = WikiCloth::Parser.new( { :data => page_contents } )

  preprocessor.section_list.each do |section|
    next if skip_sections.include?(section.title.downcase.strip)
    contents = preprocess(section.wikitext)
    words += words(contents)
  end

  misspellings = spellcheck(words)

  # Group by frequency
  frequencies = misspellings.inject(Hash.new(0)) { |hash, value| hash[value] += 1;
hash }
  frequencies.sort_by { |value| frequencies[value] }.last

  return misspellings.size, words.size, frequencies
end

def random_revision_sample(page_title)
  sample_revisions = Hash.new
  hashed_revisions = Hash.new
  all_revisions = @wikipedia.revisions(page_title)
  # Group revisions by year
  all_revisions.each do |rev|
    timestamp = DateTime.parse(rev.attribute('timestamp').to_s)
    (hashed_revisions[timestamp.year] ||= []) << rev.attribute('revid').value
  end

  # Randomly select a single revision for each year of history
  hashed_revisions.each { |year, revisions| sample_revisions[year] =
revisions[rand(revisions.size)] }

  sample_revisions
end

def record_results(page_title, results)
  # ['page', 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011]
  output = [page_title]

  @record_years.each do |year|
    if results[year].nil?
      output << '-'
    else
      output += [results[year][0]]
    end
  end

  CSV.open(@results_file, "a+b") do |csv|
    csv << output
  end
end

def sample(page_title)
  sample_revisions = random_revision_sample(page_title)
  results = Hash.new

  sample_revisions.each do |year, revision|
    # puts "new revision for #{year} - #{revision}"
    page = @wikipedia.get_revision(revision)
    count, total_words, frequencies = process_page(page)
    error_percentage = ((count.to_f / total_words.to_f) * 100).round(2)
    error_percentage = 0.0 unless error_percentage.finite?
    results[year] = [error_percentage]
  end

  record_results(page_title, results)
end


random_pages = @wikipedia.random(SAMPLE_SIZE, 0) # Get 2,400 random pages from the
main namespace

random_pages.each_with_index do |page, index|
  puts "Sampling page #{index+1} of #{SAMPLE_SIZE} (#{page})"
  sample(page)
```

```
end

puts "ALL DONE! Let's see what the results have to say :-)"
```

## Appendix-B: The Dictionary Builder

```
#!/usr/bin/env ruby

# =Create Dictionary Program Overview
#
# Version:: 0.3 | September 27, 2011
# Author:: Jon Stacey
# Email:: jon@jonsview.com
# Website:: http://jonsview.com
#
# ==Description
# This program creates the valid english word dictionary needed for my
# MNGT 852 Database Organization research project.
#
# This application processes and combines word lists from three sources:
# 12-dicts, SCOWL, and Wiktionary.
#
# The resulting dictionary is written to a text file [one word per line]
#
# ==Usage
# ruby ./create_dictionary.rb
#
# ==License
# Copyright (c) 2011 Jon Stacey. All rights reserved.
#
# I grant the right of modification and redistribution of this application
# for non-profit use under the condition that the above Copyright and
# author information is retained.
#
# ==Disclaimer
# This script is provided "AS-IS" with no warranty or guarantees.
#
# ==Changelog
# 0.3 - 9/27/2011: Made pretty for use in paper.
# 0.2 - 9/25/2011: Downcase all words; functionized; incorporate SCOWL word lists,
incorporate Wiktionary dictionary.
# 0.1 - 9/15/2011: Initial creation

def postprocess(words)
  # Only accept ASCII characters.
  words.delete_if do |word|
    true unless word.force_encoding('UTF-8').ascii_only?
  end

  # Lowercase all words
  words.each do |word|
    word.downcase!
  end

  words.uniq!
  words.sort!
end
```

```
def dicts12
  dict_dir = File.new('/Users/jon/Documents/School/Database Organization/Paper #3 -
Wikipedia Project/Research/Word Dictionary/12dicts-5.0/').path

  files = ['2+2gfreq.txt',
           '2+2lemma.txt',
           '2of4brif.txt',
           '2of12.txt',
           '2of12inf.txt',
           '3esl.txt',
           '5desk.txt',
           '6of12.txt',
           'neol2007.txt']

  files.collect! { |x| dict_dir + x } # Prepend dict_dir to each file

  words = Array.new

  files.each do |collection|
    file = File.open(collection)
    strings = file.readlines.map(&:chomp)

    strings.each do |s|
      # Regex: word boundaries. All letters, numbers, periods, hyphens,
      # and spaces.
      s.scan(/[A-Za-z0-9'.\-\s]+/) do |word|
        word.strip!
        words << word.strip if word.split('').last != '-' && word.size > 0
      end
    end
    file.close
  end

  puts words.size.to_s + " words in 12-dicts lists."

  words
end

def scowl
  dict_dir = File.new('/Users/jon/Documents/School/Database Organization/Paper #3 -
Wikipedia Project/Research/Word Dictionary/SCOWL/').path

  files = Dir.glob(dict_dir + '**')

  words = Array.new

  files.each do |collection|
    file = File.open(collection)
    new_words = file.readlines.map(&:chomp)

    words = words + new_words
  end
  puts words.size.to_s + " words in SCOWL."

  words
end

def wiktionary
  require 'nokogiri'
  # Dump downloaded from http://dumps.wikimedia.org/enwiktionary/latest/ on 9/25/2011
@ 12:44PM
  # Dump size is around 200MB.

  input_file = File.new('/Users/jon/Documents/School/Database Organization/Paper #3 -
Wikipedia Project/Research/Word Dictionary/enwiktionary-latest-pages-
articles.xml').path

  reader = Nokogiri::XML::Reader(File.open(input_file))
  words = Array.new

  count = 0
  reader.each do |node|
    if node.name == 'page'
      node.read until node.name == 'title'
      node.read # To get contents
      word = node.value
      next unless word.length > 2
      node.read until node.name == 'text'
      node.read # to get contents
      language = node.value

      if language.include?('==English==') && !language.include?('==Suffix==') && !
language.include?('==Prefix==')
        # Wiktionary words could also be phrases, so we have to sort that out [e.g.
"booster injection"]
        phrases = word.split

        words = words + phrases
        count += 1
        puts count.to_s + ' - ' + word.to_s
      end

      # Nasty hack. This loop gets stuck for some reason, so we will preemptively
terminate at the last word.
      # It's not worth the trouble to find out what's goign wrong here.
      # You'll have to update the count yourself if you use a different wiktionary
dump.
      if count == 377917
        puts words.size.to_s + " words from Wiktionary dump."
        return words
      end

    end
  end # reader.each
end # def wiktionary
```

```
dictionary = dicts12 + scowl + wiktionary
dictionary = postprocess(dictionary)

puts ""
puts "Writing #{dictionary.size} words to dictionary file."

# Write the dictionary to a simple text file
# One word per line
File.open('/Users/jon/Desktop/dictionary.txt', 'w') do |f|
  dictionary.each { |w| f.puts w }
end
```

## Appendix-C: WikiCloth Library Modifications

```
diff --git a/lib/media_wiki/gateway.rb b/lib/media_wiki/gateway.rb
index b718792..eb84dc1 100644
--- a/lib/media_wiki/gateway.rb
+++ b/lib/media_wiki/gateway.rb
@@ -54,6 +54,43 @@ module MediaWiki
       @username = username
     end

+    # Fetch random MediaWiki page title(s).
+    #
+    # [num] Number of pages to fetch. Default = 1.
+    # [namespace] Only list pages in these namespaces [provide the namespace ID 
number]
+    # [limit] The API maximum limitation. 10 is the Wikipedia default for users.
+    #
+    # Returns list of random page titles.
+    def random(num = 1, namespace = '', limit = 10)
+      pages = []
+      begin
+        if num > limit
+          rnlimit = limit
+          num -= limit
+        else
+          rnlimit = num
+          num -= num
+        end
+        form_data = {'action' => 'query', 'list' => 'random', 'rnlimit' => rnlimit, 
'rnnamespace' => namespace}
+        res = make_api_request(form_data).first.elements["query/random"]
+        pages += REXML::XPath.match(res, "//page").map { |x| x.attributes["title"] }
+      end while num > 0
+      return pages
+    end
+
+    # Fetch MediaWiki page in MediaWiki format by revision ID.
+    #
+    # [revid] Page revision ID [revid] to fetch
+    #
+    # Returns content of page as string, nil if the revision does not exist.
+    def get_revision(revid = nil)
+      form_data = {'action' => 'query', 'prop' => 'revisions', 'rvprop' => 'content', 
'revids' => revid}
+      page = make_api_request(form_data).first.elements["query/pages/page"]
+      if valid_page? page
+        page.elements["revisions/rev"].text || ""
+      end
+    end
+
     # Fetch MediaWiki page in MediaWiki format.  Does not follow redirects.
     #
     # [page_title] Page title to fetch
```

```
@@ -66,23 +103,45 @@ module MediaWiki
         page.elements["revisions/rev"].text || ""
       end
     end
-
+
     # Fetch latest revision ID of a MediaWiki page.  Does not follow redirects.
     #
     # [page_title] Page title to fetch
     #
     # Returns revision ID as a string, nil if the page does not exist.
-    def revision(page_title)
+    def latest_revision_id(page_title)
       form_data = {'action' => 'query', 'prop' => 'revisions', 'rvprop' => 'ids', 
'rvlimit' => 1, 'titles' => page_title}
       page = make_api_request(form_data).first.elements["query/pages/page"]
       if valid_page? page
         page.elements["revisions/rev"].attributes["revid"]
       end
     end
+
+    # Fetch listing of all revisions of a MediaWiki page.
+    #
+    # [page_title] Page title to fetch
+    #
+    # Returns ??
+    def revisions(page_title)
+      revisions = []
+      rvstartid = 0
+
+      begin
+        form_data = {'action' => 'query', 'prop' => 'revisions', 'rvprop' => 'ids|
timestamp', 'rvlimit' => 500, 'rvstartid' => rvstartid, 'rvdir' => 'newer', 'titles' 
=> page_title}
+
+        res, rvstartid = make_api_request(form_data, '//query-continue/revisions/
@rvstartid')
+
+        revisions += REXML::XPath.match(res, "//rev")
+
+      end while rvstartid
+
+      revisions
+    end
```

## References

AlSumait, Loulwah, Pu Wang, Carolotta Domeniconi, and Daniel Barbara. "Embedding Semantics in LDA Topic Models." *Text Mining: Application and Theory*. By Michael Berry and Jacob Kogan. Chichester, U.K.: Wiley, 2010. 183-204.

Adler, B. T., Luca Alfaro, Santiago M. Mola-Velasco, Paolo Rosso, and Andrew G. West. "Wikipedia Vandalism Detection: Combining Natural Language, Metadata, and Reputation Features." (2011).

Atkinson, Kevin. "Kevin's Word List Page." *Kevin's Word List Page*. Web. 05 Oct. 2011. <http://wordlist.sourceforge.net/>.

Harris, William. "Scientific Method Steps" HowStuffWorks. Discovery. Web. 05 Oct. 2011. <http://science.howstuffworks.com/innovation/scientific-experiments/scientific-method6.htm>.

"Number of Words in the English Language: 1,010,649.7." The Global Language Monitor. 23 July 2011. Web. 06 Oct. 2011. <http://www.languagemonitor.com/global-english/number-of-words-in-the-english-language-1008879/>.

Van den Berg, Hendrik. *GRBA 852 International Business Textbook*. Lincoln: University of Nebraska-Lincoln, 2011. Web.

Weiss, Sholom M., Nitin Indurkhya, and Tong Zhang. *Fundamentals of Predictive Text Mining*. London: Springer, 2010.

"Wikipedia:Database Download." *Wikipedia, the Free Encyclopedia*. Web. 05 Oct. 2011. <http://en.wikipedia.org/wiki/Wikipedia:Database_download>.