6/21/2016

Evaluating Automatic Keyword Extraction for Internet Reviews

by Alice Leung

Submitted to the department of Mathematics and Computer Science in fulfilment of the requirements for the joint degree of Master's in Language and Communication Technologies

Supervisors: Miguel Couceiro (Nancy), Matt Woodward (Seattle)
UNIVERSITY OF LORRAINE
REALSELF INC.

Abstract

Keyword extraction and topic indexing are long-explored tasks in NLP circles. The task consists of extracting a set of terms or phrases that sum up the content of a piece of text. Professional indexers or the text's authors themselves are usually the individuals responsible for the manual task of keyword extraction; however, many studies in the literature have begun exploration into the idea of automatic keyword extraction, because the process of manual keyword extraction is time-consuming and arduous, and the volume of content to be handled is only getting larger. While this task is often used to help with information retrieval, text mining, and summarization in text corpora, it has taken a slightly different direction when it comes to textual content on the internet. A form of keyword extraction called tagging has been used often for internet content, like blogs, review sites like TripAdvisor, and news articles. Unlike traditional keyword extraction, tagging of internet content serves purposes such as marking content based on users' interests, content discovery or recommendation of new content, and search engine optimization. Furthermore, the tagging process is not done by professional indexers, but by the users and consumers of the content.

RealSelf is a reviews website which provides a platform for consumers to write reviews about plastic surgeons and cosmetic treatments, as well as a forum for new consumers to gain knowledge about various cosmetic procedures. There is a large volume of textual content on RealSelf, the majority of which is community-generated, and is tagged manually by moderators of RealSelf. However, with the various automatic keyword extraction systems available now, we explore the feasibility and accuracy of automating the tagging of content. Three different automatic keyword extraction systems are compared: (1) IBM Watson's Alchemy Language, a commercial API whose algorithms are not publicly known; (2) RAKE, a simple yet powerful unsupervised keyword extraction method based on stopword removal and term frequency; (3) and Maui, a supervised keyword extraction method using novel features and a Naïve Bayes algorithm. A baseline is calculated using TF-IDF, a well-known and powerful statistic for finding important words of a document. A novel method of evaluation is introduced which gives credit for partial results. Preliminary results show that Maui performs the best when evaluated with ten-fold cross-validation, exceeding 50% in both precision and recall. The other two systems achieve slightly low precision but relatively high recall. Alchemy Language performs almost as well as Maui in precision.

Table of Contents

1 Chapter 1 Introduction

Keyword extraction is concerned with the automatic extraction of a set of terms or phrases that best describes the content of a piece text [1]. This task is commonly used for information retrieval, text mining, and summarization, and other text analysis tasks. Extracted keywords correspond to a document's topics. The conventional method of keyword extraction is the TD-IDF measure, which targets words that appear frequently in a document of a corpus but appear infrequently in the rest of the corpus.

With goals similar to those of keyword extraction, tagging is the process of labeling text, typically web-based, with relevant tags. When keyword extraction is performed on web-based content, it overlaps greatly with the definition of tagging, as tagging is popular in internet domains like blogs and forums. Internet domains also have the highest and most rapid influx of new content, for which keyword tags are ever necessary for content discovery and document categorization.

This paper describes an experiment which takes advantage of the multi-contribution aspect of a folksonomy and uses it to create a gold standard. Three different automatic keyword extraction systems are applied on a set of data that has been tagged by the folksonomy, and compared to the manually assigned keywords to determine the best performing system.

1.1 Motivation

Traditionally, professional indexers were tasked with organizing documents by pinning topics to a document. The topics usually come from a pre-defined vocabulary of higher-level concepts. If no vocabulary is used, topics are freely chosen keywords and keyphrases chosen by following pre-defined cataloging guidelines. The process of topic indexing requires a deep understanding of the content as a whole, the parts that make up the whole, and the expected audience of the content [2]. Needless to say, the job of a professional indexer is a time-consuming and labor-intensive one. Moreover, with the booming increase of electronic information, it becomes an infeasible task to do manually; yet it too becomes an increasingly necessary task, because many documents in any electronic repository lack keyphrases and will remain undiscovered. Aside from content discovery, keyphrases also help with text clustering and text search [3].

Tagging of internet content is usually done by end-users for organizing content, or for sharing content that is interesting to them. In other words, the taggers are not professional indexers nor the authors of the data, but rather a system of all users who consume the content. This organic system can also be called a *folksonomy* [4], to which any tags can be added. The benefit of a folksonomy is that the tagging is organic and unbiased, unlike a document tagged by its own author. Another benefit is that masses of people are tagging the same documents, rather than a single indexer contributing one set of tags with no corroboration. On the other hand, therein lies also the drawback of a folksonomy: inconsistency. Due to the low level of consistency usually inherent in folksonomies, most researchers elect not to treat them as a reliable indicator of quality. The inconsistency occurs because people tag with different degrees of specificity, and because of the synonymy and polysemy of human language. Therefore, we would like to be able to leverage the benefits of a folksonomy while minimizing the drawback of inconsistency, in order to make a more balanced gold standard.

There is no question that keyword extraction for documents is important. Automating the process of keyword extraction for internet documents solves the problem of infeasibility of manual tagging. There are already many published papers detailing different supervised and unsupervised methods of automatic keyword extraction. If such methods were to be applied to internet content, surely they would have to be evaluated first and compared to the other methods to determine the most effective method for the domain. Evaluation of automatically extracted keywords is usually done using a gold standard, obtained from a set of keywords manually extracted by a professional indexer. However, depending on the domain, there is a risk of introducing bias into the selection of keywords when taking a gold standard curated by a single person without corroboration. One possible solution is to leverage the folksonomy by using a voting system to create the gold standard.

At the company RealSelf, there are a handful of employees tasked with the job of indexing. They tag all content with relevant keywords. But as the amount of content grows, tagging will become more and more infeasible, assuming the number of indexers at the company is unchanged. The only solutions are to either increase the number of indexers proportionally to the increase in content, or to devise a way to automate the process using automatic keyword extraction systems as machine taggers. A myriad of automatic keyword extraction systems has been developed and improved upon, from simple statistical methods like TF-IDF, to more complex ones like supervised machine learning methods, all with variable performance depending on the content; the question is, which system is best suited to the content of RealSelf?

1.2 RealSelf

RealSelf[1] is a fast-growing company based in Seattle, Washington. It provides a platform for users to access information about plastic surgery and other aesthetic treatments, to contact and write reviews for doctors, and to be part of a caring community of people interested in cosmetic procedures where they can share their experiences and ask questions. In that regard RealSelf can be considered a kind of social network [5].

RealSelf covers over 425 medical and beauty topics, of which about 375 are published and publicly visible; the rest are under improvement and to be published in the future. Topics include specific treatment names, such as Breast Augmentation, Laser Hair Removal, and Liposuction, but also specific brands of treatment, like Botox injections, LASIK laser eye surgery, or Invisalign dental aligners. Each month, RealSelf receives traffic from 9 million unique visitors, while 300,000 of them are monthly active users (MAUs), that is, users who are registered with an account and have at least one instance of activity on the site. These users have contributed over 150,000 treatment reviews to the site, meaning that the vast majority of RealSelf's content is community-generated.

For most visitors of RealSelf, the website helps people who are contemplating undergoing a treatment to find the information they require in order to make an informed decision about whether the treatment is really desired or necessary, and if so, what to expect in terms of costs, the treatment process and recovery process, and individual doctors' characteristics and recommendations, which helps the decision of selecting a doctor. For the loyal users who are regularly active on the site, RealSelf is not only a base of knowledge and doctor reviews, but also a true community where they can give and receive support, interact with others who have undergone or will undergo procedures, and bare their secrets and inhibitions. It is remarkable the extent to which active users are willing to share personal photographs, private life happenings, and internal thoughts with the community in hopes that it may help others along their journey.

1.3 Keyword tagging at RealSelf

Each treatment review (henceforth "review") on the RealSelf site is currently tagged with relevant keywords. The process is done manually by a team of content moderators, whose job it is to approve the

[1] https://www.realself.com/

textual content of the reviews and to tag themes of the review for easy categorization. The words that can be tagged come from a master list numbering 3900 possible tags, ranging from treatment names, symptoms, health conditions, scar types, body parts and skin types, to details or stage of the treatment, like pre-op and post-op, recovery, and medical tourism. The intended purpose of tagging RealSelf's content with keywords is two-fold: (1) to improve content discovery of unseen reviews by making better recommendations of related content when a user views a review; and (2) organize the reviews into categories or topics so that users can easily find reviews that interest them. However, the tags associated with reviews are not providing the expected improvement, due to the fact that the task of human tagging can be inconsistent, and made even more difficult by the necessity of searching through the thousands of tags in the pre-defined tag list to find the tags that suit the content. Tags are intended to be chosen from the master list, given that there is no option to free-tag, nor a simple way of adding tags to the master list without permissions. The result is that the keywords that are finally manually tagged are few in number, possibly uncomprehensive, and inconsistent.

To ameliorate the quality and consistency of tags, and to avoid the human bias during the tagging process and in the limitation of the master list, an experiment with machine tagging was designed; in other words, automatic keyword extraction.

2 Chapter 2 Data Set and Manual Tagging

With the success of this experiment, RealSelf hopes to improve content discovery and provide better tools for discovering content to its users. The short term goal is to tag the content with any and all relevant keywords, while the long term goal is to use the extracted keywords across the entire database of reviews to categorize the reviews and allow users to discover similar content to what they are interested in viewing.

Many strategies for keyword extraction have been discovered, including well-known ones like word frequency analysis, supervised Keyword Extraction Algorithm (KEA), and word co-occurrence relationships, with varying degrees of success. Certainly, however, their success is also largely dependent on the type of content on which they are applied. RealSelf's content is for the large part generated by internet users, and in general is not edited for grammar or spelling before being published. Therefore, the nature of different texts can range from grammatical to nonsensical. For the needs of RealSelf, it is important that we utilize the keyword extraction system that retrieves the best quality of tags from our content specifically.

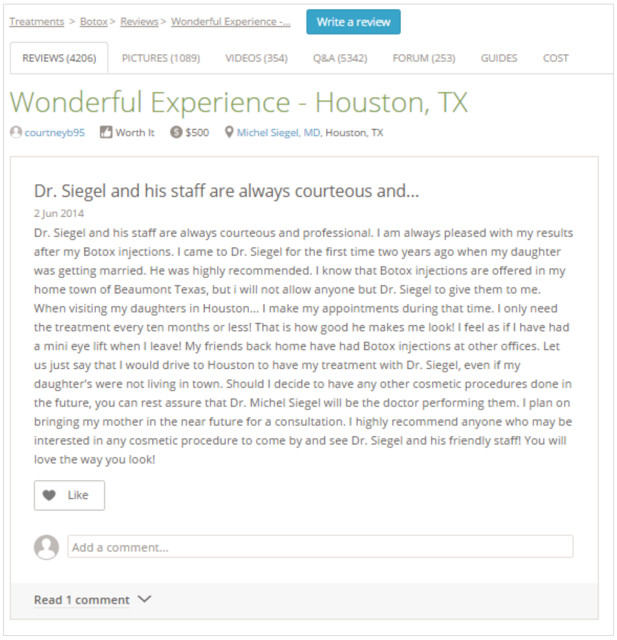
In order to get the best results we can hope for, we decided to apply different keyword extraction techniques on a subset of data, for comparison, and select the technique that seems to best handle this kind of data. First, we selected three different keyword extraction methods: Alchemy Language's keyword extractor, Rapid Automatic Keyword Extraction (RAKE), and Maui algorithm. The data was analyzed using each of the three methods, and keywords were extracted to make up three sets of tags corresponding to each of the methods. Simultaneously, the same data was annotated by humans to set a gold standard to which each set of tags could be compared.

2.1 Data

The content on RealSelf can be roughly categorized by treatment topic, so most content is associated to a topic like Liposuction or Tattoo Removal. Within a topic there are various types of content, including reviews, pictures, videos, questions and answers (Q&A), forum, and guides. Most of this content is currently being tagged manually. However, because reviews make up the bulk of a topic's content, it was decided to focus on reviews for this experiment; after obtaining good results from the experiment, the keyword extraction method could be extended to all other textual content.

Characteristics of reviews. Typically, a user writes a review about a procedure he/she has already undergone, or about his/her experience with a particular doctor. In many cases, the user writes a review entry even before having a procedure, instead documenting his/her progress from the start of the journey, which usually involves the research and planning stages. In either case, users can post as many review updates (henceforth "updates") as they like, whether to present satisfactory or dissatisfactory results, or to document each further step in their journey.

Figure 1 and 2 display sample reviews from the Botox topic and the Mommy Makeover topic. Figure 1 is an example of a user writing only about the experience with the doctor. As can be seen, the Botox review is a single entry written after the patient has already had several injections from the doctor. On the other hand, Figure 2 reads more like journal logs: the original entry was written 18 days pre-operation, and focuses more on the emotions and incremental events. It is followed by periodic updates up until the day of operation, then followed by updates detailing the post-op status and recovery phase (not shown). Therefore, reviews can contain only one entry, or they can include any number of updates resulting in many entries.



Wonderful Experience - Houston, TX

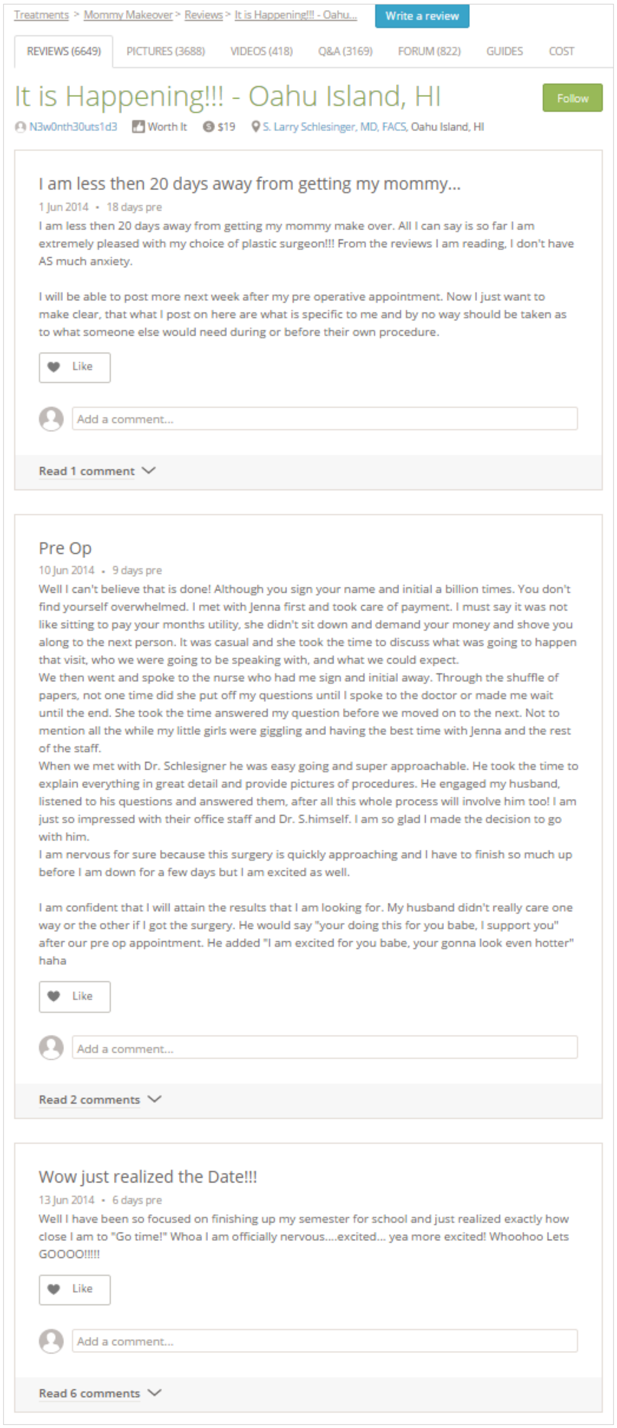
courtneyb95 Worth It $500 Michel Siegel, MD, Houston, TX

Dr. Siegel and his staff are always courteous and...

2 Jun 2014

Dr. Siegel and his staff are always courteous and professional. I am always pleased with my results after my Botox injections. I came to Dr. Siegel for the first time two years ago when my daughter was getting married. He was highly recommended. I know that Botox injections are offered in my home town of Beaumont Texas, but i will not allow anyone but Dr. Siegel to give them to me. When visiting my daughters in Houston... I make my appointments during that time. I only need the treatment every ten months or less! That is how good he makes me look! I feel as if I have had a mini eye lift when I leave! My friends back home have had Botox injections at other offices. Let us just say that I would drive to Houston to have my treatment with Dr. Siegel, even if my daughter's were not living in town. Should I decide to have any other cosmetic procedures done in the future, you can rest assure that Dr. Michel Siegel will be the doctor performing them. I plan on bringing my mother in the near future for a consultation. I highly recommend anyone who may be interested in any cosmetic procedure to come by and see Dr. Siegel and his friendly staff! You will love the way you look!

Figure 1: Sample Botox review



It is Happening!!! - Oahu Island, HI

N3w0nth30uts1d3 Worth It $19 S. Larry Schlesinger, MD, FACS, Oahu Island, HI

I am less then 20 days away from getting my mommy...

1 Jun 2014 • 18 days pre

I am less then 20 days away from getting my mommy make over. All I can say is so far I am extremely pleased with my choice of plastic surgeon!!! From the reviews I am reading, I don't have AS much anxiety.

I will be able to post more next week after my pre operative appointment. Now I just want to make clear, that what I post on here are what is specific to me and by no way should be taken as to what someone else would need during or before their own procedure.

Pre Op

10 Jun 2014 • 9 days pre

Well I can't believe that is done! Although you sign your name and initial a billion times. You don't find yourself overwhelmed. I met with Jenna first and took care of payment. I must say it was not like sitting to pay your months utility, she didn't sit down and demand your money and shove you along to the next person. It was casual and she took the time to discuss what was going to happen that visit, who we were going to be speaking with, and what we could expect.
We then went and spoke to the nurse who had me sign and initial away. Through the shuffle of papers, not one time did she put off my questions until I spoke to the doctor or made me wait until the end. She took the time answered my question before we moved on to the next. Not to mention all the while my little girls were giggling and having the best time with Jenna and the rest of the staff.
When we met with Dr. Schlesigner he was easy going and super approachable. He took the time to explain everything in great detail and provide pictures of procedures. He engaged my husband, listened to his questions and answered them, after all this whole process will involve him too! I am just so impressed with their office staff and Dr. S.himself. I am so glad I made the decision to go with him.
I am nervous for sure because this surgery is quickly approaching and I have to finish so much up before I am down for a few days but I am excited as well.

I am confident that I will attain the results that I am looking for. My husband didn't really care one way or the other if I got the surgery. He would say "your doing this for you babe, I support you" after our pre op appointment. He added "I am excited for you babe, your gonna look even hotter" haha

Wow just realized the Date!!!

13 Jun 2014 • 6 days pre

Well I have been so focused on finishing up my semester for school and just realized exactly how close I am to "Go time!" Whoa I am officially nervous....excited... yea more excited! Whoohoo Lets GOOOO!!!!!

Figure 2: Sample Mommy Makeover review with updates

Distribution of data over topics. RealSelf is a community-centric reviews platform that is constantly updated with new reviews by its users. Given that there are hundreds of thousands of reviews already on the site and new ones being added every hour, it was not feasible to run an experiment with *all* reviews. Instead, 7 specific topics were chosen: Breast Augmentation (BA), Brazilian Butt Lift (BL), Botox (BT), Invisalign (IV), Liposuction (LS), Mommy Makeover (MM), and Rhinoplasty (RP). These topics were chosen for their popularity among users, and for their diversity in *type* of treatment, that is, the targeted body area, whether minimally invasive or not, and length of treatment, preparation, or recovery time.

For example, Breast Augmentation and Brazilian Butt Lift are clearly concentrated on the breast area and backside respectively, while Invisalign is a treatment exclusively for the teeth, and Rhinoplasty is a surgery to reshape the nose. Botox is generally used for the facial area, although injection sites can be the forehead, corners of the eyes, chin area, or anywhere on the face. On the other hand, Liposuction can be done on many different parts of the body, not limited to one. A Mommy Makeover is in fact not a single procedure but rather a combination of procedures which can include breast lift, tummy tuck, and liposuction, designed to return a mother's body to its state before having had children. The difference in body areas ensures that the keywords extracted will vary across topics.

Most treatments can be classified as either invasive or non-invasive/minimally invasive. Breast Augmentation, Brazilian Butt Lift, Liposuction, Mommy Makeover, and Rhinoplasty are invasive surgeries that require going "under the knife". On the other hand, Botox is minimally invasive, and Invisalign is non-invasive. We expect that invasive and non-invasive/minimally invasive treatments will yield different characteristics of reviews regarding the patient's pain level, apprehension, or other emotions.

The different treatment topics also vary in treatment time, including preparation, number of consultations, duration of the actual procedure, and recovery time. Patients who have surgical procedures like Breast Augmentation and Brazilian Butt Lift which are highly invasive often go to several different doctors to get multiple opinions and cost estimates before settling on one doctor. Patients also have to prepare many supplies in advance, and plan for taking leaves from work or school. Further, the recovery time post-operation is rather long, up to several months' time. We expect these points to result in reviews that are possibly shorter in length, but followed up by many updates, because there are more steps in the process and patients tend to document regularly as new developments happen. We expect the same situation of reviews with many updates for Invisalign, because although Invisalign is a non-invasive procedure, the duration of a single procedure can be very extensive, around 12 months. To the contrary, Botox is typically a one-time procedure that lasts only 30 minutes and requires no particular preparation, while the results can be seen within one week after the procedure. For this topic, we expect reviews to be long and comprehensive, probably detailing all the patient's events and emotions in one body of text, and having few or no updates to the review.

For the above reasons, we chose the particular subset of 7 topics to focus on. From each topic, a number of reviews was extracted for testing from June of year 2014. This time point was chosen because it was late enough that the RealSelf community was active and had stable momentum, yet not so recent that users are still making updates to the reviews. It was sensible to have approximately the same amount of data from each topic; however, because the length of reviews can vary so wildly in length, using the *number* of reviews per topic was not a good indicator of quantity of data. Instead, we used word count to measure the amount of data: approximately 5,000 words per topic, regardless of how many reviews or review updates they span, rounded to the nearest whole review in the case of partial reviews. All reviews and updates were saved in comma-separated spreadsheet files, along with the review title and entry titles.

Topic	**Breast Augmentation**	**Brazilian Butt Lift**	**Botox**	**Invisalign**	**Liposuction**	**Mommy Makeover**	**Rhinoplasty**
# of reviews	7	1	26	3	11	6	12
# of total entries	25	32	30	37	40	30	16

Table 1: Distribution of data across treatment topics

Components of documents. When a user begins a review, he/she must designate a review title. For the entry itself, or for writing updates to a previously written review, the user has the option to set an entry title himself/herself; but if no entry title is specified, the first ~50 characters are taken from the beginning of the review text, to make up the entry title (trailing incomplete words are truncated). In Figure 2, it can be seen that a review has a global title in green, while each individual entry has an entry title. The first entry title is identical to the first 11 words of the review text, indicating that no entry title was designated by the author for this entry. The second and third entry titles were indeed written by the author's hand.

This distinction is significant when considering the data to be analyzed, because the entry title is an additional piece of text that should factor into the keyword extraction; however, if the entry title is identical to a portion of the text, the same few words will be analyzed twice, giving unfair weight to those words during analysis of candidate keywords. This issue was handled by discerning the two cases: in the case that the entry title was an original, it was appended to the text body as if it were a separate sentence of the review; in the other case, the entry title was simply discarded from the review, since it would have overlap with the first sentence. This way, the automatic keyword extraction systems would not detect doubled frequencies of the words in the entry title, if the entry title were simply a snippet of the review's first sentence.

2.2 Manual tags

In order to compare different keyword extraction methods with one another, it was necessary to collect manual tags to create a gold standard for evaluating the various automatic keyword extraction systems. The manual tags would also be utilized to train an auto-tagging system that is based on machine learning, which will be described in more detail in section MAUI. Thus, the manual tags would serve two purposes.

The usual procedure of creating a gold standard of keywords is having one professional indexer who extracts the keywords based on his expertise and training, but also based on a set of guidelines defined by the researchers or interested parties. However, for this experiment, we felt that setting a gold standard based on what one individual deems to be the appropriate keywords would introduce unwanted bias when using it to evaluate the automatic systems. Who is to say that the one indexer knows what words are most relevant to the document? Which of his/her selected keywords would be disagreed on by another indexer? Therefore, we decided to take the idea of leveraging a folksonomy to create a gold standard; in other words, to crowdsource the tagging task.

2.2.1 Collecting manual tags

Rather than pitching the tagging task to a crowd of anonymous users on the web, it was prudent to keep the experiment within RealSelf members, as the content may seem distasteful or alien to the general public. We enlisted the help of 23 employees of RealSelf who hold a variety of job positions, from software developers to content moderators to product managers. Each volunteer contributed to the tagging process by tagging 1 to 7 of the topics described in Data. Some, but not all, volunteers tagged all 7 topics; this was originally by design due to their time constraints, but was also an advantage as it imitated the kind of consistency to be expected from a true folksonomy, with some passionate users tagging any and all content, and other more casual users who tag a smaller amount of content that interest them.

For this experiment, the idea of giving a set of guidelines was contemplated, but after some consideration it was decided to give contributors very little constraint, in order to remove the researcher's bias and to allow keywords to emerge organically. The guidelines only enforced a few instructions:

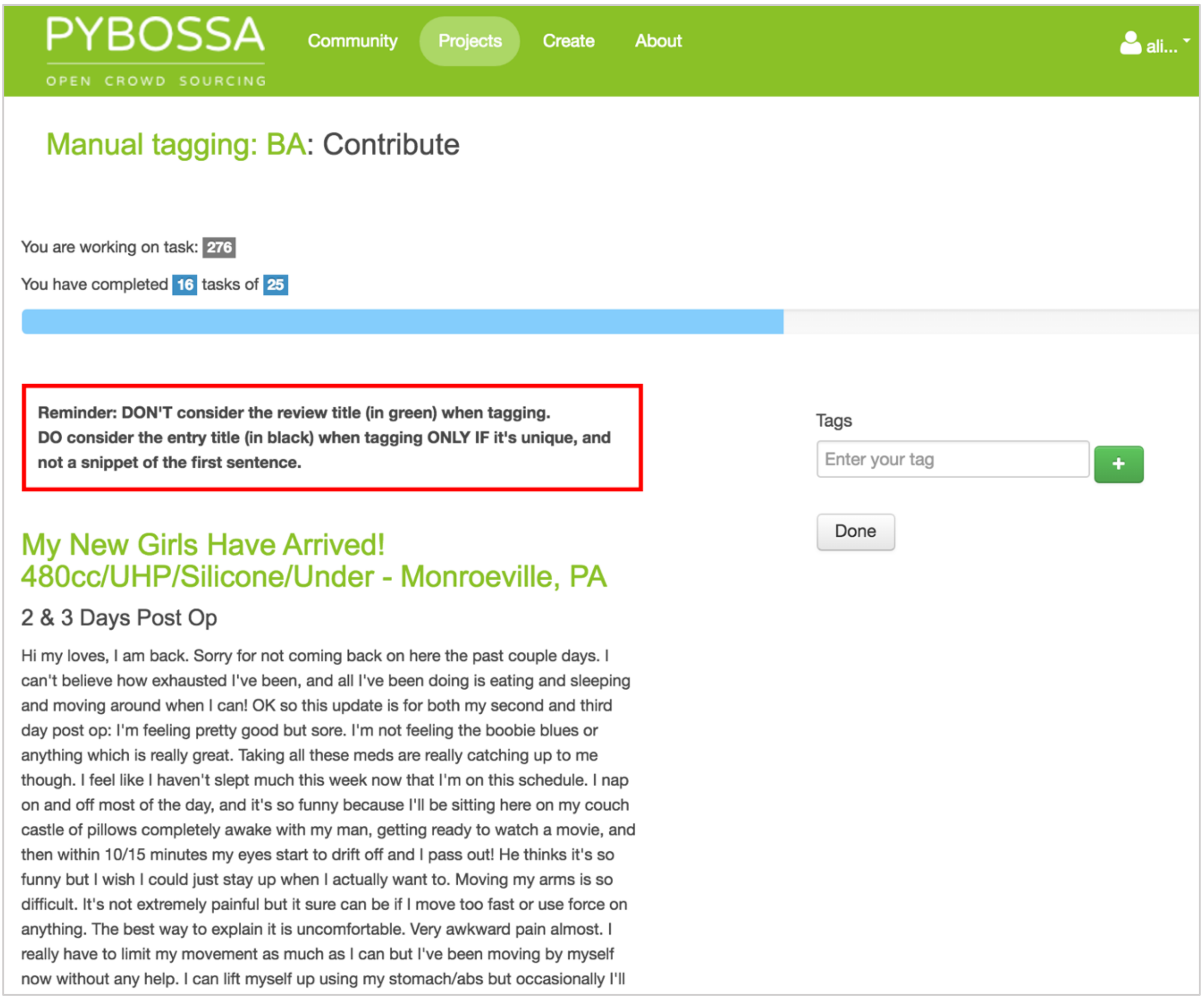
- To *not* consider the review title as part of the data to be tagged, and to consider the entry title as part of the data to be tagged *only* if it is designated by the author, and not simply a snippet of the review's first sentence. This is in line with the way the automatic tagging systems handle the review content, so we expect the exact same content to be analyzed manually and automatically for each review.
- To take keywords appearing explicitly in the text, without changing the form of plurals, verbs, or other variations. The reason for this is that the automatic keyword extraction systems analyze individual documents, without a corpus for context, nor a pre-defined controlled vocabulary of keywords. This means that the keywords extracted by these automatic systems will come from a shallow analysis of the document text, such as term frequency, and will not be transformed using external sources. In order to match the form of the keyword that the systems would extract, it is necessary that the manual taggers extract the keywords in their manifested forms as well.

To collect tags from the group of contributors, a crowdsourcing platform called Pybossa[2] was used. Pybossa is an open source crowdsourcing framework developed by company Scifabric[3] based on inspiration from BOSSA[4] (Berkeley Open System for Skill Aggregation), a software framework for distributed thinking. It is designed for collecting information from volunteers on the internet and consolidating it into data objects for easy analysis [6]. A Pybossa project is essentially an HTML page with some JavaScript that displays a task to the user from the Pybossa server, and records the response given by the user [7]. A common example of a use for this kind of crowdsourcing platform is volunteers classifying photos into categories, or identifying human faces in photos, so that the information can later be used to train computer vision systems and improve their accuracy. We leverage this framework to collect keywords extracted by volunteers after they have been shown a document.

A Pybossa server was installed on a company machine so that employees could access the projects. One Pybossa project was created for every treatment topic, resulting in 7 individual projects. Each project

[2] http://pybossa.com/
[3] http://scifabric.com/
[4] https://boinc.berkeley.edu/trac/wiki/BossaIntro

consists of a number of documents or tasks, as listed as *number of total entries* in Table 1. The user interface and site behavior was designed such that each entry is displayed on a page along with its review title and entry title, as seen in Figure 3. Alongside the document text, a text input box is shown where the user can insert all keywords that he or she deems relevant. When the user is finished extracting keywords, he or she clicks “Done” which prompts a confirmation pop-up, and then stores the keywords to the server. The user then moves on to the next task.



Manual tagging: BA: Contribute

You are working on task: 276

You have completed 16 tasks of 25

Reminder: DON'T consider the review title (in green) when tagging. DO consider the entry title (in black) when tagging ONLY IF it's unique, and not a snippet of the first sentence.

My New Girls Have Arrived! 480cc/UHP/Silicone/Under - Monroeville, PA

2 & 3 Days Post Op

Hi my loves, I am back. Sorry for not coming back on here the past couple days. I can't believe how exhausted I've been, and all I've been doing is eating and sleeping and moving around when I can! OK so this update is for both my second and third day post op: I'm feeling pretty good but sore. I'm not feeling the boobie blues or anything which is really great. Taking all these meds are really catching up to me though. I feel like I haven't slept much this week now that I'm on this schedule. I nap on and off most of the day, and it's so funny because I'll be sitting here on my couch castle of pillows completely awake with my man, getting ready to watch a movie, and then within 10/15 minutes my eyes start to drift off and I pass out! He thinks it's so funny but I wish I could just stay up when I actually want to. Moving my arms is so difficult. It's not extremely painful but it sure can be if I move too fast or use force on anything. The best way to explain it is uncomfortable. Very awkward pain almost. I really have to limit my movement as much as I can but I've been moving by myself now without any help. I can lift myself up using my stomach/abs but occasionally I'll

Figure 3: Pybossa interface for manual tag contributions

Users are required to create an account when they first join the Pybossa project. This allows their input to be tracked individually, and to be linked to a user ID. Pybossa is designed in such a way that each document, or task, could only be completed once by each user. This means that if a user submitted a number of tags for a task and wanted to revise them, it would not be possible. Similarly, if when submitting the tags there was a server error or technical problem that caused an empty list to be submitted, the user could not go back and re-enter the tags. For this reason, a couple of the tag lists had to be discarded, as they were empty or incomplete and could not be corrected. This was one large drawback of using the Pybossa framework.

The process of tagging all documents of one single topic took about 45 minutes to complete. For each document in the data, at least 4 volunteers tagged the same document, with some documents being tagged by as many as 9 volunteers. The data was downloaded, in `.json` format, from the server for each document and for each topic, such that all tags submitted for one document were grouped together for easy analysis.

2.2.2 Creating the gold standard

From the mass of tags contributed by the volunteers, it was necessary to devise a method of consolidating the tags into one gold standard tag set, which would then be used for evaluating the automatic keyword extraction systems, and also for training and testing a model for Maui, a system based on machine learning.

The general idea was to include every keyword that had been tagged by at least two volunteers. For example, if user 23 submits ["Neck Pillow", "Nursing Pillow"], and user 20 submits ["Nursing Pillow", "POS", "waste", "money", "neck pillow"], then “neck pillow” and “nursing pillow” would certainly be included in the final set. Upon inspection of the submitted tags, an unexpected issue was discovered. We cannot trust that all volunteers were maximally meticulous when submitting their free tags, meaning that there was bound to be a number of words submitted that were misspelled, and because they would not match with the correctly spelled word, they could conceivably affect whether a word entered in the final set or not. While some researchers may deem it lacking in academic rigor to edit any input, considering our comparatively small group of volunteers and small amount of data, our gold standard could potentially be skewed unfairly due to a couple of words with missing or swapped letters, so we decided to first clean the data by running it through a spell checker.

Each submitted tag in the data was run through a spell checker called PyEnchant[5], a spell checking library for Python that wraps the generic spell checking library Enchant[6]. After choosing the Dictionary object that PyEnchant will use to check the spelling of words, in this case the American English dictionary, words can be passed into the `check` function which returns True if the word is in the dictionary, and False otherwise [8]. The `check` function was called on all manually tagged keywords, and if it returned False, the `suggest` function was called, which returns suggestions for a misspelled word. The next step was made interactive, such that a person read through the choices in the list of suggestions, and manually selected the correct spelling, or typed one if none of the suggestions were appropriate. Although this introduced an element of human bias into the process, it was the most efficient way to correct the misspelled words. For most misspelled words it was easy to deduce the intended spelling: for example, "confience" was intended to be "confidence"; "expereince" was clearly intended to be "experience"; "volumptous" was meant as "voluptuous". Sometimes, named entities like certain cities, names of medications, or people's last names were not recognized in the dictionary, and it was necessary to add them. This method, though helpful for normalizing the data, could not pinpoint cases where a word is misspelled, but the resulting word is also a correct word in the dictionary; for example, "exception doctor" should be corrected to "exceptional doctor", but since "exception" is an English word as well it would return True on the `check` function, and be passed. Figure 4 shows a snippet of the console when doing the interactive spell checking: correctly spelled words are logged and passed, while incorrectly spelled words pause the program, waiting for user input.

Note that the purpose of spell checking was to correct instances where the volunteer had misspelled a word *unintentionally* during free-tagging, but not when the volunteer had directly extracted a misspelled word from the text as the author had written it. For example, "Knoxville", a large city in the state of Tennessee, is fondly named "Kville" by its residents. "Knoxville" was recognized as a word, and "Kville" was not; while we were tempted to correct instances of "Kville" to "Knoxville", it was decided to retain the author's choice of spelling because the surface text is what will be analyzed by the various auto-tagging systems, and if "Knoxville" is not visible in the surface text, an auto-tagging system would not detect it.

[5] http://pythonhosted.org/pyenchant/
[6] http://www.abisource.com/projects/enchant/

```
depression

anxiety

plastic surgery

general anesthesia

rhinoplasty

surgeons

emotionall prepared

Which is the most likely spelling for emotionall? If the choice isn't
available, type it in.

['emotional', 'emotionally', 'emotion all', 'emotional l']
```

Figure 4: Interactive spell correction of manually extracted keywords. Correctly spelled words are logged above.

Another example is when the author of the review misspells a word herself, like “attachements” instead of the correctly spelled “attachments”. Although “attachements” is the wrong spelling, it was not corrected when it was part of the author’s review text. An auto-tagging system would extract only “attachements” from the text, and if the gold standard instead had “attachments” the system would miss both precision and recall points through no fault of its own.

When all misspelled words were corrected to an appropriate state, all tags were consolidated into a gold standard by the following method: for each document, all tags were counted in a hash structure, and all tags with a count of 2 or more were included in that document’s final set. Thus, a final tag set was curated for each document.

3 Chapter 3 Automatic Keyword Extraction Systems

In this section is detailed the construction and/or implementation of the three keyword extraction systems pitted against each other: Alchemy Language, RAKE, and Maui. In addition, a baseline was also implemented with a TF-IDF measure, so results could be compared to the barest keyword extraction system. The data that was parsed by each machine tagging system was prepared as described in Section 2.1.

3.1 Alchemy Language Keyword Extractor

One of IBM Watson's products is Alchemy Language, a suite of 12 text analytics tools that use natural language techniques to analyze text content and add semantic information [9]. Some of the tools in the collection include sentiment analysis, relation extraction, language detection, entity extraction, and, the most relevant, keyword extraction. A free demo[7] is available on the Alchemy Language website that allows one to try all the tools in real time. The tools are also publicly accessible via a commercial API, called AlchemyAPI. The API receives over three billion calls a month, as many companies already use the text analysis tools for social media monitoring, financial analysis, academic research, and more. A free API key is given for personal use, allowing up to 1,000 daily transactions, while paid accounts start at $0.007 USD per transaction. Unfortunately, because this is a commercial API, the underlying algorithm that drives the tool is not publicly accessible. However, it is known that Alchemy uses complex linguistic, statistical, and neural network machine learning algorithms to perform analysis [10]. Given that Alchemy Language's underlying algorithms are untouchable, and given that the Alchemy Language API is a paid service, we can use it as a baseline for comparison with the other two keyword extraction systems, which do have room for exploration and modification, and are open-source. We harness the Alchemy Language keyword extractor to automatically tag the data set.

For ease of integration, SDKs (software development kits) in 9 popular programming languages are provided to easily access the Alchemy functions [11]. In the Python SDK, chosen for use in this experiment, an `AlchemyAPI` object is created which contains as functions each of the 12 text analytics tools. Each function makes an HTTP POST call to its corresponding API endpoint, inserting the two parameters that were passed into the function call: `flavor` and `data`. `flavor` refers to the type of content, whether

[7] https://www.alchemyapi.com/products/demo/alchemylanguage

HTML, plaintext, or web-based content (from a URL). In this experiment, we opted to submit the data in plaintext form, since the same text would be used in the other keyword extraction methods for consistency of file formatting. `data` refers to the textual data to be analyzed, either the HTML code, text, or the webpage fetched from the URL. The returned output can be in XML, JSON, or linked data formats, but in the Python SDK the response is returned in a Python object, already converted from JSON.

We use the SDK to call AlchemyAPI's `TextGetRankedKeywords` API endpoint on reviews from the seven topics. Extracted keywords are returned with a corresponding relevance score between 0 and 1.0, with 1.0 being the most relevant. All keywords with a score above 0.5 were saved to file; the threshold of 0.5 was decided upon after a cursory look at the quality of the extracted keywords, concluding that most keywords with a score under 0.5 were decidedly irrelevant.

3.2 Rapid Automatic Keyword Extraction (RAKE)

RAKE was developed by Rose, Engel and Cramer [12] as an alternative to corpus-based keyword extraction methods. Unlike traditional methods that compare word frequencies, RAKE operates on the premise that keywords and keyphrases do not overlap with stopwords, and that the more non-stopwords are found adjacent to each other, the higher the probability that it is a keyphrase. In other words, candidates of longer length have higher relevance scores than candidates of shorter length. So a candidate "natural numbers" will have a higher relevance score than "numbers", and a candidate "minimal supporting sets" will have a higher relevance score than "supporting sets". Therefore, the RAKE method is highly dependent on a reliable and comprehensive list of stopwords, because the list determines where the text is divided up and where the boundaries of candidate keywords are, which decides the candidates' length.

Candidate selection. First, the text is split at the sentence level. Sentence delimiters include the usual period ("."), question mark ("?"), and exclamation point ("!"), but also comma (","), colon and semicolon (":", ";"), right and left parentheses ("(", ")"), right and left square brackets ("[", "]"), tab and newline characters ("\t", "\n"), apostrophe ("'"), hyphen ("-"), and double quotes ("). Perhaps it is more appropriate to name such an extensive list of delimiters *phrase delimiters* rather than *sentence delimiters*.

The words contained in the stopword list, or stoplist, are crucial to determining the candidate keyphrases. The original Python implementation[8] of RAKE is improved on by Alyona Medelyan [13] in her own project[9], and uses a stoplist called SMART Stoplist, the list of stopwords used in the SMART information retrieval system developed at Cornell University in the 1960s [14]. This stoplist is lengthier than most typical stoplists, and although it is considered the standard stoplist for English [15] and is used by multiple information systems and NLP tasks [16] [17] [18] it is not uncommon to edit it for certain implementations to better suit the task or domain.

[8] https://github.com/aneesha/RAKE
[9] https://github.com/zelandiya/RAKE-tutorial

```
Entry: 314089


I'm looking for an affordable place to have cosmetic surgery done

| | | | affordable place | | cosmetic surgery |


 I've read so many reviews on here and look at hundreds of pictures
and tons of doctors

| read | | reviews | | | | | hundreds | pictures | tons | doctors


 I don't even know where to start

| | | | | | start


 Completely lost and oblivious

completely lost | oblivious


 None of my friends or family have had surgery done

| | | friends | family | | surgery |


I am in complete need of a tummy tuck

| | | complete | | | tummy tuck


 then I also want lipo in my arms

| | | | lipo | | arms


 Hopefully a previous patient with great results can help me out

| | previous patient | great results | | | |
```

Figure 5: Stopword removal in sentences of a review

All stopwords in a given document are replaced with a boundary symbol ("|"), so that non-stopwords can be extracted easily, whether singletons or phrases; phrases consisting of multiple words must have fewer words than the `max_words_length`, which by default is 5, but can be set by the researcher. Phrases consisting of more than the maximum word length are discarded. Figure 5 illustrates the process of stopword removal, leaving only non-stopwords to be considered as candidate keyphrases.

Keyword score calculation. A score needs to be calculated for each of the candidate keywords. First, scores are calculated for each individual word. Then, the score of a sequence of words is the sum of each member word's individual score. Each word has a degree score and a frequency score.

Degree score is the sum of lengths of all sequences where the word appears. A graph of word co-occurrences is made, where a word is considered to co-occur with another if they are found in the same phrase. So if a word "minimal" appears in a bigram "minimal sets" and a trigram "minimal supporting sets", the degree score for word "minimal" is (2 - 1) + (3 - 1) = 3; the -1 on each side accounts for the number of words "minimal" occurs with, not the number of words in the sequence. Thus, degree score favors words that occur often in longer candidate keywords [12].

Frequency score is the number of times the word occurs in the whole text, regardless of which words it co-occurs with.

According to the paper by Rose et al. [12], each individual word score is calculated by dividing degree by frequency, *degree/frequency*. However, I have discovered a significant discrepancy between the authors' paper and the Python implementation regarding individual word score calculation: In the Python code, individual word score is calculated by first adding degree and frequency, and then dividing by frequency, (*degree+frequency)/frequency*. While this seems trivial, it can cause slight differences in the ranking of individual word scores and thus candidate scores, causing candidates of longer word length to rank higher than others.

Individual word scores are summed to get the candidate score, and then sorted by score. Rose et. al take the top $\frac{1}{3}$ of total keywords extracted to be the finalized list of keywords. Another option is to take all keywords above a certain threshold; for example, to take all keywords with a score above 2.0, as in Figure 6.

```
Keyword:  affordable place , score:  4.0

Keyword:  great results , score:  4.0

Keyword:  tummy tuck , score:  4.0

Keyword:  previous patient , score:  4.0

Keyword:  completely lost , score:  4.0

Keyword:  cosmetic surgery , score:  3.5
```

Figure 6: Keywords extracted from review in Figure 4

Adaptation of RAKE to the context of RealSelf. The content of reviews on RealSelf are of a unique kind; unlike documents that are traditionally used for testing and developing NLP tasks, such as newspapers, technical documents, or storybooks, the content on RealSelf is informal and unedited, created by a global public in a social network environment, where there is often less regard for spelling and grammar, and frequent usage of abbreviations, acronyms, and slang. This may cause difficulties when applying traditional methods of keyword extraction. The advantage of RAKE over other keyword extraction methods is that the code is entirely editable based on the user's needs, and can be customizable to an extent to suit the content of this experiment.

- **Contracted forms:** As mentioned above, the stoplist can be added to or shortened, or replaced with a different stoplist, which can tremendously affect the keywords that are extracted. SMART Stoplist, which comes bundled with the Python implementation of RAKE, seems to cover for informal texts more than most other respected stoplists, containing stopwords like "ain't", "c'mon", "hello", and many contractions like "isn't", "they'll", and "i've". But while these kinds of truncated words are indicative of less formal writing, internet users writing blogs and other similar texts can be counted on to not only use the contracted forms, but also to omit the apostrophes that bind two words together. Therefore, the first change to the SMART Stoplist was the addition of all contracted forms, minus the apostrophes, so these words would be targeted as stopwords just as the apostrophe-marked forms would.
- **Expletives:** Other changes to the stoplist were made in a loop, by testing out the program to see what keywords were extracted, modifying the stoplist to allow relevant keywords, or to disallow irrelevant keywords, and repeating the process. For example, some female writers, in bouts of

emotion, include expletives that are cleverly censored using asterisks (*) or other symbols to replace some or all of the letters of the expletive. Uncensored profanity would likely have been barred from the site, so all instances of profanity are censored this way. To prevent censored words from becoming candidate keywords, a regular expression `\w*\*+\w*.*` was added to the stoplist, matching any word containing at least one asterisk.

- **Slang and idiosyncracies:** In addition, a few stopwords were appended to the list that were idiosyncratic, or tend to only appear in internet texts. For example, slang words like "sup", "btw", and "y'all/yall" appeared frequently in the texts, which would not be targeted by the stoplist, although the extended forms "what's", "up", "by", "the", "way", "you", "all" would be accounted for by the stoplist. Also, it was not uncommon to find snippets in Spanish, particularly "hola", as many patients elect to go abroad to Latin American countries like the Domincan Republic for their treatments.
- **Coverage of verbs and synonyms:** Finally, upon closer inspection of the stoplist, its broad coverage was lacking in some areas. For example, though many verbs are included in the stoplist, coverage of verb forms Is inconsistent: for the verb "consider", the list included the first person present and present participle forms ("consider", "considering"), but is missing the third person present and past participle forms ("considers", "considered"). This occurs to some degree with most other verbs in the stoplist, so some additions were made so all verb forms were included. Another example is of synonyms of verbs: while "suggest" was present in the stoplist, one of its synonyms "recommend" was not, yet "recommend" is a word that appears often in review texts on RealSelf. Therefore, several additions of verb synonyms and their verb forms were made.
- **Hyphens:** Another complication discovered when using the SMART stoplist was the handling of hyphens (-). In general, hyphens or dashes are used for joining words together [19], like "pre-op", or for inserting parenthetical statements, e.g. "A flock of sparrows – some of them juveniles – alighted and sang" [20]. However, as described above, the process of phrase splitting during candidate selection delimits on conventional sentence delimiters like the period and the question mark, but also on hyphens. This is appropriate for cases where the hyphen is used for parenthetical statements, because it is desirable to treat the parenthetical statement as a sentence separate from the enclosing sentence. On the other hand, it is *not* desirable to have this behavior when the hyphen compounds two words together, especially in cases where the meaning of the compound cannot be deduced without both words joined together. Because we found it more important to keep the integrity of joined words, than to reliably separate on

parenthetical statements, this was resolved by removing the hyphen from the list of phrase delimiters. To make things even more complicated, internet writers are prone to adding unnecessary spaces before or after the hyphen, as in "pre -op" or "pre- op", or not using spaces at all when making parenthetical asides. Unfortunately, that makes it very difficult to predict which function of the hyphen the author intended; in formal texts, a hyphen used for parenthetical asides should be preceded and followed by a space, while hyphens used for joining words should not be bordered by spaces on either side. Therefore, two stopwords were added to the stoplist corresponding to these space-hyphen combinations (" -" and "- ").

- **Word boundaries:** At this point, another issue came to light. According to the Python code, while the stoplist is being compiled into a regular expression pattern, a word boundary ("\b") is appended to the beginning and end of every stopword in the list. Thus a word "and" would be compiled as `\band\b`. This was evidently to avoid matching instances of stopwords when they appeared as part of another word, like "and" within "stand". Because the hyphen was removed from the list of phrase delimiters, hyphens remained in the text, and, as hyphens are considered word boundaries, stopwords that were joined by hyphen with non-stopwords would be deleted in the stopword removal process. For example, if "follow" (a non-stopword) is joined with "up" (a stopword) to make "follow-up", "up" would be deleted, leaving just "follow-". This was not problematic before, because the hyphen was considered a phrase delimiter and did not remain in the text when parsing for stopwords.

  The solution to this was to create a custom word boundary for this task that excludes hyphens. Following the explanation of boundaries at Rex Egg [21], a word boundary `\b` can be understood as a non-character-consuming look-around that checks, on the left of a word, that the character preceding is a non-word character `\W` and the character following is a word character `\w`; and on the right of a word, that the character preceding is a word character and the character following is a non-word character. In regular expression terms, the word boundary on the left of a word is equivalent to `(?=\w)(?<!\w)`, where `(?=\w)` looks ahead to match a word character `\w,` and `(?<!\w)` looks behind to *not* match a word character `!\w`. Equivalently, the word boundary on the right of a word, in regular expression terms, is equivalent to `(?<=\w)(?!\w)`, where `(?<=\w)` looks behind to match a word character, and `(?!\w)` looks ahead to not match a word character. These two patterns can be combined easily in an alternation using a vertical bar, `(?=\w)(?<!\w)|(?<=\w)(?!\w)`. With this information on the break-down of the simple word boundary `\b`, it was possible to modify the word boundary to exclude hyphens (and apostrophes, which caused similar issues): on the left side of a word, `(?=\w)(?<![\w\-\’])` and on the right side of a word, `(?<=\w)(?![\w\-\’])`. Thus, stopwords connected by hyphens to non-stopwords would not get removed, so words like “follow-up” and “take-out” would remain intact.

Evidently from the explanatory paragraphs above, there is no doubt that the stoplist plays the largest part in deciding what is a keyword and what is not. However, it was not the only adjustment made to mold RAKE to RealSelf’s content.

- The sentence tokenizer built into RAKE consists of a regular expression containing the usual sentence delimiters, period, question mark, and exclamation mark, but also an array of other phrase delimiters like comma, parentheses, square brackets, new line and tabular, colon and semi-colon, and ellipses. While we were in agreement with the splitting of sentences on these various delimiters, there was a case to be handled, not uncommon to text analytics: the handling of splitting sentences on periods, unless the period is part of an abbreviation like “Mr.” or “Dr.”. In this domain, doctors are a perpetual theme in textual content, so we can be certain that there are a lot of references to doctors by the abbreviation “Dr.”, so the regular expression for sentence splitting was modified to include four regular expression look-behinds in front of the original pattern: `(?<!\bMr)(?<!\bMs)(?<!\bMrs)(?<!\bDr)` in order to preserve the “Dr. <last-name>“ construct.

Characteristics of keywords extracted from RAKE. As explained in the keyword score calculation section above, a candidate keyword’s score is based on each member word’s degree and frequency score. It stands to reason that the more member words in a candidate keyword, the higher the score of the candidate. For example, “post op” would score lower than “three months post op”, since the score of “three months post op” is composed of four individual scores while “post op” is only composed of two. The result is that RAKE heavily prefers bigrams, trigrams, and four-grams, and rarely passes through unigrams as final keywords. The exception is when a unigram, or a single word, occurs frequently in the document, typically at least three times. The drawback of this is that single words that would certainly be considered keywords would likely be dismissed, like city names, doctor last names, and specific treatment names consisting of only one word. To illustrate another example, if a document contains a patient’s proclamation of being “ecstatic” with her results, and a second document contains another patient proclaiming to be “absolutely ecstatic” with her results, “absolutely ecstatic” (but not “ecstatic”) would be extracted from the second document, while “ecstatic” would be entirely missed from the first document. In summary, almost all keywords extracted by RAKE are composed of two or three words, where the first word is often a modifier.

3.3 Maui

Maui, which stands for Multi-Purpose Automatic Topic Indexing, has at its core the Weka machine learning toolkit [22], developed by the Machine Learning Group at the University of Waikato[10]. Maui itself was developed by Olena Medelyan [3] as a reincarnation or improvement of Kea, the well-known Keyword Extraction Algorithm [23] also developed by the Machine Learning Group.

Kea. Kea is an algorithm for extracting keywords or keyphrases from a document. It can be considered a supervised machine learning method, as it trains a model based on manually labeled documents, and predicts keywords from unseen documents. The process can be roughly divided into two stages: candidate selection and machine learning based filtering. In the candidate selection stage, Kea splits the text into sentence sequences defined by phrase boundaries, such as punctuation marks, dashes, parentheses, and numbers, and then splits these sequences into tokens and n-grams. Non-alphanumeric characters and numbers are deleted, phrases beginning or ending with a stopword are eliminated, and phrases consisting of only a proper noun are deleted. Of the remaining phrases, all words are stemmed using the Lovins stemmer [24], and stemmed phrases that occur only once in the text are eliminated. This speeds up the training and extraction processes without affecting the results. Kea takes the phrases with a length of up to three as candidate phrases.

The next stage of the keyword extraction process is assigning features to each of the candidate phrases so that a classifier can learn what is a keyword and what is not. The original algorithm utilized three features: TF-IDF scores, position of first occurrence, and what Medelyan [4] calls keyphraseness, or keyphrase-frequency. TF-IDF, a standard technique in text analysis, gives weight to a term to determine how important a word is to an individual document as opposed to the rest of the corpus [25]. Position of first occurrence is calculated as the number of words that precede the first appearance of the word, divided by the number of words in the document; the idea is that very high or very low values indicate

[10] http://www.cs.waikato.ac.nz/ml/weka/

high probability that the word is a keyword, because they appear in opening document parts like title and introduction, or in closing document parts like the conclusion [4]. Finally, keyphrase-frequency is the the number of times an assigned keyword appears as an assigned keyword in all training documents besides the document in question. This third attribute evidently only makes sense if the documents to be analyzed are of the same domain. Together, the three numerical values are discretized and processed by a Naïve Bayes method, training a model which can then compute the estimated probability that a word is a keyword.

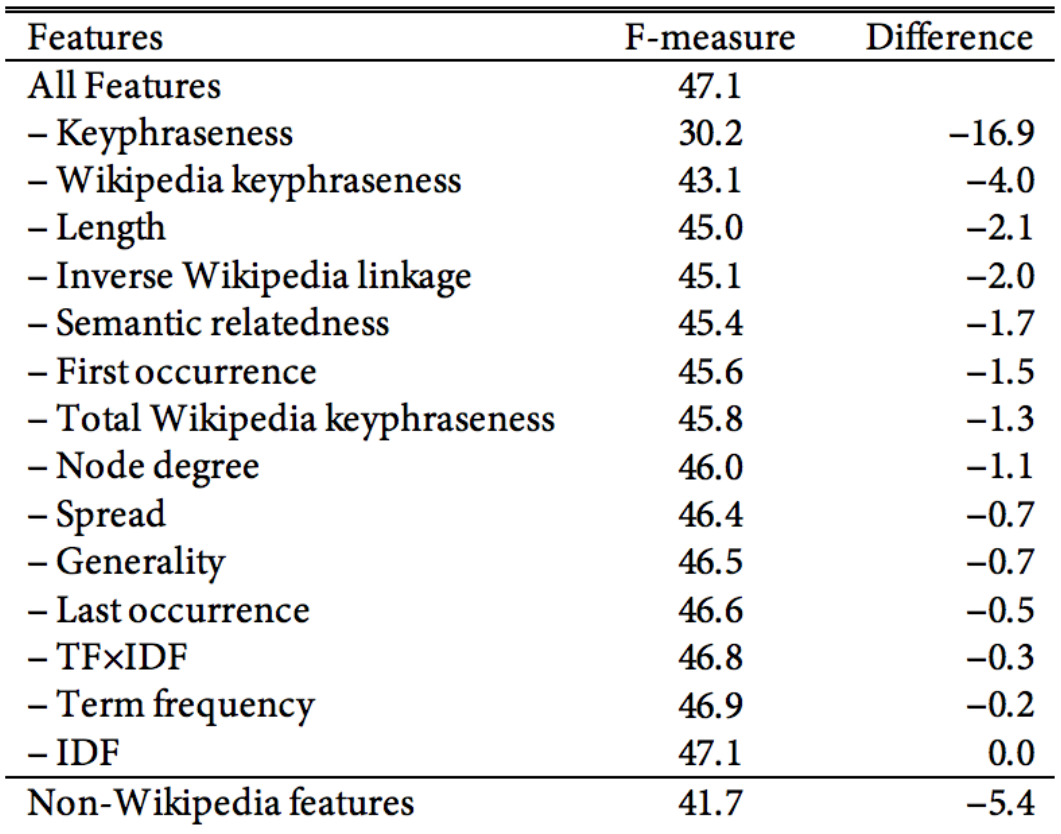
Maui. Maui[11] is an open-source Java implementation built on Kea, but has significant additions and improvements. Kea was limited to keyword extraction, which is one kind of topic indexing, while Maui can perform more kinds of topic indexing, including term assignment from a controlled external vocabulary, and mapping keywords to terms in Wikipedia. Furthermore, Maui contains the complete machine learning library of Weka, whereas Kea only contains a few of Weka's classes. Finally, Maui includes the Jena software library, for incorporating external controlled vocabularies, and Wikipedia Miner [26], for providing object-oriented access to Wikipedia dumps and computing semantic relatedness between articles.

Maui's process can be described in the same two stages. While the stage of candidate selection is identical to that of Kea, the stage of feature selection has significant additions that make for an improved keyword extraction system. In addition to the three baseline features used in Kea – TF-IDF score, position of first occurrence, and keyphraseness – three features are added that have been useful in previous work, as well as three novel features that have not been previously evaluated. The six additional features include phrase length, node degree, Wikipedia-based keyphraseness, spread, semantic relatedness, and inverse Wikipedia linkage. **Term length** as a feature is based on the reasoning that the longer a phrase is, the greater the degree of specificity, which can be useful in training as the model learns the preferred specificity in the training corpus. **Node degree** is a measure of the semantic relatedness between candidate tags, which in this incarnation is calculated using Wikipedia where the node degree of a word is the number of Wikipedia hyperlinks connecting its corresponding Wikipedia page to other Wikipedia pages corresponding to other candidate words from the same document; given the assumption that a document about a certain topic will have many related concepts or words, so a high node degree indicates

[11] https://code.google.com/archive/p/maui-indexer/downloads

that a candidate is highly connected to other phrases in the document and thus more likely to be significant. **Wikipedia-based keyphraseness** is the likelihood that a phrase is a link in the Wikipedia corpus, calculated by dividing the number of Wikipedia pages in which the phrase appears as a hyperlink by the number of Wikipedia pages containing the phrase, and multiplying this number by the phrase's document frequency. **Spread** describes the distance between a word's first and last occurrence in a document relative to the number of words in the document, which intends to identify phrases that are mentioned both in the beginning and the end of a document. **Semantic relatedness** describes a method different from the node degree feature, employing a Wikipedia based approach proposed by Milne and Witten [27] that determines the total relatedness of a phrase to all other candidate phrases based on the cosine similarity between the sets of outgoing links (links directing out to other articles) and incoming links (links from other articles directing into the article). Finally, **Inverse Wikipedia linkage** calculates a score by normalizing the number of Wikipedia articles that link to a candidate phrase's Wikipedia article and dividing by the total number of links in Wikipedia (the same principle as TF-IDF calculation), playing on the assumption that a concept that is frequently used to describe other concepts has a high likelihood of being a keyword in the document.

Medelyan [3] compares the level of influence each of these features has on performance by applying Maui's algorithm on an internet collection called CiteULike-180 that has been collaboratively tagged. 10-fold cross-validation is performed on a bagged decision trees model, and the features are eliminated one by one to show the difference in each feature's individual contribution to the overall result. Table 2 shows the contribution of each individual feature, as well as some features that are not actually employed in the algorithm.



Features	F-measure	Difference
All Features	47.1	
– Keyphraseness	30.2	−16.9
– Wikipedia keyphraseness	43.1	−4.0
– Length	45.0	−2.1
– Inverse Wikipedia linkage	45.1	−2.0
– Semantic relatedness	45.4	−1.7
– First occurrence	45.6	−1.5
– Total Wikipedia keyphraseness	45.8	−1.3
– Node degree	46.0	−1.1
– Spread	46.4	−0.7
– Generality	46.5	−0.7
– Last occurrence	46.6	−0.5
– TF×IDF	46.8	−0.3
– Term frequency	46.9	−0.2
– IDF	47.1	0.0
Non-Wikipedia features	41.7	−5.4

Table 2: Impact of individual features on the CiteULike-180 internet collection. The "Difference" column shows the negative impact on performance when the feature is removed.

Spread, for example, is the distance between the **First occurrence** and **Last occurrence** normalized by the document's length in words, though first and last occurrence are not used as features in training. **Term frequency** and **IDF** make up the **TF-IDF** score, though only TF-IDF is used. Note that the **Generality** feature for Wikipedia concepts is fairly similar to **term length**; instead of taking the length of the phrase as an indicator of specificity, take the generality of a Wikipedia article by analyzing its position in the hierarchy, where the higher up a concept is, the more general it typically is.

From the table it is visible that TF-IDF, usually one of the strongest features, is one of the weakest contributors when all other features are combined. Keyphraseness is by far the strongest feature, contributing 16.9 points in F-Measure, followed by Wikipedia keyphraseness. Wikipedia-based features have a high impact on the performance, since the last row excludes all features relying on Wikipedia as a knowledge source, and the F-Measure drops by 5.4 points.

Training a model. There are two steps required to perform supervised keyword extraction: (1) building and saving a model trained on previously labeled data, and (2) using the same model to make predictions about unseen data. Once a model has been built it can be reused as necessary on any new document, though it is recommended to train on data that it is in the same domain as the unseen data. Maui requires a set of labeled documents for training, so that, after candidates with a frequency of one have been discarded from the document to reduce the size of the training set, the candidate keywords that are extracted can receive a value indicating whether it is a positive example or not. The positive and negative class values combined with the feature values train a model that can be used to predict the class values of unseen documents. Using the Weka toolkit included in Maui, any classifier can be utilized to train the model, though Medelyan [3] discovered that Naïve Bayes and bagged decision trees perform better than other classifiers.

Training Maui on RealSelf content. Since Maui is a supervised keyword extraction method, it requires labeled examples from which to build a model. Therefore, testing with Maui took place only after all manual tags had been collected from the volunteers. The platform for crowd contribution of tags, Pybossa, is detailed in Section 2.2.1, and the method of consolidating the tags into a final gold standard set is detailed in Section 2.2.2.

Each document in a `.txt` file and its corresponding gold standard tags in `.key` format were given the same name, as this is what Maui expects; thus, a document named `ba1.txt` represents the first document in the Breast Augmentation topic, and its corresponding gold standard tags are contained in `ba1.key`, which is just a text file with a `.key` extension. Unfortunately, it was discovered at this time that 8 documents in particular across the data set had corresponding `.key` files that were empty; that is, consolidating the tag sets using the method in Section 2.2.2 resulted in an empty tag set for those documents. No keyword had been voted for by more than one volunteer. This seemed to occur for documents that were extremely short, or documents that had only an entry title and no review body. For example, the content of `ls11.txt` is "Post Op pictures." This was the entry title of the review, and the review body had no content, except for some pictures of the Liposuction patient's post-operation state. The corresponding `ls11.key` file contained no tags at all; upon inspection of the tags contributed by the individual volunteers, the reason was clear: volunteer #1 tagged ["post op pictures"], volunteer #4 tagged ["post op", "pictures"], and volunteer #6 tagged ["post-op"], while the fourth and fifth volunteers tagged nothing at all. No two volunteers agreed on a single tag; therefore, no tag was admitted into the final set. The other seven documents had similar situations. Since this occurred in only 8 documents of

210, it was decided not to do any special handling of these cases, and instead just remove them entirely from the training and test sets. This resulted in a training set of 202 labeled documents.

To evaluate the accuracy of a supervised machine learning model, it is not feasible to simply calculate precision and recall against the gold standard as one would do with unsupervised methods. The reason is that the results of evaluation would be biased, because the model has been trained on the very labeled examples that would be used for evaluating the model's performance. Instead, some methods of evaluating a supervised learning model include 10-fold cross-validation, leave-one-out, and random sampling. In 10-fold cross-validation, the document set is divided randomly into ten equal, disjoint sets, and for each one of the ten sets, it is used for testing, after the remaining 9 sets have been used for training. Thus training and testing is repeated ten times, such that each of the ten sets has been used for testing once. This is the method we use for evaluating Maui's performance.

10-fold cross-validation. For training, the `MauiModelBuilder` expects the `.txt` files and their corresponding `.key` files to be in the `data/automatic_tagging/train/` directory. It matches each text document to its labels, calculates feature values for the features described earlier, and builds a model which is saved for later. After training, the model is reused by the `MauiTopicExtractor` to make predictions on unseen data, which is expected to be in the `data/automatic_tagging/test/` directory. For 10-fold cross-validation, the goal is to validate the performance of the model, so it is necessary to have documents *and* labels in `data/automatic_tagging/test/` directory as well so that the keywords predicted by the model can immediately be compared against the `.key` labels and precision, recall, and F-measure can be immediately calculated. On the other hand, if only documents are included without their `.key` labels, then the purpose is not to evaluate the performance of the model, but to apply the model on data needing to be analyzed and utilized.

A Python script was written that divides the 202 documents into 10 roughly equal parts. For each of the 10 runs, one tenth is moved to the test directory, while the rest is moved to the train directory for training. The `MauiModelBuilder` is called from the script, creating and saving a new model, and then `MauiTopicExtractor` is called which utilizes the model to make predictions on the documents in the test directory, and compare its output to the labels already present in the test directory. `MauiTopicExtractor` requires that a number of keywords to be extracted is specified, with the `-n` flag when calling the object; in this experiment, we compare the results obtained when different numbers are specified. This process is repeated for each of the 10 folds, noting that a new model is made for each run. The calculation of precision, recall, and F-Measure is handled by the `MauiTopicExtractor` and saved to file for averaging and further evaluation.

3.4 Baseline with TF-IDF

A baseline score was calculated so that performances of the other automatic keyword extraction systems have a point of reference. TF-IDF, or term frequency and inverse document frequency, indicates how important a word is to a document in a collection [28]. It has historically been used for text mining, keyword extraction, document classification, and other tasks. It succeeds on the assumption that words that are important to a particular document will appear frequently, and also that they will appear infrequently in other documents of the collection. TF-IDF was chosen to be the baseline because it is often selected as a baseline for its simplicity in keyword extraction and topic indexing.

TextBlob[12] is a Python library used for text processing, with the well-known NLTK[13] library embedded within it. It features part-of-speech tagging, rudimentary sentiment analysis, tokenization, and other tools [29]. Once a piece of text, in a string, is made into a TextBlob object, a collection of text analysis functions are available, like `.tags`, which tags the text with parts of speech, and `.sentiment`, which returns the sentiment of the text. We use the functions `.words` and `.count(word)`, which tokenizes all words of the text, and then counts the occurrences of the `word` in the tokenized text. This gives the term frequency of a word. Concurrently all documents in the collection are checked for the presence of this word; the total number of documents in the collection is divided by the number of documents that contain the word. This gives the inverse document frequency of the word. Term frequency and inverse document frequency are combined by multiplication.

$$tf(word, document) = \frac{wordFrequency}{totalWordsInDocument}$$

$$idf(word, collection) = \log\left(\frac{lengthCollection}{numDocumentsContainingWord}\right)$$

$$tfidf(word, document, collection) = tf(word, document) * idf(word, collection)$$

[12] https://textblob.readthedocs.io/en/latest/index.html
[13] http://www.nltk.org/

A higher TF-IDF score indicates higher significance of that word, and a lower TF-IDF score indicates lower significance. TF-IDF score for 1 word of a document are meaningful when compared with other TF-IDF scores for other words of the same document, but not so when compared with TF-IDF scores of other documents in the collection. Therefore, it is impossible to set a threshold score above which words are accepted as keywords, and below which words are not accepted. Instead, we decided to use the top 10 highest scoring words as keywords, as this number of keywords was shown to be most effective when testing with Maui.

4 Chapter 4 Evaluation and Results

This section discusses the performance of AlchemyLanguage, RAKE, and Maui when each is applied to the data set of RealSelf content and compared with the gold standard created by volunteer staff. Furthermore, a baseline score was calculated by the common TF-IDF method, so that the results could be compared to the simplest and most traditional measure. Precision, and recall were calculated by traditional methods:

$$Precision = \frac{numberCorrect}{numberExtracted}$$

$$Recall = \frac{numberCorrect}{totalExtracted}$$

Since 8 of the 210 documents were without labels, that is, no keywords were agreed on by at least two volunteers, the data set consisted of 202 documents. After each document is processed by a keyword extraction system, precision and recall scores are calculated for each document, and the average precision and average recall are used to calculate the F-Measure:

$$F = \frac{2 * avgPrecision * avgRecall}{avgPrecision + avgRecall}$$

4.1 Modified evaluation for partial matches

This method would reward automatically extracted keywords that exactly matched a keyword in the gold standard. However, it would penalize cases where the correct keyword was extracted, but with a different degree of specificity. For example, if a keyword in the gold standard was “implants”, and the system extracted “high profile implants” as a keyword, the system would not be rewarded for being partially correct. This occurs only when the keyword consists of more than one word; otherwise, the degree of specificity would not be altered. Therefore, a modification was made to the evaluation method to account for partial correctness. Where *numberCorrect* was previously incremented by whole numbers, the modification allows *numberCorrect* to be incremented by fractions if the keyword has a partial match. Figure 7 shows the designed algorithm in pseudo code.

```
for keyword in auto_tags
  if keyword in gold_tags
    increment numberCorrect by 1
  else
    if keyword consists of more than one word
      split keyword into individual words
        for each individual word
          if individual word in gold_tags
            increment numberCorrect by 1/length of keyword
repeat for keywords in gold_tags
```

Figure 7: Algorithm for evaluation including partial matches

If an automatically extracted keyword consists of more than one word, instead of simply checking if it completely matches any gold standard keyword, also check if its individual words match any gold standard keyword, and if so, increment the score by the proportion of the match in relation to the entire keyword. The same is then done in reverse, checking if the gold standard multi-word keywords contain individual words that match an automatically extracted keyword. Note that in reverse, gold standard keywords should not be matched with automatic keywords that have already been partially matched. This theoretically increases the *numberCorrect* score at least marginally; from there, precision, recall, and F-measure can be calculated as before.

Each automatic extraction system was evaluated with the base evaluation method of precision and recall, and also with the modified method for comparison.

4.2 Baseline results

The baseline score gives an indication of base performance that automatic keyword extraction systems should strive to beat. The results of the TF-IDF baseline are in Table 3, with both base evaluation and modified evaluation.

	Average precision	**Average recall**	**F-Measure**
Base evaluation	24.9	24.77	24.83
Modified evaluation	36.57	37.45	37.00

Table 3: Performance of TF-IDF as baseline

Note that this TF-IDF calculation is essentially the bare algorithm, without any adjustments made to improve the score. Researchers have long experimented with adding information to the calculation to improve results, such as part of speech, chunks, and stopwords. The bare algorithm was used because the objective of the experiment is to find the best performing system of Alchemy Language, RAKE, and Maui, not to find the best performing variation of TF-IDF. If the keyword extraction systems can perform better than the baseline, then they have exceeded the base expectation and can be considered for future usage.

4.3 Results with Alchemy Language

The Alchemy Language API was the only automatic tagging system used in this experiment that acted as a "black box", as the code is not publicly provided and its inner workings are not known. Further, it is a paid commercial service intended for use by companies like RealSelf, so it should presumably yield the highest performance of all. Nonetheless, it was interesting to see how well this application, using similar technology to IBM Watson's computer [30], performed against more transparent and less established keyword extraction systems.

	Average precision	**Average recall**	**F-Measure**
Base evaluation	40.87	28.48	33.57
Modified evaluation	50.27	34.44	40.88

Table 4: Performance of AlchemyLanguage

Alchemy Language achieved an F-Measure of 33.57 when evaluated with the base method, and an F-Measure of 40.88 when adjusted for partial matches. Precision is remarkably high, reaching over 50 when accounting for partial matches, meaning that on average half of its predicted keywords are correct or make up a correct keyword. Its precision is quite higher than its recall in both cases of evaluation; the system likely overlooks many candidates that are keywords, though the 10 point increase in precision when evaluating with the modified method indicates that many of its extracted keywords are almost correct, just at a different degree of specificity.

4.4 Results with RAKE

Many adjustments were made to the original RAKE implementation in Python, details of which can be found in Section 3.3.2. The stoplist was continually modified in a loop, checking the performance before and after additions and subtractions to the stoplist, in order to see which collection of stopwords yielded the best performance. Further additions to the stoplist that improved the performance included "entire", "couple", and "time", among several others. The best performance after modification of the stoplist is shown in Table 5.

	Average precision	Average recall	F-Measure
Base evaluation	27.12	17.38	21.18
Modified evaluation	39.15	25.07	30.56

Table 5: Performance of RAKE

The large difference between the two evaluation methods indicates, as with Alchemy Language, that RAKE often extracts the correct keyword but at the wrong degree of specificity, or with more or fewer words than it should have.

Because RAKE heavily favors multiword keyphrases rather than singletons, the vast majority of its extracted keywords consist of two words. Of these, many are formed of an adverb, like "absolutely", followed by a significant adjective or verb. Gold standard tags, on the other hand, rarely include modifiers like adverbs and adjectives, but rather consist of the single significant word. A stopword was experimented with to address this: a pattern `\w{3,}ly` which matches any three or more letters followed by –ly, intended to remove all adverbs ending in –ly. As expected, the precision went up, by 0.08 points to 39.23; however, recall decreased by 0.89 points to 24.18. This was probably due to the fact that without the modifying adverb, the singleton word was not strong enough to be extracted by itself. Since F-Measure decreased by 0.66 points to 29.90, this change was not carried forward; nonetheless, it is interesting to see that removing modifiers, intending to streamline the extracted keywords, in fact hurts performance.

4.5 Results with Maui

Maui's algorithm intends to match the tags that have been assigned by at least two volunteers for each document. We call the Maui Topic Extractor, specifying 20 as the number of keywords to extract from each document. After performing 10-fold cross-validation and obtaining precision, recall, and F-Measure values for each fold, we calculate the average scores across each of the ten folds. This is repeated once for extracting 15 keywords per document, once for extracting 10 keywords per document, and once more for extracting 5 keywords per document to arrive at the results shown in Table 6.

# of keywords extracted	Average precision	Average recall	Average F-Measure
5	**67.66**	37.31	47.97
10	56.11	52.91	**54.32**
15	44.21	62.12	51.34
20	37.09	**63.86**	46.5

Table 6: Performance of Maui when extracting 5, 10, 15, and 20 keywords per document

Each time, the F-Measure closely approaches or exceeds Medelyan's [3] reported best F-measure of 47.10 when utilizing all features to train a bagged decision trees model. It is reasonable that precision decreases as the number of keywords extracted increases: if Maui extracts too many keywords, more than there are gold standard tags, the precision will be low since the number of false positive keywords is high. Conversely, as the number of keywords extracted increases, recall also increases, because the algorithm extracts more words, and thus will extract more correct words (though it will also extract more incorrect words as well, resulting in low precision). It appears that extracting 10 keywords per document yields the highest F-Measure, and yields relatively balanced precision and recall.

Precision and recall are highly dependent on the number of keywords extracted, as more keywords will likely improve recall but hurt precision, and fewer keywords will improve precision but hurt recall. It stands to reason that longer documents will have more assigned keywords while shorter documents will have fewer assigned keywords. The disadvantage of Maui is that there is no way, at least on the surface, to extract a number of keywords based on each document's length. Automatic keyword extraction algorithms like RAKE and TF-IDF extract only as many keywords as fit the algorithm, and no less; but Maui will extract exactly as many keywords as is specified in the function call, and for each document regardless of its length. If there were a way to make dynamic the number of extracted keywords, the performance scores would surely be greatly improved, though that experiment is outside the scope of this one.

4.6 Discussion

Combining the results of each of the three keyword extraction systems, it is easy to see their varying performance. Table 6 compares the three systems' best performances with each other.

	Average precision	**Average recall**	**F-Measure**
Baseline TF-IDF	36.57	37.45	37.00
Alchemy Language	50.27	34.44	40.88
RAKE	39.15	25.07	30.56
Maui	**56.11**	**52.91**	**54.32**

Table 7: Best performances of Alchemy Language, RAKE, and Maui

Maui exceeds the baseline set by TF-IDF by far, in precision, recall, and F-Measure, when it is set to extract 10 keywords per document. It in fact exceeds Medelyan's [3] reported highest F-Measure of 47.10 with Maui. This means that Maui matches over half of all tags on which at least two human taggers have agreed. Alchemy Language outperforms RAKE. Their precision and recall scores are comparable.

Results show that although the baseline achieves a decent F-Measure of 37.00, Maui outshines the others with a remarkable F-Measure of 54.32, when it is set to extract 10 keywords per document. RAKE performs the worst of the three systems under exploration, with an F-Measure of 30.56. It is the only keyword extraction system that performs worse than the baseline. All three systems have better precision than recall, averaging a difference of 10 points between precision and recall scores. TF-IDF is markedly different in this aspect, as its average precision and recall are very close only differing by 1 point.

It is understandable that Maui performs the best, because as a supervised learning model it trains directly on the gold standard tags in order to extract ones with similar features. RAKE began as a skeleton of a keyword extraction system with barely an F-Measure above 10, but after many layers of modification to the stoplist and other aspects of the algorithm, blossomed into a system that is moderately suited to the language of internet reviews.

5 Chapter 5 Conclusions and Future Work

In this work I have explored the variance in performance of three different automatic keyword extraction systems: Alchemy Language, RAKE, and Maui. They differ greatly in their origin, and in their design: Alchemy Language is a commercial API, whose natural language processing algorithms are unknown to the public; RAKE is a simple yet powerful algorithm that extracts keywords by essentially removing all stopwords and returning what is left; and Maui is a supervised machine learning method that learns the features of keywords and uses them to predict for new documents. With a subset of reviews data from RealSelf, the three automatic keyword extraction systems are applied, using TF-IDF as a baseline. Evaluation of the machine tags is done by creating a gold standard, curated from the tags of a folksonomy of volunteer staff; any tag that has been agreed on by at least two volunteers is part of the gold standard. Traditional precision and recall values are calculated for TF-IDF, serving as the baseline, and then calculated for Alchemy Language and RAKE, and average values over documents are taken for calculating F-Measures. A novel, modified method of evaluation is also introduced, which rewards systems for partially matching a gold standard tag. Maui, being a supervised machine learning model, requires a different method of evaluation, in this case, 10-fold cross validation. The model is trained ten times and tested on ten different portions of the data set, and precision and recall values are calculated for each run and averaged to get F-Measures.

Results show that although Alchemy Language does decently well, Maui performs the best of the three when it is set to extract 10 keywords per document. RAKE performs the worst of the three systems, over 6 points worse than the baseline score. All three systems have better precision than recall, averaging a difference of 10 points between precision and recall scores, except for TF-IDF which gives comparable precision and recall. Generally speaking, Maui, Alchemy Language, and even basic TF-IDF perform passably well in the domain of internet reviews. RAKE is likely not a first choice for keyword extraction for this domain.

One conclusion to be drawn is that although Alchemy Language may seem like a comprehensive package of natural language processing tools, its keyword extraction algorithm is far from being the most performant, and the reputation it carries may not be worth its price tag, at least in this domain of text. Nonetheless, with accuracy of about one-third, it may be more suitable to companies for whom moderate accuracy and ease of implementation are desirable.

5.1 Future work

The more gold standard data we have, the more reliable a keyword extraction system can be. Many of the modifications made to RAKE and Maui were highly dependent on how the changes affected their performance when tested on the labeled data; however, 202 documents cannot be considered a large corpus. With more time and resources, a larger, more accurate, and more comprehensive data set could be created, and used to better tune the automatic keyword extraction systems.

Due to time constraints and the complexity of Maui's design, it was not feasible to dig into Maui's implementation to modify its various functions and features and see how performance was affected. For example, though Medelyan [3] spends several pages describing the different keyword features that influence classification and how much they each impact performance, it is conceivable that the impact of features would be different when tested on a domain such as internet reviews about cosmetic treatments. Further, bagged decision trees is shown to be the most effective in Medelyan's work, but other models may perform even better for RealSelf's content. Therefore, the logical next move is to investigate the facets of Maui and tune it so that performance is further improved.

The objective of this work was to explore different methods of keyword extraction, and ultimately decide on one, or several, to be used for all past and future RealSelf content. As Maui has proven to be the most performant by far, it could possibly go on to be the primary system for automatic keyword extraction at RealSelf. In that vein, a more enhanced version of Maui called Maui Pro[14] has been developed by the company Entopix[15], whose founder is Olena Medelyan, the very producer of Maui. Maui Pro is presumably suited to companies who frequently perform data and text analysis, and who require a paid, polished software along with technical support. It would also be an interesting venture to test out this enhanced version at the enterprise level and compare it with the tweaked open-source Java implementation.

[14] http://entopix.com/maui/
[15] http://entopix.com/

6 References

[1] S. Beliga, M. Ana and S. Martinčić-Ipšić, "An Overview of Graph-Based Keyword Extraction Methods and Approaches," *Journal of Information and Organizational Sciences.,* vol. 39, no. 1, pp. 1-20, 2015.

[2] L. S. Bonura, The Art of Indexing, Wiley, 1994.

[3] O. Medelyan, "Human-competitive automatic topic-indexing," 2009.

[4] O. Medelyan, E. Frank and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Singapore, 2009.

[5] C. Moss, "'RealSelf' Is Like A Social Network For People Who Want Cosmetic Surgery," 10 February 2014. [Online]. Available: http://www.businessinsider.com/realself-for-cosmetic-surgery-2014-2?IR=T.

[6] D. P. Anderson, "Bossa Overview," 2007. [Online]. Available: https://boinc.berkeley.edu/trac/wiki/BossaOverview.

[7] M. Reimer, "Pybossa Overview".

[8] R. Kelly, "PyEnchant Tutorial," 2016. [Online]. Available: http://pythonhosted.org/pyenchant/tutorial.html.

[9] AlchemyAPI, Inc, "Keyword Extraction API," [Online]. Available: http://www.alchemyapi.com/products/alchemylanguage/keyword-extraction.

[10] J. Turian, "Using AlchemyAPI for Enterprise-Grade Text Analysis," 2013.

[11] AlchemyAPI, "9 Things You Should Know About AlchemyAPI," 2013.

[12] S. Rose, D. Engel, N. Cramer and W. Cowley, "Automatic keyword extraction from individual documents," in *Text Mining: Applications and Theory*, West Sussex, Wiley, 2010, pp. 1-20.

[13] A. Medelyan, "NLP keyword extraction tutorial with RAKE and Maui," 1 January 2016. [Online]. Available: https://www.airpair.com/nlp/keyword-extraction-tutorial. [Accessed 26 March 2016].

[14] C. Buckley, "Implementation of the SMART Information Retrieval System," Cornell University, Ithaca, 1985.

[15] C. Silva and B. Ribeiro, "Background on Text Classification," in *Inductive Inference for Large Scale Text Classification: Kernel Approaches and Techniques*, Berlin Heidelberg, Springer Science & Business Media, 2009, p. 9.

[16] J. Leveling, D. Ganguly and G. J. Jones, "Term Conflation and Blind Relevance Feedback for Information Retrieval on Indian Languages," in *Multilingual Information Access in South Asian Languages*, Berlin Heidelberg, Springer, 2013, pp. 295-309.

[17] Lextek, *Onix Full Text Indexing and Retrieval Toolkit,* 2003.

[18] I. S. Dhillon and D. M. Modha, "Concept Decompositions for Large Sparse Text Data using Clustering," *Machine Learning,* vol. 42, no. 1, pp. 143-175, 2001.

[19] G. Blake and R. W. Bly, The Elements of Technical Writing, New York: Macmillan Publishers, 1993, p. 48.

[20] K. Yin, "Em Dashes and Ellipses: Closed or Spaced Out?," 10 May 2011. [Online]. Available: http://www.apvschicago.com/2011/05/em-dashes-and-ellipses-closed-or-spaced.html.

[21] K. Berica, "Regex Boundaries and Delimiters - Standard and Advanced," 2 March 2015. [Online]. Available: http://www.rexegg.com/regex-boundaries.html.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations,* vol. 11, no. 1, 2009.

[23] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin and C. G. Nevill-Manning, "Domain-Specific Keyphrase Extraction," in *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

[24] J. B. Lovins, "Development of a stemming algorithm," *Mechanical translation and computational linguistics,* vol. 11, pp. 22-31, 1968.

[25] B. Lott, "Survey of Keyword Extraction Techniques," 2012.

[26] D. Milne, "An open-source toolkit for mining Wikipedia," in *New Zealand Computer Science Research Student Conference*, Auckland, 2009.

[27] D. Milne and I. H. Witten, "An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links," in *ACM Conference on Information and Knowledge Management*, Napa Valley, CA, 2008.

[28] A. Rajaraman and J. Ullman, "Data Mining," in *Mining of Massive Datasets*, Cambridge, Cambridge University Press, 2011, pp. 1-17.

[29] S. Loria, "TextBlob: Simplified Text Processing," 2016. [Online]. Available: http://textblob.readthedocs.io/en/dev/.

[30] A. Williams, "AlchemyAPI Raises $2 Million For Neural Net Analysis Tech, On Par With IBM Watson, Google," 7 February 2013. [Online]. Available: http://techcrunch.com/2013/02/07/alchemy-api-raises-2-million-for-neural-net-analysis-tech-on-par-with-ibm-watson-google/.

[31] I. H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, San Francisco, CA: Elsevier, 2005.

7 Table of Figures

8 Table of Tables