2014

AUTOMATIC LABELING OF RSS ARTICLES USING ONLINE LATENT DIRICHLET ALLOCATION

Zhe Lu
Michigan Technological University

AUTOMATIC LABELING OF RSS ARTICLES USING ONLINE LATENT DIRICHLET ALLOCATION

By

Zhe Lu

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Computer Science

MICHIGAN TECHNOLOGICAL UNIVERSITY

2014

This report has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Computer Science.

Department of Computer Science

Report Advisor: *Laura Brown*

Committee Member: *Timothy Havens*

Committee Member: *Min Wang*

Department Chair: *Charles Wallace*

Contents

List of Figures

List of Tables

Abstract

The amount of information contained within the Internet has exploded in recent decades. As more and more news, blogs, and many other kinds of articles that are published on the Internet, categorization of articles and documents are increasingly desired. Among the approaches to categorize articles, labeling is one of the most common method; it provides a relatively intuitive and effective way to separate articles into different categories. However, manual labeling is limited by its efficiency, even thought the labels selected manually have relatively high quality. This report explores the topic modeling approach of Online Latent Dirichlet Allocation (Online-LDA). Additionally, a method to automatically label articles with their latent topics by combining the Online-LDA posterior with a probabilistic automatic labeling algorithm is implemented. The goal of this report is to examine the accuracy of the labels generated automatically by a topic model and probabilistic relevance algorithm for a set of real-world, dynamically updated articles from an online Rich Site Summary (RSS) service.

Chapter 1

Introduction

The modern Internet contains billions of piece of information, and it is continuously updated [2]. A challenge for today and the future is to separate information into proper categories, which benefits both the organization that maintains it and the users who consume it. Among different categorizing approaches, labeling is one of the most commonly used methods: assigning specific words or phrases related to the content and use them as keywords to group content under the same label.

In this report, we focus on news article labeling, specifically Rich Site Summary (RSS) [3] news article labeling. Originally, the RSS news articles sometimes have manually assigned words as labels to indicate the main topics of the given articles. The labels have high quality because it is usually the author or the editor of the article who assigns the labels, which can

be considered as the reflection of the topics within the articles. However, the efficiency of manual labeling is low and the coverage of the labels is limited. The task of this report is to find a way to automatically label the news articles based on the context of a set of news articles and the articles' content.

To automatically label an article, we first need to generate a list of potential labels that catch the main topic of the articles. To do this, we can make use of the probabilistic topic modeling approaches to extract the topics within the set of articles. One of the most successful topic modeling approaches is Latent Dirichlet Allocation (LDA) by Blei, Ng and Jordan [1]. LDA is a generative probabilistic method that models a collection of documents as a mixture over a latent set of topics, and each topic is modeled as a mixture of the observed representation of elements of the collections. In this case, we can consider the set of articles as the collection, the words that are used in the articles are the representation and the latent topics are the mixture of words that can summarize the idea of the articles. With LDA, we can generate a list of topics containing words for a set of articles, and use these words as our potential labels.

Once we have the set of label candidates, the second step is to find the most accurate words that represent the main idea of the articles. To achieve this goal, we use a relevance scoring approach based on KL-divergence [4] as the post-processing step of the topic modeling and rank the candidates with their scores [5]. The smaller the divergence, the higher the score, and more "important" the candidate is to the article. Within the ranked candidates, we can

then select a subset of top candidates and use them as our final labels to the articles.

By combining these two methods, we can generate a list of labels for a specific article, and use them to categorize the article as well as summarizing the content. Moreover, since the articles are constantly and dynamically updated and can be seen as an infinite set of documents, we use Online-LDA [6], an extension of LDA, that fits the online fashion by iterating through incoming documents to handle large scale data.

The rest of the paper is organized as follow. In Chapter 2, we describe the background of the sources, techniques and approaches that we use in this report. In Chapter 3, we compare the Online-LDA with original LDA and hierarchical-LDA and discuss the advantages of Online-LDA over the other two. Chapter 4 describes the modifications we make in our report to the original Automatic Labeling algorithm. In Chapter 5, we propose the process of the online automatic labeling. Chapter 6 presents the evaluation we conduct to examine the accuracy of the labels generated by our web application. The conclusions and future work are in Chapter 7.

Chapter 2

Background

This chapter introduces the concept of the technologies and algorithms used in this report. For technologies, Rich Site Summary and the RSS service Feedly are briefly introduced. For algorithms, the original Latent Dirichlet Allocation, its extensions hierarchical-LDA and Online-LDA, and the automatic labeling algorithm are discussed.

2.1 Rich Site Summary

Rich Site Summary (RSS, also dubbed Really Simple Syndication) is an Internet service that uses a family of standard web feed formats [3] to output frequently updated content, such as news articles, blogs and podcasts. The output of RSS often includes the meta-data

of the content: RSS source, content title, posted time, author, and full or summarized text.

Users of RSS usually "subscribe" RSS documents ("feeds") from different websites, and then use an aggregator service to manage those feeds. Once subscribed, the aggregator will constantly update when the website has new content, so that users don't have to manually check the website for updates.

By using the standard feed format, an RSS feed can be easily parsed through programs, and therefore there are a large number of RSS aggregator applications that are available for users. The most famous one is Google Reader, which was shutdown due to Google's strategy changes. After Google Reader's shutting down, Feedly has become a popular RSS aggregator.

2.1.1 Feedly

Similar to Google Reader, Feedly is a RSS aggregator application that contains subscription management, RSS content parsing and browsing, etc [7]. Feedly also extends the functionalities of RSS aggregator to a productivity tool, by adding links to different productivity applications such at Evernote.

Additionally, Feedly also provides a set of public APIs which allows third party developers access to core Feedly features. Thus, third party applications can make use of Feedly's

retrieval and management of users' subscriptions [7]. This report will use Feedly's public API to retrieve news articles from users' accounts, and then use the articles to perform Online LDA automatic labeling.

2.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (LDA) [1] is a generative probabilistic topic model for a corpus. Intuitively, LDA assumes that the documents in the corpus are represented as random mixtures over a specific set of topics, which can be called latent topics. To be more specific, the composition of a document can be divided into the following steps: the author has a set of topics in mind; when he or she is writing the document, words will be chosen based on the topics with a certain probability. Therefore, the entire document can be represented as a mixture of multiple topics. According to this assumption, given a document, the latent topics within can be extracted from the vocabulary used in the document, which is the core task of LDA.

Figure 2.1 shows the graphical model of LDA. Let N be the number of words in a document, the topic mixture be θ, the topics vector be z, and the words vector be w, the generative process of LDA for the each document in a corpus is shown in Figure 2.2.

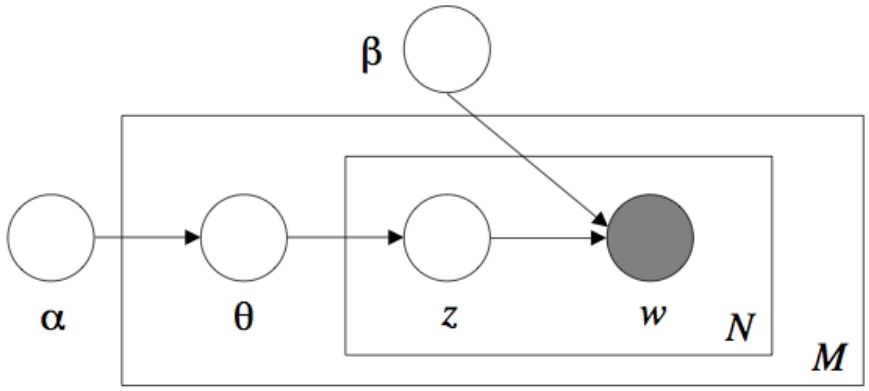

Figure 2.1: Graphical model representation of LDA.[1]

Algorithm 1: Generative process of LDA

1. Choose $N \sim$ Poisson(ξ);
2. Choose $\theta \sim$ Dir(α);

for *each word w in document d* **do**

3. Choose a topic $z_n \sim$ Multinomial(θ);
4. Choose a word w_n from $p(w_n|z_n,\beta)$;

end

Figure 2.2: Generative process of LDA

Based on the process above, the joint distribution of θ, z, and w can be written as:

$$p(\theta,\mathbf{z},\mathbf{w}|\alpha,\beta) = \prod_{d=1}^{\mathscr{D}} p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n,\beta), \tag{2.1}$$

where $\mathscr{D}$ is the corpus containing all documents, $\mathbf{w}$ is the vector of all words in the corpus, N is the total words in the corpus and α and β are parameters to be inferred.

The inference of parameters requires the use of estimation, as the distribution $p(\mathbf{w}|\alpha,\beta)$ is intractable in general. Two approaches are widely used for the inference of the parameters:

variational inference [8] and Markov Chain Monte Carlo methods.[9].

LDA is a good algorithm for providing candidate labels, but the Internet is a giant data set and RSS articles are frequently updated; hence, the original LDA is not suitable for labeling RSS articles.

2.3 Hierarchical Latent Dirichlet Allocation

Hierarchical Latent Dirichlet Allocation (hLDA) [10] is an extension of LDA that addresses learning topic hierarchies from data using a nested Chinese restaurant process.

A nested Chinese restaurant process can be described as a hierarchy of Chinese restaurants, where each of the restaurants has infinite number of tables, and each table refers to another restaurant. One restaurant is assigned as the root restaurant, where visitors visit it first. Assuming that there are m visitors coming to the city and are planning to visit the restaurants, every day they visit one restaurant in the hierarchy starting from the root restaurant, and the next day they will choose the restaurant referred from the table they chose previously. At the end of the trip, each of the visitors will form a path in the hierarchy, and together the visitors form a subtree within the infinite branch tree of restaurants.

The Chinese restaurant process is formed by two equations that describes a scenario of the

seat choosing process of an imagined Chinese restaurant:

$$p(\text{occupied table } i \mid \text{previous customers}) = \frac{m_i}{\gamma + m - 1}, \tag{2.2}$$

$$p(\text{next unoccupied table} \mid \text{previous customers}) = \frac{\gamma}{\gamma + m - 1}, \tag{2.3}$$

where m is the number of customers at table i and γ is a scalar parameter of the process. The two equations describes a scenario of choosing a seat in a restaurant with an infinite number of tables, each of which has an infinite number of seats. A sequence of customers arrive at the restaurant, the first customer sits at the first table, and for the rest of the customers, they can choose to sit with the previous customer, or sit at an empty table that is not occupied by any other customers. The scalar parameter γ represents the probability of a customer choosing a new table versus choosing an occupied table.

Nested Chinese restaurant processes (nCRP) can be used in topic modeling for recovering levels of abstractions of a topic. Hierarchical LDA is such topic model that builds on nCRP, which arranges the topics into a tree structure, with more general topics appear in higher levels (near root) and specialized topics in lower level.

Because of the tree structure of hLDA topics models, it gives a good set of label candidates as we can consider different levels of generalization of the topics. However, because hLDA still requires a full pass of the corpus, it is not suitable for an online system.

2.4 Online-Latent Dirichlet Allocation

Online Latent Dirichlet Allocation (Online-LDA) [6] is an extension of the original LDA algorithm, which implements the method in an online fashion for handling large scale data analysis.

The original LDA method can be seen as a probabilistic factorization of the observed word counts matrix, each of element of the matrix indicates the number of appearances for a specific word in a document. The factorization outputs a topic weights matrix containing the topics' contributions to the documents, and a dictionary of topics. The Online-LDA method can thus be seen as an extension that uses an online matrix factorization approach.

Online-LDA uses a variational Bayes approach to perform parameter estimation [1]. Originally, the variational Bayes approach for LDA is to maximize the Evidence Lower BOund (ELBO)

$$\log p(\omega|\alpha,\eta) \geq \mathscr{L}(w,\phi,\gamma,\lambda) \triangleq \mathbb{E}_q[\log p(w,z,\theta,\beta|\alpha,\eta)] - \mathbb{E}_q[\log q(z,\theta,\beta]. \quad (2.4)$$

The posterior over the per-word topic assignment z is parameterized by ϕ, the posterior over per-document topic weights θ is parameterized by γ and the posterior over topics β is parameterized by λ.

The equation can be factorized to

$$\begin{aligned}\mathscr{L}(w,\phi,\gamma,\lambda) = \Sigma_d\{&\mathbb{E}_q[\log p(w_d|\theta_d,z_d,\beta)] + \mathbb{E}_q[\log p(z_d|\theta_d)] \\ &- \mathbb{E}_q[\log q(z_d)] + \mathbb{E}_q[\log p(\theta_d|\alpha)] \\ &- \mathbb{E}_q[\log q(\theta_d)] + \mathbb{E}_q[\log p(\beta|\eta)] \\ &- \mathbb{E}_q[\log q(\beta)]/|\mathscr{D}|\}.\end{aligned} \tag{2.5}$$

Also, $\mathscr{L}$ is optimized using coordinate ascent over the variational parameters ϕ, γ, λ, [6]

$$\phi_{dwk} \propto \exp\{\mathbb{E}_q[\log \theta_{dk}] + \mathbb{E}_q[\log \beta_{kw}]\},$$

$$\gamma_{dk} = \alpha + \Sigma_w n_{dw}\phi_{dwk},$$

$$\lambda_{kw} = \eta + \Sigma_d n_{dw}\phi_{dwk},$$

where k is the k-th topic, d is the d document in the corpus, and w is the w-th word in the vocabulary set.

The variational Bayes inference for LDA uses a Expectation Minimization (EM) approach to estimate these three parameters, which requires a full pass of the corpus, and thus limits its use when there are new documents coming constantly.

The basic idea of Online-LDA is to use mini-batches instead of the entire corpus to estimate

Algorithm 2: Online-LDA procedure

for *Every mini-batch t* **do**

1. E-step: find a locally optimized parameter ϕ_t and γ_t with fixed λ
2. M-step: compute $\widetilde{\lambda}$ using ϕ_t and the observed word count n_t. And then update λ with $\widetilde{\lambda}$.

end

Figure 2.3: Online-LDA procedure.

the parameters. The mini-batches are used for fitting λ. The algorithm constantly reads mini-batches, and updates λ after each iteration.

The procedure of Online-LDA is shown in Figure 2.3. Online-LDA is claimed to have faster convergence speed, and because it is capability of handling large scale document analysis and frequent updating documents, this report will use this method to generate candidates for article labeling.

2.5 Automatic labeling of Multinomial Topic Models

Automatic labeling of multinomial topic models is a probabilistic approach to objectively label topics generated from topic models in a statistical fashion. In this method, the automatic labeling problem is seen as an optimization problem where the KL divergence between word distributions is minimized while the mutual information between a label and a topic model is maximized [4].

The approach here is to compute a relevance score for each candidate label, which

is retrieved from a topic model θ, based on a probabilistic method for selecting understandable, semantically relevant and discriminative words.

In [5], the authors mention two relevance scoring methods: zero-order relevance and first-order relevance. The zero-order relevance is described as

$$Score = \log \frac{p(l|\theta)}{p(l)} = \sum_{0 \leq i \leq m} \log \frac{p(u_i|\theta)}{p(u_i)}, \tag{2.6}$$

where l is a candidate phrase $l = u_0 u_1 \cdots u_m$ and u_i is a word. The basic idea of this method is that the more important words, i.e., high $p(u|\theta)$ a phrase contains, the better it is.

For first-order relevance, the goal is to measure the closeness of two multinomial distribution $p(w|l)$ and $p(w|\theta)$ by using KL divergence. Ideally the best label candidate should have zero KL divergence. However, the distribution $p(w|l)$ is unknown, and therefore needs to be approximated. In [5], a method is described for using $p(w|l,\mathscr{C})$, where $\mathscr{C}$ is a context collection, to replace $p(w|l)$. The scoring can be described as

$$\begin{aligned} Score(l,\theta) &= -KL(\theta||l) \\ &= \Sigma_w p(w|\theta) \log \frac{p(w,l|\mathscr{C})}{p(w|\mathscr{C})p(l|\mathscr{C})} \\ &- KL(\theta||\mathscr{C}) - \Sigma_w p(w|\theta) \log \frac{p(w|l,\mathscr{C})}{p(w|l)}, \end{aligned} \tag{2.7}$$

where $-KL(\theta||l)$ is the KL-divergence of $p(w|\theta)$ and $p(w|l)$. The first component is the expectation of point-wise mutual information between l and the terms in the topic model

θ given the context $\mathscr{C}$. The second component is the KL divergence between the topic and the context collection, which can be ignored since it is identical for all candidates. The third component is the bias of using context collection $\mathscr{C}$, which can be also ignored if both the label candidates and the topic model are generated from the collection.

In this report, our method of labeling news articles is similar but not identical to the method described in [5]; instead of using phrases generated in the articles, we directly use single words as labels and we generate multiple labels for each article.

Chapter 3

LDA, Hierarchical LDA and Online-LDA

This section discusses the differences between the original LDA, its extensions hierarchical LDA and Online-LDA, and the reason we select Online-LDA for automatic labeling.

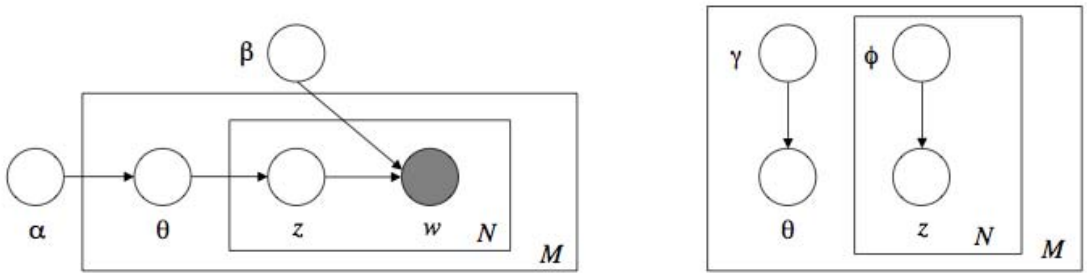

Figure 3.1: Original Graphical model representation of LDA (Left) and simplified graphical model representation of LDA for VB approximation algorithm [1].

3.1 LDA and hierarchical LDA

As mentioned in Chapter 2, the main process of LDA is the inference of the parameters α and β, which can be expressed as the posterior distribution given a document:

$$p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta)}{p(\mathbf{w}|\alpha, \beta)}.$$

However, the distribution is intractable in general; hence, the authors of [1] suggest a variational inference algorithm to approximate the inference.

The basic idea of variational inference used in LDA is to obtain an adjustable lower bound on the log likelihood [8]. Specifically, in LDA, the graphical model is modified, edges of coupling parameters θ, β and the node w are removed to obtain a simplified tractable

family of lower bounds, as shown in Figure 3.1. The family of lower bounds can therefore be presented as an variational distribution

$$q(\theta, \mathbf{z}|\gamma, \phi) = q(\theta|\gamma) \prod_{n=1}^{N} q(z_n|\phi_n). \tag{3.1}$$

Next the parameters γ and ϕ are estimated by an optimization problem

$$(\gamma*, \phi*) = \arg\min_{\gamma,\phi} D(q(\theta, \mathbf{z}|\gamma, \phi)||p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta)), \tag{3.2}$$

where the optimization problem is to minimize the KL-divergence between the variational distribution and the posterior $p(\theta, \mathbf{z}|\mathbf{w}, \alpha, \beta)$.

The estimation of parameter γ and ϕ is per document, and an Expectation-Maximization procedure is used along with the variational inference to approximate the parameter α and β:

1. E step: find the optimized parameter γ and ϕ for each document.

2. M step: use the parameters optimized in E step to maximize the lower bound of the log likelihood

   $$l(\alpha, \beta) = \Sigma_{d=1}^{M} log(\mathbf{w}_d|\alpha, \beta),$$

   until the log likelihood converges, where M is number of documents within the

Algorithm 3: Procedure of hierarchical-LDA

Data: Current states of $\{c_d^{(t)}, z_{n,d}^{(t)}\}$
for *every document in corpus* $\mathscr{D}$ **do**

1. Sample level allocations: Randomly draw $c_d^{(t+1)}$ from

$$p(\mathbf{c}_d|\mathbf{w}, \mathbf{c}_{-d}, \mathbf{z}, \eta, \gamma) \propto p(\mathbf{c}_d|\mathbf{c}_{-d}, \gamma)p(\mathbf{w}_d|\mathbf{c}, \mathbf{w}_{-d}, \mathbf{z}, \eta).$$

2. Sample Path: Randomly draw $z_{n,d}^{t+1}$ for every word from

$$p(z_{d,n}|\mathbf{z}_{-(d,n)}, \mathbf{c}, \mathbf{w}, m, \pi, \eta) \propto p(\mathbf{z}_{d,n}|\mathbf{z}_{d,-n}, m, \pi)p(w_{d,n}|\mathbf{z}, \mathbf{c}, \mathbf{w}_{-(d,n)}, \eta).$$

end

Figure 3.2: Hierarchical-LDA algorithm.

corpus.

Similarly, the extension hierarchical-LDA also uses inference to approximate the parameters. However, the authors of [10] use the MCMC Gibbs Sampling [11, 12] instead of variational inference as the inference algorithm. In hLDA, the target distribution sampled is the per-document path $\mathbf{c}_d$, which is the probability of a document to choose topics (the nodes) from the root of the nested Chinese Restaurant Process, and the per-word level allocations to topics in the paths $z_{d,n}$. The procedure of hLDA is summarized in Figure 3.2, where γ represents the probability of a document selecting the topic (the node), $\mathbf{c}_{-d}$ represents the path vector without c_d, and η represents the expected variance of the underlying topics, η represents the expected variance of the underlying topics, parameters m and π represents the expectation of allocations of words to levels in a document, and $\mathbf{z}_{-(n,d)}$ and $\mathbf{w}_{-(d,n)}$ are the vectors of the level allocation and words without $z_{n,d}$ and $w_{d,n}$. The algorithm is guaranteed to converge after a sufficient number of iterations.

An advantage of hierarchical LDA is the structural topic model generated by the algorithm. Unlike the original LDA, hierarchical LDA generates a tree structure of topic models with different level of generalization. With this algorithm, we can generate a topic model that holds the information of generalization of the words in respect of the topics, and the selection of candidates in our labeling process can be more effective since we have one more condition to look at — the level of generalization of the topics.

However, both LDA and hierarchical-LDA require iterating through the entire corpus in order to complete the inference procedure. Given the online property of the RSS news articles, we do not want to limit the size of articles in our process, and using LDA or hierarchical-LDA would require re-computing the topic model once a new set of articles are fetched from the RSS aggregator.

3.2 Online-LDA

On the other hand, another extension of LDA, Online-LDA is a topic modeling procedure designed for handling large scale data, which in this case is a large number of documents.

As mentioned in Chapter 2, Online-LDA uses variational inference as the inference algorithm, but is different from the original LDA. Online-LDA uses an online variational inference for LDA to approximate the parameters.

Online-LDA also uses an EM procedure; the E-step is similar to the original LDA, which tries to find the local optimal values for γ and ϕ with parameters λ fixed. In the M-step, however, Online-LDA computes $\tilde{\lambda}$ using γ and ϕ obtained from E-step and use it to update λ. $\tilde{\lambda}$ represents the setting of λ optimal to the situation where the entire document set is formed by a single document D times, where D is the number of unique documents available. A weight $\rho_0 \triangleq (\tau_0 + t)^{-\kappa}$ is assigned to each $\tilde{\lambda}$ computed in every iteration.

Instead of passing through the entire corpus of data, Online-LDA uses mini-batches to perform the EM procedure, which follows a multiple observations per update technique in stochastic learning. To be more specific, the update of $\tilde{\lambda}$ can be written as

$$\tilde{\lambda} = \eta + \frac{D}{S} \sum_{s} n_{tsk} \phi_{tskw},$$

where D is the size of the entire corpus, S is the size of mini-batch. The procedure of Online-LDA is shown in Figure 3.3.

In Chapter 1, we described the challenge for this application is that the RSS articles are constantly updates, meaning that we are dealing with an infinite size of input. Therefore even though the hierarchical LDA provides a structural topic model that can help further classify articles with the level of topics, due to the lack of ability to handle data online, Online-LDA fits the requirement of our goal and therefore is selected over the original LDA and hierarchical LDA.

Algorithm 4: Online-LDA algorithm

Define $\rho_0 \triangleq (\tau_0 + t)^{-\kappa}$;
Randomly initialize λ;
while *True* **do**
    Initialize γ_{tk} arbitrarily;
    repeat
        /* Optimize $\mathscr{L}$ over variational parameters ϕ and γ */
        Set $\phi_{twk} \propto \exp\{\ \mathbb{E}_q[log\theta_{tk}] + \mathbb{E}_q[\log \beta_{kw}]\ \}$;
        Set $\gamma_{tk} = \alpha + \Sigma_w \phi_{twk} n_{tw}$;
    until $\frac{1}{K}\Sigma_K |change\ in\ \gamma_{tk}| < 0.00001$;
    /* Compute $\tilde{\lambda}$ and optimize λ */
    Compute $\tilde{\lambda}_{kw} = \eta + D n_{tw} \phi_{twk}$;
    Set $\lambda = (1-\rho_t)\lambda + \rho_t \tilde{\lambda}$;
end

Figure 3.3: Online-LDA algorithm. K: number of topics; D: number of documents.

Chapter 4

Automatic Labeling

In [5], Mei et al. proposed an algorithm that uses the result of topic modeling to generate labels for the topics. We use this algorithm with modifications as the final step of generating labels for news articles.

4.1 Single Word Labels

The first difference we have made compared to [5] is that we only consider single words for the label candidates. In [5], phrases are used as label candidates over single words and sentences, where a single word is too generative to reveal details of a topic, and a sentence is too specific to cover aspects of it. However, as our labeling target is news

articles instead of topics, a phrase sometimes cannot catch the main idea of the article, and we essentially will have multiple words being used as labels that not only summarize the article but also provide some details. Therefore, we choose to use multiple single words as our label candidates.

4.2 Zero Bias

As mentioned in Section 2.5, we use first-order relevance scoring as our scoring method. To approximate the distribution $p(w|l)$, we use the mini-batch of articles in an iteration of Online-LDA as the context collection, which is also the collection that we use for the topic model. Therefore, the bias in equation 2.7 can be ignored as we use the same collection to generate both the labels and the topics.

4.3 Relaxation of Criteria

In this application, we do not consider the high coverage labels criteria [5] proposed for topic labeling. Originally the idea of high coverage labels was to select labels that cover as much semantic information of a topic as possible, as only one label is selected for one topic. In our case, we want to select multiple words as the labels for each articles to cover as many aspects of the articles as possible. Hence, the restriction of finding high coverage

labels is relaxed and the task can be simplified to finding the most useful labels in a topic for a specific article.

We also do not consider the discriminative labels criteria proposed in [5]. The argument described in the original paper is that a label that appears in many topics is not a useful label as it is not discriminative, that is, a good label should have high semantic relevance to the target topic but low relevance to other topic models. In our case, the news articles in different topics can be labeled the same as long as the label is useful in each topic. News articles are time sensitive and therefore it is possible that news articles during a specific time period are talking about the same event, but in different aspects. We want to label these related articles with similar labels, and therefore do not apply the discriminative criteria.

4.4 Additional Rules for Candidate Selection

Another modification introduced in our application is that we only consider words that explicitly appear in the article to be label candidates. The topic model can be represented as the probability of given words to be selected for the specific topic. While the words having higher probability are more important to the topic, they may not be equally important for articles, because some of the words may not appear in the articles. Ideally, to generate labels that are related to the article, we first need to measure the relation between the words and the article, and setup a threshold to select those are closely related to the article, not

necessarily appearing in the article. However at this moment, we only consider the words that have explicitly appeared in the article to simplify the analysis process.

Last, we include the words that explicitly appear in the title of the articles into the candidate set. We also assign a larger weight to the words in the title while computing relevance scores. The intuitive reason is that for most news articles, the title usually includes important words that summarize the article, and therefore should be considered as important to the article and be added into the candidate set.

Chapter 5

Implementation

This section covers the implementation details of the project. The project uses Feedly RSS aggregator service as the source of the news articles, and uses the Feedly public API to fetch articles from our test account. The automatic labeling is programmed using Python and the Google App Engine public API. The service is hosted on Google App Engine as a web application. In Section 5.1, we briefly describe the Feedly public API and our implementation of fetching articles. In Section 5.2, we introduce the Google App Engine (GAE) web application platform and its advantages and limitations. Section 5.5 and Section 5.6 are used to discuss some of the GAE limitations and the workarounds in our implementation.

5.1 Feedly

The main API call used in this project is the Feedly stream API, which creates a stream of articles sorted by time and outputs a certain number of them based on the caller's request. The stream does not have restriction on the subscriptions that the articles come from.

The parameter that limits the number of articles to be downloaded is particularly useful for our Online-LDA algorithm, as the parameters chosen for running Online-LDA affects the topic model, and reference [6] provides a set of empirically selection of parameters that perform well than the others. Therefore, we want to restrict the articles downloaded in every mini-batch to match these empirical parameters. Moreover, the articles downloaded to the web application should be consistent with the ones in Feedly to preserve the user experience. To achieve this goal, we use another API call that Feedly supports to get the number of new articles that is newer than the latest article in our database, and only start the analysis procedure if the number of new articles reaches the mini-batch size. We use this API call frequently to make sure that the number of new articles we need to download does not exceed too much compared to the empirical mini-batch size.

5.2 Google App Engine

Google App Engine (GAE) [13] is a Platform as a Service (PaaS), which allows users to run web applications on Google's infrastructure. Applications on GAE are running in a sandbox environment, which allows the applications to run across multiple servers. Google provides public APIs and toolkits for Python, Java, PHP and Go programming language that help developers integrate web applications into GAE.

GAE has several advantages for deploying, distributing and scaling web applications:

- Automatic scaling and load balancing; users can choose to use GAE's scaling method or manual scaling.
- Scheduled tasks and separated task queues that can be used to manage requests.
- Built-in Datastore and SQL provide easy management of databases.
- GAE connects to other Google cloud services, e.g Google Cloud Storage.

On the other hand, GAE has limitations on the use of its service. The free quota of GAE limits the amount of storage (1 GB), the instance time for both front end instance (28 instance hours) and back end instance (9 instance hours). Users have to enable billing to continue using GAE if they pass the free quota, and the cost depends on the usage rate of the services in GAE.

Another limitation in GAE is the request timeout in front-end instances. GAE limits the timeout of requests on front-end instances to 1 minute, and it will terminate the HTTP request and return an error if a request cannot return within 1 minute. If a task running on GAE requires to run for a long time without returning, it should be running on the back-end instances. However, GAE also has the right to shut down back-end instances without notifying developers due to several reasons, some of which can be the instance is running out of memory, or the instance has to be restarted. These random shut-down happen more frequently when using lower end instances.

5.3 Google NDB Datastore

Google App Engine provides API for NDB Datastore [14], which is a persistent, consistent and scalable schemaless storage for web applications. It is based on Google Bigtable [15], a distributed scalable data storage developed by Google. Some of the highlights for NDB Datastore are:

- advanced querying features,
- high availability of reads and writes,
- strong consistency for reads and ancestor queries, and
- eventual consistency for all other queries.

In NDB Datstore, objects stored are called entities, each entity can contain one or more properties with supported data types. NDB Datastore also has a concept of Model. It can be seen as the counterpart of the database schema, which defines the properties for a specific kind of data.

5.4 Text Pre-Processing

After the articles are downloaded, we conduct text pre-processing to generate an input articles list for the Online-LDA procedure, as well as building the vocabulary set used for both Online-LDA and automatic labeling procedure.

First, when each article is downloaded, we create a list of words that appear in the article by splitting the article with punctuations. Given a list of 525 stop-words, we then scan through the list of words and remove all appearances of stop-words. Stop-words are common words to all documents that do not add any value to the topics or labels, e.g., and, but and the. Finally, after a mini-batch of articles is downloaded, we combine all words lists together and form the input parameter for Online-LDA.

For building the vocabulary set, we take the input word lists for Online-LDA and remove all duplicate words, and create a new list containing only the unique words that appear in this mini-batch.

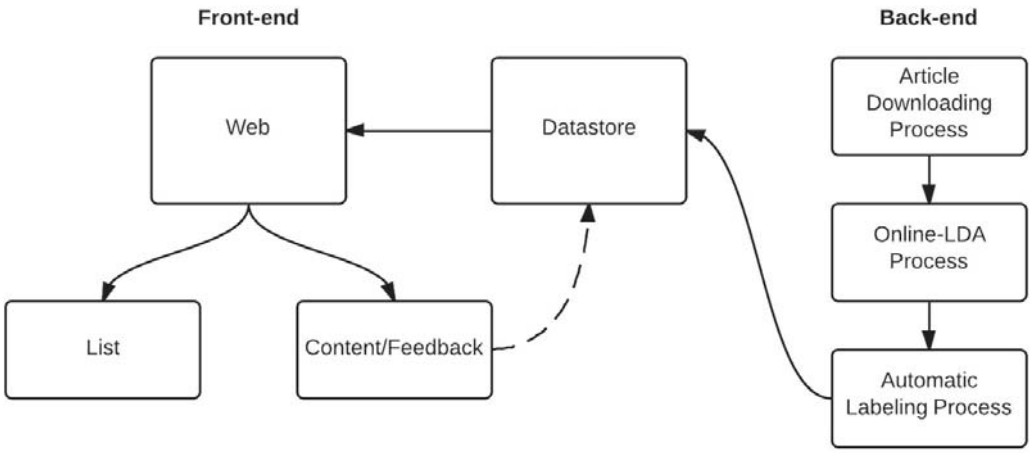

Figure 5.1: Automatic Labeling web application structure

In this report, we use the first mini-batch to generate a vocabulary set and use it as the vocabulary for the rest of the mini-batches. To include as many words as possible while maintaining the efficiency of the algorithm, we double the number of articles downloaded in the first mini-batch.

5.5 Web Application Structure

The structure of automatic labeling web application can be separated into two modules: front-end web UI module and back-end analysis module. Figure 5.1 shows the structure of the web application.

The front-end web module uses GAE's front-end instance to host the UI of the web

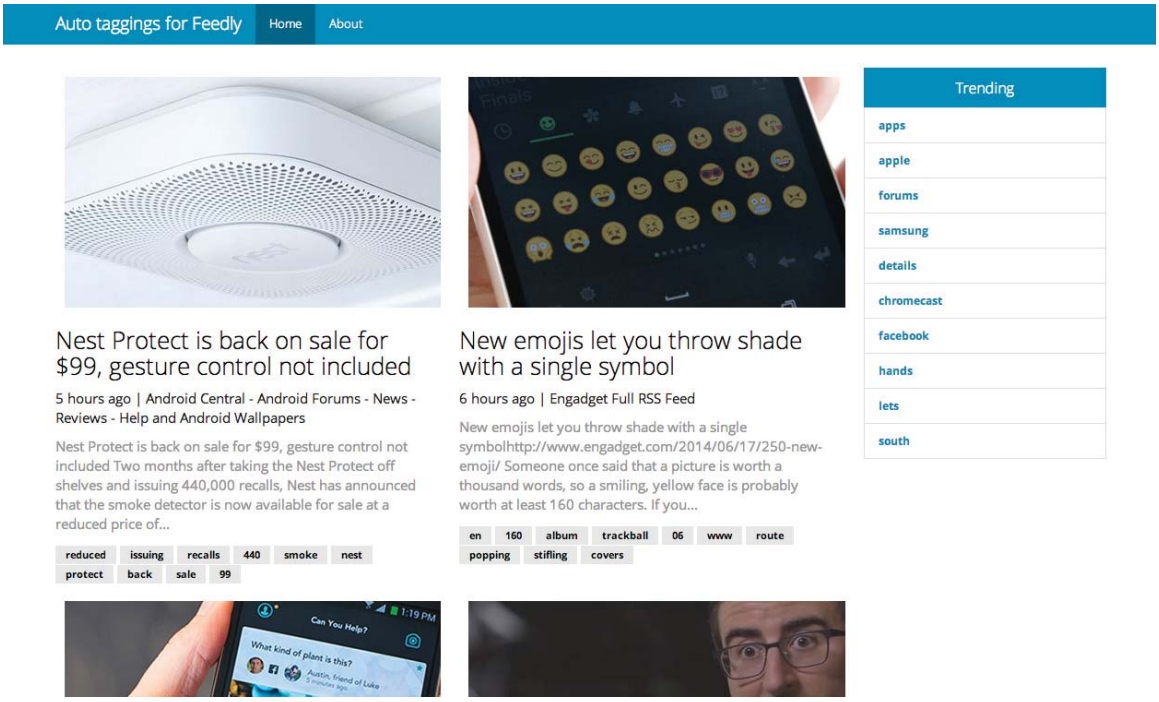

Figure 5.2: Front-end web page article list

application: a web page that displays the articles fetched and labeled with their labels; a web page the content of articles and a feedback system that allows user to rate the accuracy of the labels, and provide suggestions on what words should be used for a particular article.

Figure 5.2 shows the basic layout of the website, which contains article grids and their labels. Also there is a section displaying most frequently used labels, which represent the current trending topics within the labeled articles. Figure 5.3 shows the layout of the article content, containing the article with its meta-data, plus the user feedback system that we use to collect accuracy survey data.

The back-end module is responsible for article analysis and labeling procedure. The

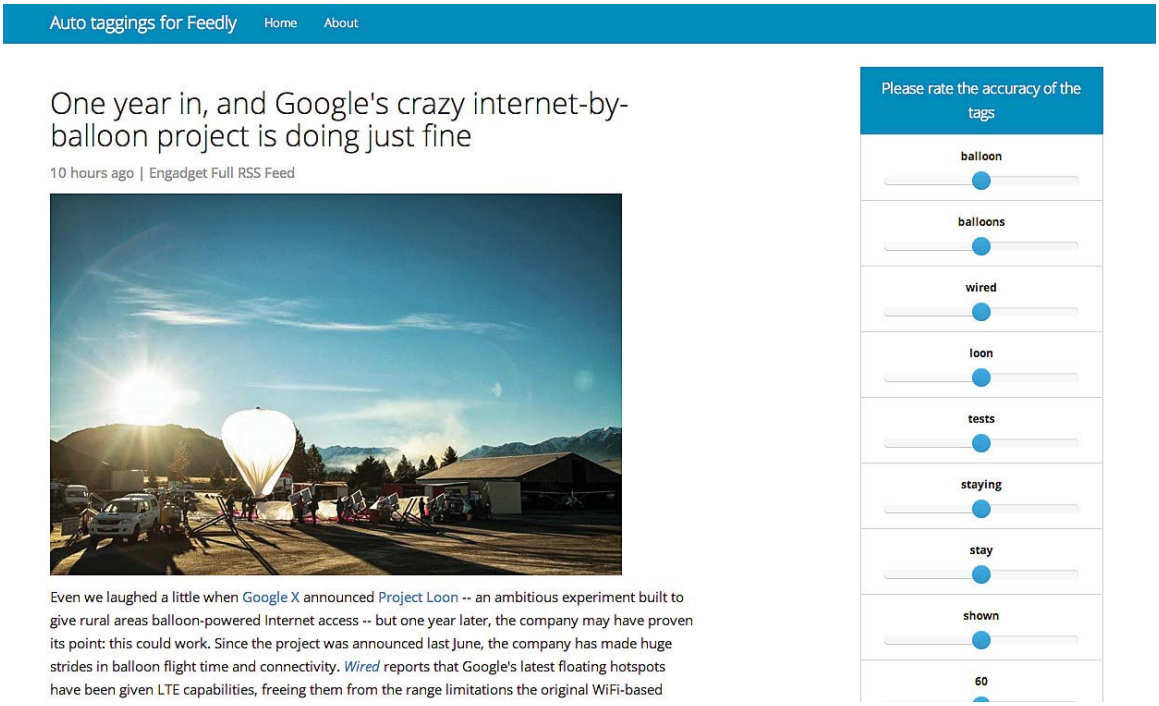

Figure 5.3: Fron-end web page article content

Algorithm 5: Text pre-processing

---

for *Every article downloaded* **do**

1. Remove all stop-words from the article,
2. Create a list of words,
3. Create a dictionary object of unique words based on the list of word as vocabulary, which will be used in Online-LDA procedure.
4. Insert the article and its meta-data (title, posted time, author, source) into a temporary NDB model.

end

Figure 5.4: Text pre-processing

procedure can be divided into three parts: fetching articles from Feedly RSS service, running Online-LDA, and running labeling process. When the module instance is launched, it first queries the Feedly server to check if there are enough new articles to be downloaded. If not, the instance is terminated. Otherwise, the instance downloads these new articles and performs the pre-processing shown in Figure 5.4.

After the pre-processing, an Online-LDA procedure is launched on the instance. This procedure will take the corpus of word lists and the vocabulary generated from the pre-processing as input, and run a mini-batch to update λ and γ. The updated vector λ indicates the likelihood of the latent topics and specific words, and the vector γ indicates the likelihood of latent topics and documents. These two vectors and the vocabulary will be used in the labeling procedure as input.

The labeling procedure starts with reading articles from the temporary NDB model, and for each article: selects a topic that is most important, i.e., has the maximum normalized probability in γ. Then the procedure selects the row corresponding to the selected topic's index, and queries the vocabulary to get a list of likelihoods of words in this topic. Within this list of words, we choose the top 20 words that have the highest probability and also explicitly appear in the article, and add them into our label candidate set. We also generate a list of words without stop-words from the article's title and add them to the candidate set.

The next step of the labeling procedure is to calculate the relevance score for each candidate in the candidate set. Here we use all articles in the temporary NDB model as the context collection $\mathscr{C}$ and perform the first-order relevance scoring. Since the candidates are all generated from the context collection $\mathscr{C}$, we do not consider the bias, and we also ignore the KL-divergence between topics and $\mathscr{C}$.

The final step of the labeling procedure is to sort the candidates with their relevance scores, and choose the top 10 words as the final labels. The procedure will store the articles with

labels into a final NDB model and remove the corresponding entries from the temporary table.

Both front-end and back-end use Google NDB Datastore to store and retrieve data. In our application, we use NDB Datastore with two main NDB models to store our core data: Label model, which stores all labels generated by our automatic labeling process and the references to the articles being labeled, and Labeled Article model, which stores all labeled articles and their meta-data. In addition, we also use a temporary NDB model, "Unlabeled Article", to store the articles downloaded but haven't been labeled with their meta-data. Another use of the NDB Datastore is the feedback from users. As indicated as a dashed arrow in Figure 5.1, we store the average scores for each label and the user suggested labels into a model in our Datastore instance.

5.6 Scaling, Task Queues and Scheduled Tasks

As mentioned in Section 5.2, GAE's limitations on web applications largely shape the implementation of the automatic labeling application, especially the work flow of the labeling process. In this section, we discuss the changes in the work flow and the use of GAE's task queue and scheduled task to work around the limitations [16].

To overcome the front-end request timeout issue, as we mentioned in previous sections,

we separate the web application into two modules, and use GAE's modules API to setup the modules. The front-end module of automatic labeling uses the GAE default module, which has the limitation of 60-second HTTP request deadline. The front-end module is not responsible for time-consuming analysis jobs and mainly focuses on handling HTTP request from and to the web page, which returns fairly quickly in most cases, and therefore we don't have to worry about exceeding the deadline.

The back-end module, on the other hand, uses a basic scaling [16] module provided by GAE to avoid reaching HTTP request deadline. By using basic scaling, the module will automatically create instances when the application receives requests, and the number of instances created is decided by the workload and the maximum number of instances assigned. The basic scaling also uses a parameter called *idle_timeout* to decide when to shut down the instances, that is, if the instance idles longer than the *idle_timeout*, GAE will shut it down. Thus, by using the basic scaling back-end instance, we avoid exceeding deadline termination of the Online-LDA and automatic labeling procedure, and the basic scaling of the module provides a good way to reduce the instances' idle time and therefore reduce the cost.

Another limitation we have to deal with is the random termination of the instances by GAE. Due to the limited resources of the instances, the compute-intensive work we are doing for labeling, and some other reasons that are out of our control, the instances are often shut down by GAE before completing the procedures. The details of the performance issue

Algorithm 6: Automatic labeling with segments

1. Set the number of articles in each iteration to be N.
2. Set the current index to be 0, enqueue a new labeling procedure.
3. **for** *each labeling procedure* **do**
   A. load N articles started from the current index (loaded from temporary storage) from the temporary NDB model,
   B. label N articles, increment current index by N,
   C. store labeled articles into the final NDB model, store the current index into temporary storage,
   D. enqueue a new labeling procedure.

end

4. All articles are labeled, delete all articles from the temporary NDB model.
5. Query Feedly server for new articles, if there are enough new articles, fetch and start Online-LDA procedure. If not, idle.

Figure 5.5: Algorithm for Automatic labeling with segments

associated with random termination is discussed in Section 6.1.

Reducing the time that a single labeling task takes is an relatively easy and efficient way to overcome the termination issue, and this can be done by dividing an original automatic labeling task into several segments, each of which labels a subset of the articles in the original task. To separate the automatic labeling procedure into segments, we take advantage of the task queue in GAE. Here we use the Push Task Queue API that GAE provides to implement the iterations. The algorithm is described in Figure 5.5. The repeated sub-procedure of labeling guarantees they will return in a small amount of time, and therefore reduce the chance to be shut down by GAE.

At the end of the automatic labeling procedure, we enqueue one article fetching procedure to see if there are enough new articles to be analyzed. If the number of new articles doesn't exceed the mini-batch size, we put the instance to idle and let it be shut down automatically.

However, to make sure articles are fetched and labeled on time, we use the Cron scheduled task API in GAE to schedule repeated article fetching procedure. The scheduled task queries Feedly server for new articles every interval, and enqueue a new article fetching procedure if there are enough new articles to be downloaded.

Chapter 6

Evaluation and Performance Analysis

6.1 Performance Analysis

This section focuses on the performance analysis of the automatic labeling procedure, specifically the effect of GAE instance and the effect of different vocabulary sets and contextual collection sizes.

6.1.1 Segment Size

We conducted an experiment to show that the effect of instance type and segment size on automatic labeling success rate. In our experiment, we use one B1 instance as our back-end

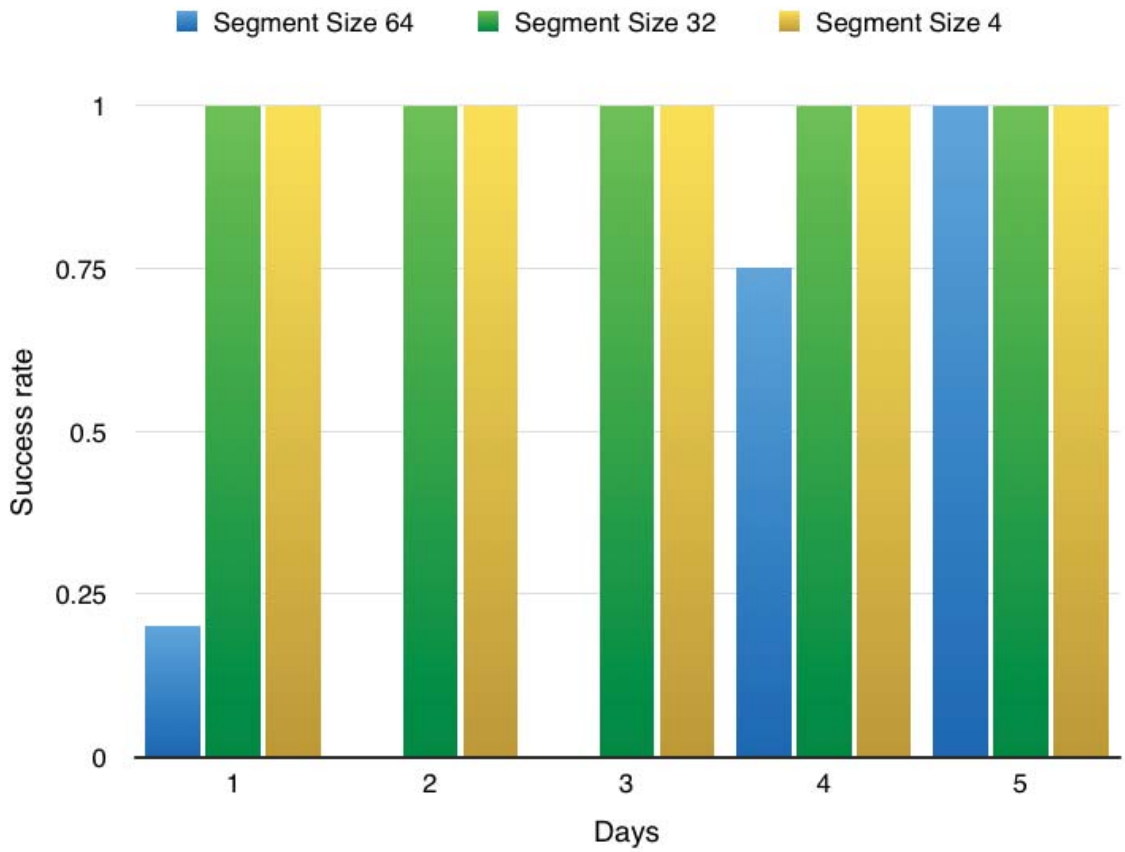

Figure 6.1: Segment size and success rate of automatic labeling procedure

instance (lowest instance type), select 2,400 words as the vocabulary set and run multiple automatic labeling procedure with different segment sizes. We then observe the running status of the instance and record the success rate, i.e., the number of completed tasks over all tasks. To be specific, we run the experiment with 3 different segment sizes: 64, 32 and 4, and we run multiple automatic labeling procedures over 5 days. Figure 6.1 shows the success rate of the automatic labeling procedure for the three segment sizes. Note that in this experiment, we only mark a task failed if it is terminated by GAE before the analysis completes.

As shown in Figure 6.1, the success rate of segment size 64 is the lowest. Interestingly, for this segment size the success rate was low in the first 3 days, but succeeded almost every task in the last two days. An explanation could be that there were different workloads within GAE over these days, where in the first 3 days it had to terminate the instance to give resources to other instances while the last 2 days the workload was relatively low. The average time that the successful segments use is 6,134.98 seconds. On the other hand, the segment size 4, a conservative segment size, yields a success rate of 100%. By using this segment size, we reduce the time that each segment requires to 500 seconds to 1,000 seconds, and since each segment only tries to label 4 articles, it completes with a small amount of resource required; the segment is never terminated earlier by GAE. However, due to the requirement of storing temporary states of the computation, the overall time spent on the segments is increased compared to using segment size of 64. For segment size 32, all segments completed successfully. Furthermore, using segment size 32 shortens the time spent on the automatic labeling procedure compared to using segment size of 4. For segment size 32, the average time for running a segment is 4,427.48 seconds, and since for every 64 articles, we need to run 2 segments, the overall time cost is worse than using segment size of 64, but better than segment size of 4. Therefore, to ensure the success rate of the segments, a smaller segment size is preferred; yet to improve the performance of the procedure, we want the segment size to be larger.

	Vocabulary size	
Contextual collection size	2,400	4,000
64	1,427.82	2,048.52
128	2,492.99	4,516.92

Table 6.1
Running time of labeling a segment of 32 articles with different contextual collection size and vocabulary size (seconds).

6.1.2 Vocabulary Set Size and Contextual Collection Size

To examine the performance of the automatic labeling given different sizes of vocabulary and contextual collection, we conducted an experiment on local environment instead of GAE to measure the running time of the procedure segments. We select local test environment over GAE environment because of the easy control of parameters adjustments and easy access to logs. GAE provides a simulated cloud environment that runs locally but simulates the running environment of real GAE web applications, we use this development environment to perform the experiment. The experiment was conduncted on a mid-2012 Macbook Pro with 2.6 GHz Intel Core i7 processor, 8 GB 1600 MHz DDR3 memory and 256GB SSD.

In the experiment, the segment size was fixed at 32 articles, and two other parameters, vocabulary size and contextual collection size, were adjusted to examine the running time. Each combination was run 5 times, each time a segment of 32 articles will be labeled, and the average running time is used for comparison. Figure 6.1 shows the result of the

experiment. As expected, increasing vocabulary size or contextual collection also increases the running time of the automatic labeling procedure, as the vocabulary size affects the size of λ, and the contextual collection size affects the number of iterations required to complete the automatic labeling procedure. However, having larger contextual size and vocabulary size will result in potentially better label quality, since the the larger vocabulary and collection could cover more aspects of the articles and hence provide better candidates.

6.2 Evaluation

6.2.1 Evaluation Setup

Our evaluation examines two different categories of news articles sources: a set of news articles dedicated to technology and consumer electrics products, and a set of news articles mixed with 5 different topics: business, cooking, design, news and technology. The details of the news article sources can be found in Appendix A. In both categories, the articles are all labeled with 10 unique labels selected set of candidates sorted by relevance scores.

Our web application was run on a single GAE B4 instance with 2.4 GHz CPU frequency and 512 MB memory limit. In the Online-LDA procedure, we chose 64 as the mini-batch size (except for the first mini-batch), and used the parameter values $K = 50$, $D = 3.3 \times 10^6$, $\alpha = 0.02$, $\eta = 0.02$, $\tau_0 = 64$ and $\kappa = 0.7$. A vocabulary set of 6,000 words generated from

the first 128 articles was used as the vocabulary set of the procedure. For each automatic labeling procedure, we chose a contextual collection size of 64, and a segment size of 8 was used to minimize the random termination occurrence due to the large vocabulary used in the web application.

In our evaluation, users get access to our web application through their web browser, and then choose articles from the article list to read. While users are in the article content page, there are 10 labels assigned to the article from our automatic labeling procedure, together with 10 control interface used to rank the accuracy of the labels: not related, not very accurate, neutral, somewhat accurate, and accurate. Users can also type labels in the input box below the 10 labels to provide further suggestions on how the article should be labeled. When users complete the feedback form, they can submit the result to the web, which is running on the front-end instance. The front-end instance then computes the average score for each label in the article with the points assigned: 1 for not related, 2 for not very accurate, 3 for neutral, 4 for somewhat accurate and 5 for accurate. Every label has a score of 3 by default.

6.2.2 Evaluation Results and Analysis

Average Scores for Tech and Mix Test Account

Over 5 days of gathering user feedback on the web application, we collected feedback

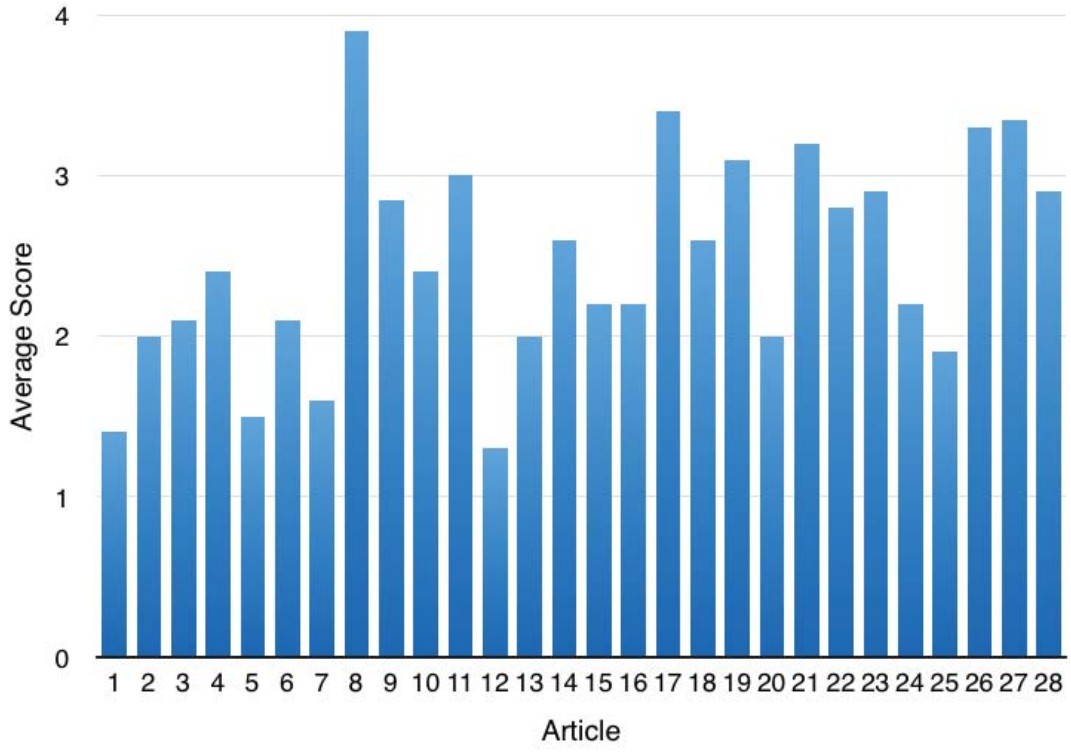

Figure 6.2: Average score of Tech account labels

information for 41 unique articles (28 in Tech account, 13 in Mix account) from 51 users; 39 out of the 41 articles contains labels that users provided according to their suggestions of the labels.

The average scores for each article under the two test accounts are summarized in Figure 6.2 and Figure 6.3. The overall average scores in Tech website is higher than that of Mix. The result is expected, as the Mix account contains more topics and is more dynamic than the Tech account, which focuses only on technology news. The dynamic vocabulary and the large number of topics contained in the Mix account impacts the effectiveness of selecting a vocabulary set, especially when we generate a vocabulary set directly from a mini-batch

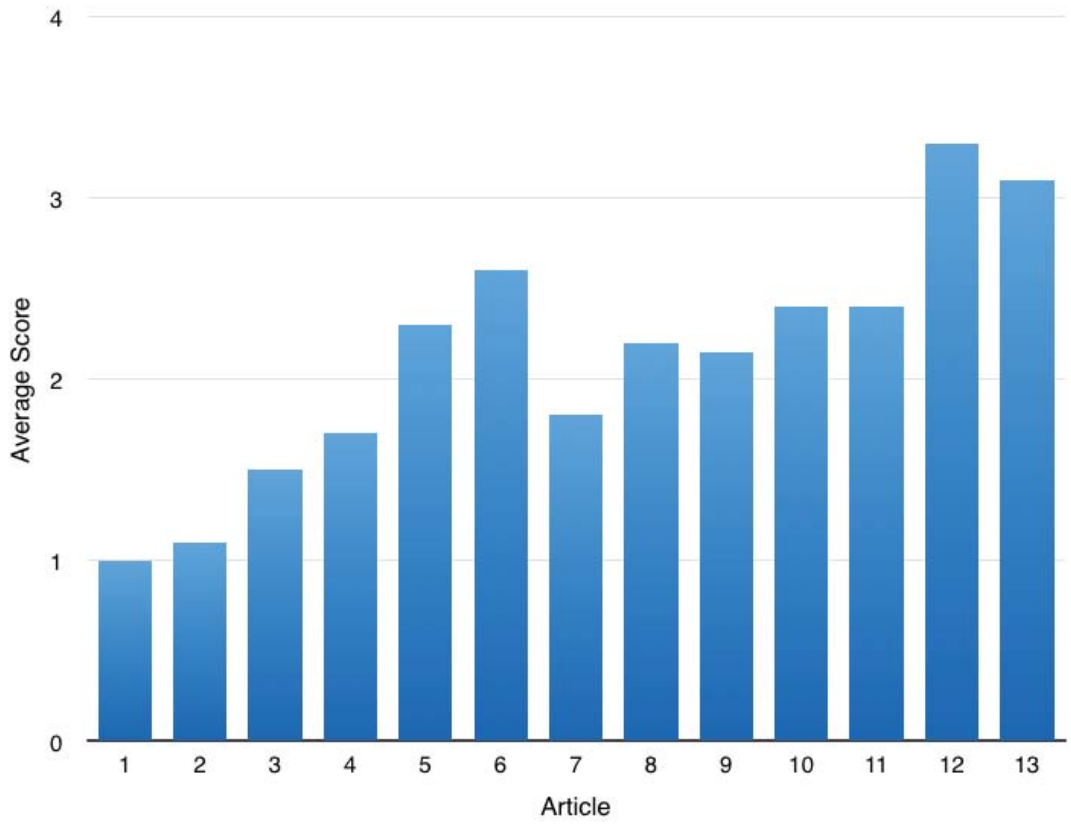

Figure 6.3: Average score of Mix account labels

of Online-LDA. Given that the topics might exceed the number that we assign prior to Online-LDA procedure, as well as the vocabulary might not cover all aspects of the topics, it is more likely that the words that should be considered as label candidates actually appear later and are not included into the vocabulary set. In contrast, the Tech account received higher average scores in general, as the news articles focus on the technology and consumer electronic products, given that the number of topics are smaller and are more likely to be close to the value we assign, and the topics are relatively static compared to Mix account. The model is more accurate in terms of representing the latent topic model, and the vocabulary set has a higher chance to contain words that should be considered as

	# of noun	# of others	Ave. score for noun	Ave. score for others	p-value
Tech	185	95	2.657	2.105	0.000242
Mix	87	43	2.316	1.744	0.00861

Table 6.2
Number of nouns and other types of labels and their average scores.

label candidates.

Overall Evaluation Result Analysis

Even thought Tech account has a higher average score, neither Tech nor Mix gives a good average score in general. We compare the labels generated from automatic labeling algorithm and the user feedback, and try to explore the reasons.

Appendix B shows the labels selected by automatic labeling algorithm and the users' suggested labels for the 39 articles. An observation that can be drawn is that the users suggested labels are usually nouns that cover the main subjects of the articles; on the other hand, the labels selected by the algorithm are a mixture of nouns and other types of words. By comparing the labels selected by the algorithm and the users' feedback, one can see that the words users prefer are more general, while the algorithm selects words that appear more frequently and are more specific. The words users prefer and are not included in the algorithm generated set are the ones that appear only once or a few times in the contextual collection and the article itself. The observed parameter of the algorithm is the set of words and their frequency, and the words that appear only once will not be treated as important and associated with higher probability in the algorithm.

Figure 6.2 shows the number of nouns and other types of words selected by the algorithm as well as their average scores and the p-value from a two sample t-test. The automatic labeling algorithm also tend to assign higher scores for nouns than the other type of words. The small p-value indicates that the scores for nouns and others are significantly different in both accounts and the average scores for nouns and the other types of words reflects users' preference on nouns; the average scores for nouns are higher than the other types in both Tech and Mix test accounts.

Finally, the words selected from the titles received a relatively higher scores compared to the other words generated from the contents. Among the labels generated, 23 of them are selected from the titles (22 in Tech account, 1 in Mix account). The average score for this type of words is 4.5, which is much higher than the other labels. This proves the importance of the words in the titles for representing the topic of the articles.

6.2.3 Article Labels and Vocabulary

Our approach to generate vocabulary for Online-LDA and automatic labeling is to generate this directly from the first mini-batch. However, it is possible that the words that appeared in the first mini-batch cannot cover the articles downloaded later, and therefore affect the accuracy of the labels. We chose 8 articles (2 per day) from the 307 Tech account articles that were downloaded across 4 days, and compared their average ratings. Note that each of

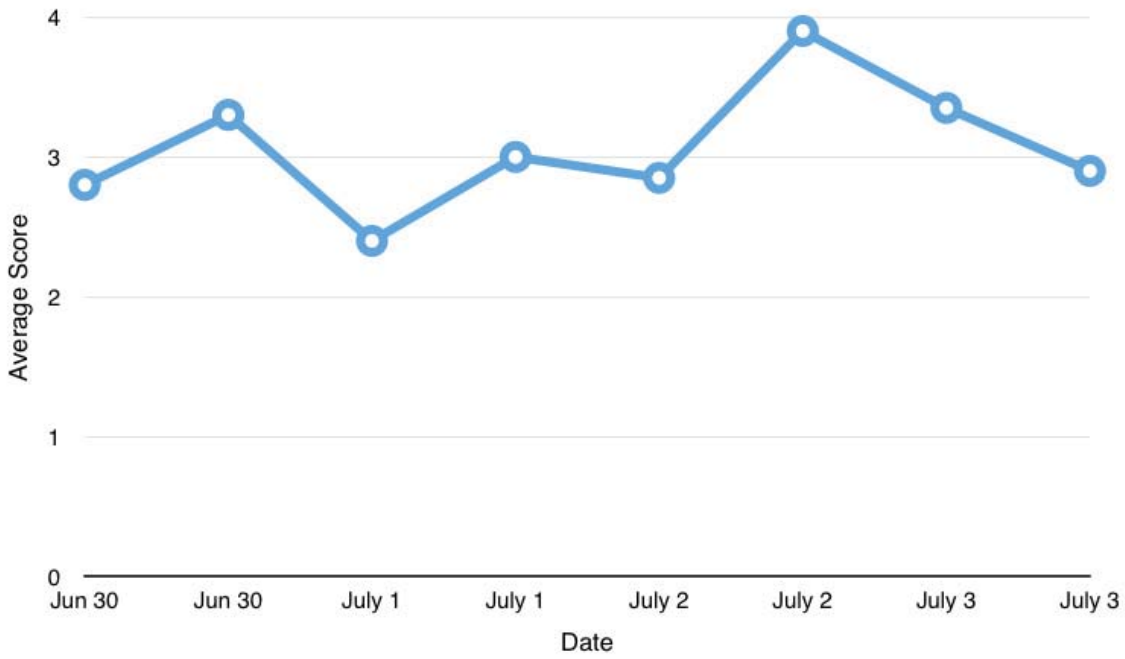

Figure 6.4: Average scores of the labels from the 8 Tech account articles across 4 days

the articles contains 2 users' feedback, and we took the average of their ratings.

In Figure 6.4, the average ratings do not have a decreasing pattern; one of the article in the third day even has the highest average rating among all 8 articles. Given that the articles in the Tech account are all related to technology, the number of topics are smaller and the words used in articles might be similar. Therefore in this scenario, for a smaller latent topic set, for 4 days the vocabulary set is still valid for Online-LDA and automatic labeling; it still covers most of the words used in the articles throughout the days. However, further experimentation is needed to examine the effect of static vocabulary versus frequently updated news articles.

6.2.4 Comparing with Online-LDA Evaluation Results

In [6], the corpus *Nature* and Wikipedia was used for evaluation. A list of 4253 words was used as the vocabulary set for *Nature* corpus, and a list of 7702 words was used as the vocabulary set for Wikipedia corpus.

Compared to [6], the average length of the articles in our RSS news source is smaller than that of either *Nature* or Wikipedia, which reflects the word frequencies in the automatic labeling algorithm: in our topic model, most of the label candidates only appear once within a mini-batch.

Another difference between the evaluation in [6] and ours is the vocabulary set. The vocabulary is selected and pruned in [6], but the vocabulary in our evaluation is directly generated from the articles. In a real world situation, due to the fact that articles in RSS sources are constantly updated, the vocabulary in RSS news is more dynamic than static articles; therefore, a small portion of articles cannot cover all of the articles in iterations of Online-LDA algorithm. If the vocabulary cannot reflect words used in the articles, the algorithm cannot provide useful label candidates in the topic model.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this report, we have explored and implemented the approach of an automatic labeling for RSS news articles. We combined the Online-LDA [6] and automatic labeling algorithm [5] to generate labels automatically for news articles. The approach takes a stream of RSS news articles as input, constructs a topic model on the mini-batches of articles, and use the topic model as label candidates for automatic labeling algorithm. The labels generated for an article are the words that appear explicitly in the article and have the highest relevance score among the candidates. We have implemented this approach with Python and deployed it on Google App Engine as a web application, which automatically download articles from

Feedly RSS service and generates labels for each of the articles.

7.2 Future Work

There are several places that we expect to improve in the future. First, the vocabulary is currently set and fixed upon the first mini-batch. An approach to incorporate the update of the vocabulary set is necessary for this online application as the stream of articles eventually will bring new words and we want to consider them as part of the potential label candidates. Second, there is room for a better algorithm to choose label candidates, as we only consider words that explicitly appear in the articles.

In terms of implementation, potential improvements can be found in better use of GAE Datastore to improve the data structure of the web application to gain efficiency. We also look forward to using other Feedly APIs to enhance the quality of labeling by including more information, e.g., the sources information and the manual labels for the articles if available.

As for evaluation, we can improve the quality of the feedback by collecting more ratings from users. The users submitted scores are usually from personal perspective, which can be biased. To make the feedback as objective as possible, we can collect more feedback and examine the statistics to evaluate the quality of the labels generated.

References

[1] Blei, D. M.; Ng, A. Y.; Jordan, M. I. *the Journal of machine Learning research* **2003**, *3*, 993–1022.

[2] The size of the World Wide Web (The Internet). `http://www.worldwidewebsize.com/`.

[3] Board, R. A. *Web available* **2007**.

[4] Kullback, S.; Leibler, R. A. *The Annals of Mathematical Statistics* **1951**, pages 79–86.

[5] Mei, Q.; Shen, X.; Zhai, C. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 490–499. ACM, 2007.

[6] Hoffman, M.; Bach, F. R.; Blei, D. M. In *Advances in Neural Information Processing Systems 23*; Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., Culotta, A., Eds.; Curran Associates, Inc., 2010; pages 856–864.

[7] Feedly Cloud API. `http://developer.feedly.com/`.

[8] Jordan, M. I.; Ghahramani, Z.; Jaakkola, T. S.; Saul, L. K. *Machine learning* **1999**, *37*(2), 183–233.

[9] Jordan, M. I. *Learning in Graphical Models:[proceedings of the NATO Advanced Study Institute...: Ettore Mairona Center, Erice, Italy, September 27-October 7, 1996]*, Vol. 89; Springer, 1998.

[10] Blei, D. M.; Griffiths, T. L.; Jordan, M. I. *Journal of the ACM (JACM)* **2010**, *57*(2), 7.

[11] Geman, S.; Geman, D. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **1984**, (6), 721–741.

[12] Gelfand, A. E.; Smith, A. F. *Journal of the American statistical association* **1990**, *85*(410), 398–409.

[13] Google App Engine. `https://developers.google.com/appengine/?csw=1`.

[14] Python NDB. `https://developers.google.com/appengine/docs/python/ndb/`.

[15] Chang, F.; Dean, J.; Ghemawat, S.; Hsieh, W. C.; Wallach, D. A.; Burrows, M.; Chandra, T.; Fikes, A.; Gruber, R. E. *ACM Transactions on Computer Systems (TOCS)* **2008**, *26*(2), 4.

[16] App Engine Modules in Python. `https://developers.google.com/appengine/docs/python/modules/`.

[17] Krestel, R.; Fankhauser, P.; Nejdl, W. In *Proceedings of the third ACM conference on Recommender systems*, pages 61–68. ACM, 2009.

Appendix A

News Article Sources

A.1 Technology

- Android
  - Android and Me
  - Android Central
  - Cult of Android
- OS
  - Cult of Mac
  - Maximum PC
- Blogs
  - Engadget
  - Lifehacker
  - The Verge

A.2 Mix

- Business
  - Business Insider
- Cooking
  - Food Network Blog
  - Joy the Baker
- Design
  - Cool Hunting
  - Design Milk
  - Yanko Design
- News
  - The Huffington Post
- Technology
  - Lifehacker
  - ReadWrite
  - The Verge

Appendix B

Evaluation Result: Generated Labels and Feedback Labels

B.1 Tech

Title	Generated labels	Feedback labels
A 23-foot-tall home-made Transformer is the world's most intimidating lawn ornament	kid, alero, eye, father, catch, robot, problems, prime, kind, names	Transformers
HTC One M8, M7 and Moto G GPe receiving Android 4.4.4 update	vulnerability, openssl, updategoogle, bug, rolled, rolling, edition, related, nexus, htc	android, google, google play, google play edition, motorola
Top app, device and accessory sales for July 2, 2014	murica, farther, wednesday, monday, birthday, usual, earlier, happy, top, app	sale, android, google play
Microsoft's smartwatch is reportedly a fitness band with smartphone notifications	thurrott, heart, filing, patent, rumored, wearable, iwatch, works, sold, steps	microsoft, smartwatch, wearables, watch, notifications
Microsoft asks gamers to help shape future Xbox One updates	moment, redmond, neogaf, providing, livetv, gamifies, singlequestion, spotted, unblinking, guess	microsoft, xbox, xbox one, updates

HTC announces Q2 earnings, first profit this year	creator, slow, based, quarterly, nt, quarter, center, billion, tablets, 43	android, smartphones
The best Twitter apps for Android	lives, allinclusive, playbyplay, playbyplays, timeline, tweets, besttwitterapps, schedule, shared, features	twitter, android,Twitter
E-sports tournament now open to all genders after internet outrage	ii, federation, iesf, competitions, starcraft, hearthstone, policy, esports, tournament, open	gender, sexism, gaming,
IFTTT Android wear channel automates things from your wrist	select, actions, based, trigger, recipes, simpler, criteria, automated, triggered, folks	IFTTT, android, android wear, smartwatch, Android
Showtime's Anytime TV service starts streaming on your Xbox 360	cord, showtime, previousgeneration, showtimefree, jackie, cutters, penny, provider, ondemand, stream	Anytime, xbox 360, xbox one, xbox,Xbox
T-Mobile launching Nokia's Lumia 635 with Windows Phone 8.1 on July 5th	choice, nokia, storage, step, sale, home, hit, tmobile, launching, nokias	windows 8.1, lumia, launch, smartphone,Lumia
Android Wear app now available for all in Google Play	minute, administering, grab, begin, opened, hands, expecting, arrives, glass, haven	android, google, google play, android wear, wear, wearables, watch
Latest Galaxy Tab S commercial highlights the Super AMOLED display	panels, wqxga, 2560, showcase, lineup, 1600, centre, resolution, displays, contrast	samsung, galaxy, tablet, amoled, android
Start with These Camera Settings to Take Great Fireworks Photos	speed, shooting, shutter, bunch, manual, photography, diy, noted, blast, pictures	
Google dumps porn from its ads	imminent, scrambling, related, banned, forbidden, dating, escort, pornographic, jumped, tangoed	google, ads, adwords

Third-person Oculus Rift hack delivers a true out-of-body experience	obscenely, amateur, peer, controlled, carried, wearer, staring, experienced, stereo, makers	oculus rift, wearable, prototype, gopro, third person
Duolingo puts language learning on Android Wear	pushed, introduces, exciting, updates, app, duolingo, puts, language, learning, android	
Massive update to Google Docs and Sheets brings a new UI, Android L support and more	enhancing, greatly, quickoffice, partially, native, moregoogle, confirmed, theme, rolling, word	android, google, google docs, docs, drive, android L
Microsoft seeks Office for Android testers as it readies tablet version	upcoming, nadella, prerelease, word, conference, interest, expected, participants, push, scheduled	microsoft, android, office, microsoft office, tablet
Berkeley will give free weed to homeless medical marijuana patients	cbs, paying, dwell, lowincome, spokesperson, unanimously, sean, city, total, cruel	weed, medical, marijuana, homeless
Channel Your Inner Entrepreneur to Excel at Work	jared, snyder, organizing, state, complete, lifehacker, china, global, identify, channel	entrepreneurship, entrepreneur
Apple brings two-factor authentication to iCloud.com	verify, password, attempting, id, identity, enabled, temporary, icloud, code, require	Apple, authentication
Flying the uncertain skies with the latest Phantom drone	flying, uncertain, skies, latest, phantom, drone, vision, corner, wild, pro	DJI, flight, camera,
Sunrise Calendar jumps from iOS to Mac, with support for Facebook, Evernote, etc.	songkick, asana, integration, services, source, networks, switch, tomorrow, popular, update	calendar, sunrise, apps
Sign-Ups for Evolve PC Alpha Being Accepted Now	survey, shooter, sean, studios, happyhunting, shooters, participant, 1113, refrain, xbox	game, gaming, alpha
AppleâĂŹs third-quarter earnings call is coming July 22	retail, cook, billion, release, quarter, recently, store, apple, apples, thirdquarter	earnings,earnings

'Fast & Furious 7' release date moved forward to April 3rd, 2015	fastfurious, ff7, walker, dramatic, space, completed, movie, announced, july, earlier	film, release,
Disney brings popular Facebook soccer game Bola to iOS	football, tournament, 7, running, acquired, play, store, live, lets, disney	ios, facebook, soccer

B.2 Mix

Title	**Generated labels**	**Feedback labels**
Indie Bookstore Rehires Workers It Fired For Supporting The Union	lieu, morningside, nearunanimous, contended, shortly, annie, terminated, doeblin, dispute, reinstated	union
Sometimes, Canned Beer Is Actually Cheaper Than a Keg	ounce, kegduring, dudephoto, cans, compared, resulted, patrick, cups, tastes, tipping	beer, keg
Chinese Artist Exhibits Gorgeous 'Sculptures' Built By Bees	eliminate, transparent, beijingbased, cells, coolhunting, hangzhou, dice, symmetrically, thusly, courtesy	bees, art
Neymar Out Of World Cup With Fractured Vertebrae (VIDEO)	fit, vertebra, postgame, lasmar, superstar, brazilian, rodrigo, zzazreqcus, pain, espnfc	football, world cup, epsn
James Rodriguez Is The Rising Star Of 2014 World Cup	envigado, argentina, coach, brazil, shy, talented, technical, titles, champion, professional	fifa, world cup, football, soccer
Learn How to Cook Anything on the Grill With This Infographic	memorizing, chicken, beef, burgers, infographics, seafood, infographiceven, undercooked, overcooked, tells	cooking, grilling

Reclaiming "USA!, USA! USA!" From the Bigots in Murrieta	constituents, antiimmigrant, reform, policymaking, shred, nclr, anniversary, unconscionably, riowa, dedicated	immigration, policy
This Giant Picnic Blanket Will Forever Transform Your Summer Snack Fantasies	snacks, sunshine, tablecloths, almighty, duo, sandwiches, function, feat, patrik, peanut	art, picnic
Neymar Injured, Might Not Play Against Germany	speed, yellow, knocked, evan, blow, pitch, doherty, missing, thiago, cards	fifa, world cup, football, soccer,Football
NOVA: A Wireless Flash for Better iPhone Photos	presets, ugliness, bluetooth, mode, reign23, manual, preference, phones, unrelenting, soft	iphone, flash, camera, photography, photo
LOOK: Chile Releases Stunning UFO Photos	dgac, meteorological, cefaa, enlarged, jose, ceffa, collahuasi, shown, adds, normal	UFO, drones
For Your Viewing Pleasure, We Present Some Nude Piles As Art (NSFW)	nudes, raft, berlin, ambiguous, earliest, theodore, converge, togetherness, bratislava, bencicova	
How China's Most Famous Grounded Artist Collaborated With A Navajo Man Thousands Of Miles Away	vase, organizers, expecting, dropping, weiwei, causing, commissioned, landscapes, fe, pottery	Navajo, art