A Comparison of Approaches to Determine Topic Similarity of Weblogs for Privacy Protection

A Thesis Presented

by

Dong-Yi Wu

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Computer Science

Stony Brook University

December 2009

Stony Brook University

The Graduate School

Dong-Yi Wu

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis.

Amanda Stent – Thesis Advisor

Associate Professor, Computer Science Department

Luis Ortiz – Chairperson of Defense

Assistant Professor, Computer Science Department

Tamara Berg – Committee Member

Assistant Professor, Computer Science Department

This thesis is accepted by the Graduate School

Lawrence Martin

Dean of the Graduate School

Abstract of the Thesis

A Comparison of Approaches to Determine Topic Similarity of Weblogs for Privacy Protection

by

Dong-Yi Wu

Master of Science

in

Computer Science

Stony Brook University

2009

Today the popularity of social networking and content sharing websites results an increasing amount of personal information and opinions exposed to the public. Yet over-revealing such information can be harmful, exposing content authors to judgment and fraud. Consequently, content authors need to balance their desire for publicity with their need for privacy.

Traditional methods of managing access control (e.g. friends lists) are restrictive and/or require extensive manual configuration. In recent work, *content-based access control* has been proposed as a more flexible method for access control in social networks. Typically, two people may want to share content if their content addresses similar topics from similar perspectives. Content-based access control analyzes the text in content and its metadata to determine whether one content author should have access to, or may want access to, content produced by someone else.

In this thesis we present a comparison and analysis of several statistical text-based approaches to measuring topic similarity between two documents. We use natural language processing technology to automatically annotate the input documents, so we also present and evaluation of three freely-available natural language processing (NLP) systems that can provide these annotations.

Contents

List of Tables

List of Figures

cx

Chapter 1

Introduction

Nowadays social networking websites, weblogs, online picture galleries, video sharing sites and many other types of content sharing websites are very popular. One result is that an increasing amount of personal information and opinions is available and accessible to the public. However, users of social networking websites may still want to keep some information private or only shared with certain groups because revealing too much information may lead to following consequences (Hart, Johnson, and Stent, 2007):

- People have lost their jobs when an employer discovered an employee's personal blog.
- Sexual predators use social networking sites to find victims.
- Stalkers use personal information on social networking sites.
- Universities have used photographs taken at student parties and posted on photo sharing sites against students.

Access control is a mechanism that lets a content owner (e.g. a blogger) have control over the accessibility of each piece of content he/she owns. Conventional access control mechanisms provide few options, such as "private", "friends only" and "public". Also, they requires users to manually indicate the access control option for each piece of content published to the internet. In a typical scenario of using traditional access control, an blogger may have to explicitly list every individual allowed to access the post unless he is willing for the post to be public.

To simplify the process of managing public and private information on social networking sites, Hart, Johnson, and Stent (2007) have proposed *content-based access control* and started working on the testbed of this concept, PLOG (A **P**rivacy/**P**olicy aware b**LOG**ging system) . Our work aims to aid the construction of PLOG's privacy control policy over the text in posts to social networking sites, particularly blog posts.

In this thesis, we present a comparison between Latent Semantic Mapping (Bellegarda, 2005) and a simple cosine similarity approach (Schutze and Manning, 1999) to determine the similarity between blog posts. We also experiment with three different sets of input units to each of these approaches: entity and non-entity terms, verbs and entities, and simply words. Also, we evaluate three NLP (Natural Language Processing) systems currently available and suggest an NLP pipeline to automatically label words, verbs and named entities in input to our approaches.

Chapter 2

Related Work

Related work comes from several fields: Social Network Analysis, Topic Extraction and Clustering, Sentiment Extraction and Natural Language Processing. Here we briefly describe only related work at the core of the questions in this thesis:

- What makes two documents "similar"?
- How can systems like PLOG help users identify "similar" documents?
- How can systems like PLOG automatically identify "similar" documents?

2.1 Defining Document Similarity

(Mei et al., 2007) created an interesting model for analyzing both topics and topic sentiment in weblogs. Their model has five essential constituents:

1. Topic model: defined by a probabilistic distribution of words representing a semantically coherent topic.
2. Sentiment model: defined by a probabilistic distribution of words representing either *positive* opinions or *negative* opinions.

3. Sentiment coverage: defined by the relative coverage of neutral, positive and negative opinions about a topic in a document or in a collection of documents. For a topic in a document, the sum of the probabilistic distributions of neutral, positive and negative opinions equals to one.

4. Topic life cycle: is a time series representing the strength distribution of the neutral contents of a topic over a period.

5. Sentiment dynamics: stands for a time series representing the strength distribution of positive and negative sentiments over a topic.

Note that the forth and fifth constituents were specifically tailored to reflect the changes on topics and opinions over time.

Inspired by (Mei et al., 2007), we would like state to our definition of document similarity. When we say two documents are similar, we mean that the documents address similar topics and that the opinions expressed about the topic are similar. In this thesis, we concentrate only on topic similarity, which we define similarly to the notion of topic model defined in (Mei et al., 2007). We basically assume that documents about similar topics will be comprised of similar words, and the topic similarity is proportional to the frequency of occurrence of the common words.

2.2 Semi-Automatic Document Similarity Tagging

There has been quite a lot of work on clustering of documents by topic using the words in the documents (e.g. (Sood and Hammond, 2007; Mishne, 2006; Hayes and Avesani, 2007; Banerjee, Ramanathan, and Gupta, 2007; Nallapati and Cohen, 2008)).

One way to cluster documents in social networking sites, such as blogging sites, is through metadata tags associated with documents. The *TagAssist* and *AutoTag* tag suggestion systems both automatically create metadata tags through text analysis (Sood and Hammond, 2007; Mishne, 2006). They operate using similar methods. Basically, both systems find existing, tagged posts that are similar to a new, untagged post and aggregate tags associated to those posts, rank the tags, and finally recommend the highest-ranked tags to users. There are minor differences mostly in the factors for ranking tags, e.g. using tag frequencies, clustering algorithms, etc.

Hayes and Avesani (2007) proposed a system that involves pairwise comparisons between documents. Their system uses the text in the weblog posts and the metadata tags associated with the posts. Banerjee, Ramanathan, and Gupta (2007) and Nallapati and Cohen (2008) used additional factors to cluster documents, including Wikipedia entries and between-document hyperlinks.

Our system uses only the text in weblog posts as some of these authors report that tags are weak indicators for partitioning weblog data. We leave the use of these additional factors in our system to future work, as they may have privacy implications (e.g. a spammer may link to many others' documents in order to get access to their weblogs).

2.3 Fully Automatic Document Similarity Tagging

Metadata tags are currently assigned manually by the content author, so tag-based document similarity methods are only semi-automatic. There are also fully automatic methods for computing document similarity. We studied two: lexical chains and latent semantic mapping.

Lexical chains are clusters of words that are semantically related. The relationships between words in the clusters are typically synonym (same meaning), hypernym/hyponym (more generic/more specific meaning) or meronym/holonym (whole-part/part-whole relationships). For instance, in the lexical chain {*house, loft, home, cabin*}, *house* and *home* are synonyms, *attic* is part of a *house* and *cabin* is a specialization of *house* (Doran et al., 2004). Doran et al. (2004) and Silber and McCoy (2002) are two papers that use lexical chains (computed through WordNet) to identify document similarity. This method is quite fast and accurate, but can only be used to compute topic similarity.

Latent Semantic Mapping has been shown to be effective for numerous tasks including information retrieval, word clustering, document clustering, topic clustering, large-vocabulary speech recognition language modeling and more (Bellegarda, 2005). We decided to use this method for our work because it can identify underlying semantic relationships even if individual words are not related, so we hypothesized that it would help to identify pairs of documents similar in both topic and sentiment.

Chapter 3

Data

We used two sets of data for our experiments: a large set of training data containing LiveJournal blog posts, and a smaller set of testing data containing blog posts from several blogging sites. As far as we know, no blogger contributed data to both data sets.

3.1 Training Data

Our training data consists of blog posts collected from LiveJournal[1]. This data set was automatically labeled with part-of-speech and named-entity tags by Lydia (Kil, Lloyd, and Skiena, 2005; Lloyd, Kechagias, and Skiena, 2005). A part-of-speech tag indicates whether a word is a noun, a verb, an adjective, etc. A named entity tag indicates whether a word is part of the name of an organization (e.g. *Stony Brook University, IBM*), a person (e.g. *Dong-Yi Wu, President Obama, Dr. Sarah Jones*), a location (e.g. *New York, Silicon Valley*), etc.

Before using this data set in our experiments, we first filtered out all posts containing non-ASCII text. The resulting reduced data set contains 5866528 blog posts and is 15GB in size. An example post is shown in Figure 3.1.

We used the training data to build matrices using Latent Semantic Mapping.

[1]http://livejournal.com

LAST_NAME	AGE	PARK
MONTH_PERIOD	RAILROAD	U_ORGANIZATION
DIRECTION	ACTION_MOVEMENT	WAR
RELIGION	WORSHIP_PLACE	NATIONALITY_ETHNIC_GROUP
NATIONALITY	YEAR_PERIOD	ROAD
PERSON	RANK	MEDIA_TYPE
WATERFORM	STOCK	SCHOOL
SPORT	REGION	AWARD
MONTH	BODY_OF_WATER	SEISMIC_INTENSITY
GOVERNMENT_AGENCY	ADDRESS	LEGAL_TERM
FIRST_NAME	STAR	GOVERNMENT
COUNTRY	SPORTS_TEAM	TVSERIES
U_LOCATION	PORT	MARKET
WEBSITE	RESTAURANT	COLOR
COMPANY	MILITARY	MALE_FIRSTNAME
THEORY	MEDIA	PLACE
NEWSPAPER	WATERWAY	CATEGORY
DATE_PERIOD	DISTRICT	CAR_STOP
FEMALE_FIRSTNAME	SPORTS_TERM	PROVINCE
SUFFIX	TITLE	INSTITUTE
UNKNOWN	ARTIFACT	AIRPORT
STATION	TITLE_PREFIX	HOLIDAY
ISSUES	ASTRAL_BODY	TUNNEL
DAY	RULE	PLANET
ETHNIC_GROUP	LANDFORM	SEA
MOVIE	ORGANIZATION	NATURAL_DISASTER
MONUMENT	STATE	VEGETABLE
SHIP	SPACESHIP	MUSEUM
AIRCRAFT	CHEMICAL	U_OTHER
BOOK	GAMES	EVENT
BRIDGE	ETHNIC_GROUP_NATIONALITY	POLITICAL_PARTY
CAR	CHARACTER	CRIME
PREFIX	ERA	TIME_PERIOD
WEEK_PERIOD	DAM	UNIVERSITY
U_NAME	AMUSEMENT_PARK	TIME_ZONE
DRUG	ACADEMIC	MAGAZINE
POINTS	CONTINENT	CONFERENCE
CNT	THEATER	DISEASE
COUNTY	OTHER	MUSIC
PLAN	PREFIX_TITLE	CITY
OFFENCE	COMMUNITY_COLLEGE	ANIMAL

Table 3.1: The values of named entity's attribute CATEGORY found in pre-processed LiveJournal corpus

```
`` Jessica 's Journal ''
I need a new default user pic .
The 1 I have right now is really rather annoying .
I like flowers and definitely pink roses .
It 's just boring .
And yes .
I was n't really thinking creatively when I made it.Um .
Also .
Everyone I know aside from my family thinks I 'm a Democrat .
I really have no idea why .
I think I come off as much more Republican -
I mean , I 'm pro - life !
People at church probably assume I 'm Republican just because I go to church though.
I was bored about a week ago and decided to tag all the dreams I have in my memories .
I used to have some pretty messed up dreams , yo .
I still do , but I do n't detail them out anymore .
For various reasons .
Including the 1 that involves the whole not needing to share the fucked up parts of my subconscious.
I 've been cursing a lot lately in my head and online .
Odd .
```

Figure 3.1: A sample post of LiveJournal corpus

3.2 Testing Data

For our testing data we used the English weblogs section of the ACE 2005 Multilingual Training Corpus[2], which contains 119 blog posts. We annotated pairs of documents from this corpus to obtain pairs of posts found by humans to be "similar" or "not similar".

Annotators Two human annotators participated in this task. Both were native English speakers with graduate degrees.

Data Selection Evaluating all possible pairs of the 119 posts in our testing data would take too long, so we first pruned the set of document pairs to pairs that contained at least 25 words in common (not counting stop words like *to, from, a, the*). The resulting set contained 248 document pairs.

Annotation Process We used the web-based annotation tool shown in Figure 3.2. The annotator saw one pair of posts at a time. The posts were shown side by side. Below the posts were three questions that the annotator answered with respect to the post displayed on the right. The questions and their corresponding answer options were:

- *Is it on the same topic as the post on the left?*

  Yes *A little* *No* *I don't know*

[2]Available from the Linguistic Data Consortium; catalog number LDC2006T06

κ	Interpretation
< 0	No agreement
0.0 – 0.20	Slight agreement
0.21 – 0.40	Fair agreement
0.41 – 0.60	Moderate agreement
0.61 – 0.80	Substantial agreement
0.81 – 1.00	Almost perfect agreement

Table 3.2: Kappa Coefficient Interpretation

- *If your answer to the previous question is "Yes" or "A little", is it from the same perspective as the post on the left?*

  Yes   Somewhat   No   I don't know   n/a

- *Do you think the author of the post on the left would be interested in the post on the right?*

  Yes   No   I don't know

After answering the three questions for one pair of blog posts, the annotator would press the "confirm" button. His/her answers would be saved and the next document pair would be shown in the browser.

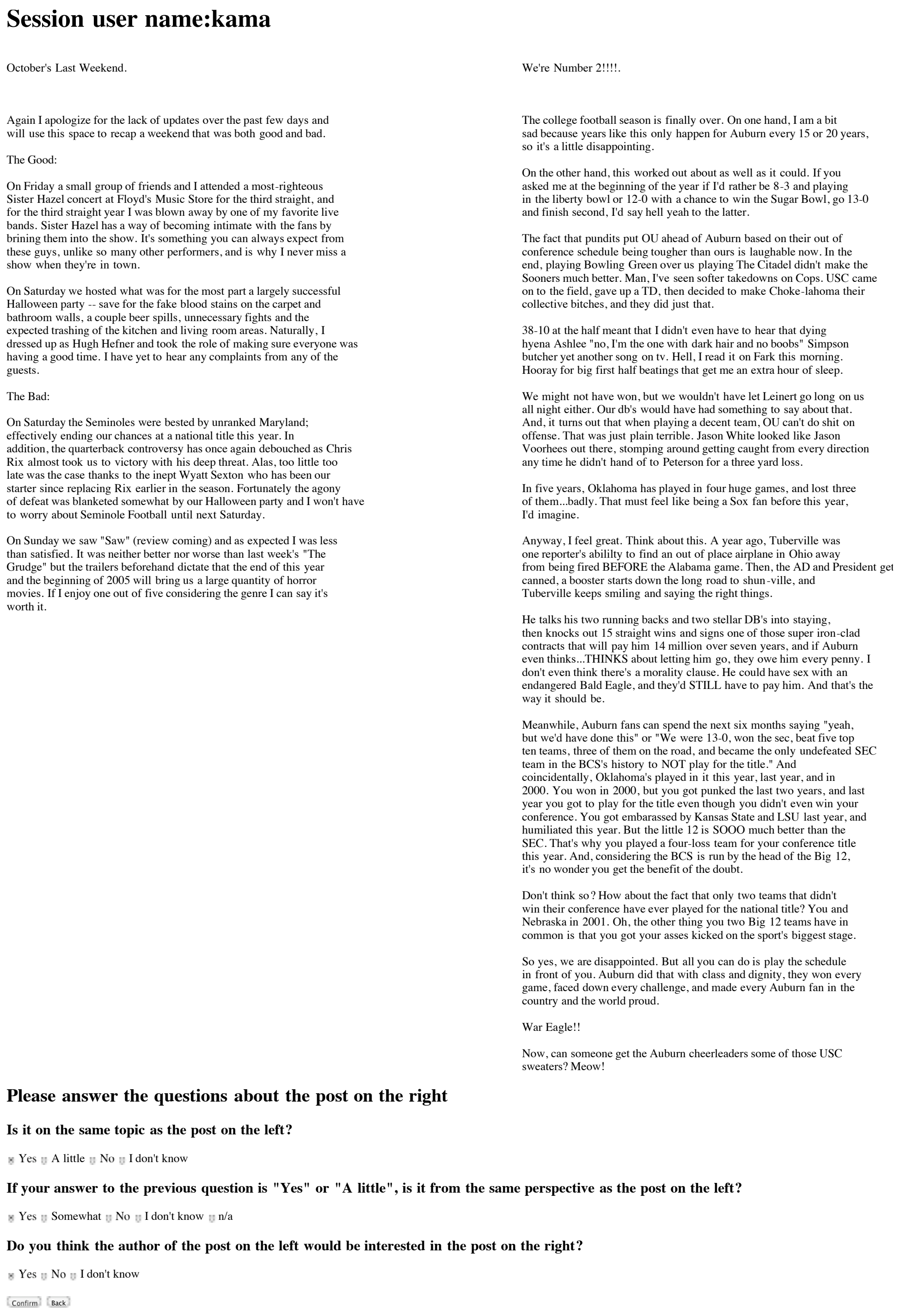
We calculated inter-annotator agreement for this annotation task using Cohen's kappa coefficient (Artstein and Poesio, 2008). Assuming the independence of two annotators, Cohen's kappa coefficient for two annotators is defined as following, where A_o and A_e stand for observed agreement and expected agreement respectively:

$$\kappa = \frac{A_o - A_e}{1 - A_e} \tag{3.1}$$

A_e for each question is inversely proportional to the number of answer options for that question. An interpretation of kappa scores commonly used in Natural Language Processing is shown in Table 3.2.

For the three questions in our annotation task we got $\kappa_1 = 0.83710$, $\kappa_2 = 0.143519$ and

$\kappa_3 = 0.826531$. For questions 1 and 3 we have high agreement, while for question 2 we have only slight agreement. This means that it is easier for the annotators to tell if two posts are on the same topic than it is for them to tell if two blog posts are from the same perspective. As question 1 is the one that is most relevant to our evaluation, we use only the answers to question 1 in the rest of this thesis.



Session user name:kama

October's Last Weekend.

Again I apologize for the lack of updates over the past few days and will use this space to recap a weekend that was both good and bad.

The Good:

On Friday a small group of friends and I attended a most-righteous Sister Hazel concert at Floyd's Music Store for the third straight, and for the third straight year I was blown away by one of my favorite live bands. Sister Hazel has a way of becoming intimate with the fans by brining them into the show. It's something you can always expect from these guys, unlike so many other performers, and is why I never miss a show when they're in town.

On Saturday we hosted what was for the most part a largely successful Halloween party -- save for the fake blood stains on the carpet and bathroom walls, a couple beer spills, unnecessary fights and the expected trashing of the kitchen and living room areas. Naturally, I dressed up as Hugh Hefner and took the role of making sure everyone was having a good time. I have yet to hear any complaints from any of the guests.

The Bad:

On Saturday the Seminoles were bested by unranked Maryland; effectively ending our chances at a national title this year. In addition, the quarterback controversy has once again debouched as Chris Rix almost took us to victory with his deep threat. Alas, too little too late was the case thanks to the inept Wyatt Sexton who has been our starter since replacing Rix earlier in the season. Fortunately the agony of defeat was blanketed somewhat by our Halloween party and I won't have to worry about Seminole Football until next Saturday.

On Sunday we saw "Saw" (review coming) and as expected I was less than satisfied. It was neither better nor worse than last week's "The Grudge" but the trailers beforehand dictate that the end of this year and the beginning of 2005 will bring us a large quantity of horror movies. If I enjoy one out of five considering the genre I can say it's worth it.

We're Number 2!!!!.

The college football season is finally over. On one hand, I am a bit sad because years like this only happen for Auburn every 15 or 20 years, so it's a little disappointing.

On the other hand, this worked out about as well as it could. If you asked me at the beginning of the year if I'd rather be 8-3 and playing in the liberty bowl or 12-0 with a chance to win the Sugar Bowl, go 13-0 and finish second, I'd say hell yeah to the latter.

The fact that pundits put OU ahead of Auburn based on their out of conference schedule being tougher than ours is laughable now. In the end, playing Bowling Green over us playing The Citadel didn't make the Sooners much better. Man, I've seen softer takedowns on Cops. USC came on to the field, gave up a TD, then decided to make Choke-lahoma their collective bitches, and they did just that.

38-10 at the half meant that I didn't even have to hear that dying hyena Ashlee "no, I'm the one with dark hair and no boobs" Simpson butcher yet another song on tv. Hell, I read it on Fark this morning. Hooray for big first half beatings that get me an extra hour of sleep.

We might not have won, but we wouldn't have let Leinert go long on us all night either. Our db's would have had something to say about that. And, it turns out that when playing a decent team, OU can't do shit on offense. That was just plain terrible. Jason White looked like Jason Voorhees out there, stomping around getting caught from every direction any time he didn't hand of to Peterson for a three yard loss.

In five years, Oklahoma has played in four huge games, and lost three of them...badly. That must feel like being a Sox fan before this year, I'd imagine.

Anyway, I feel great. Think about this. A year ago, Tuberville was one reporter's abililty to find an out of place airplane in Ohio away from being fired BEFORE the Alabama game. Then, the AD and President get canned, a booster starts down the long road to shun-ville, and Tuberville keeps smiling and saying the right things.

He talks his two running backs and two stellar DB's into staying, then knocks out 15 straight wins and signs one of those super iron-clad contracts that will pay him 14 million over seven years, and if Auburn even thinks...THINKS about letting him go, they owe him every penny. I don't even think there's a morality clause. He could have sex with an endangered Bald Eagle, and they'd STILL have to pay him. And that's the way it should be.

Meanwhile, Auburn fans can spend the next six months saying "yeah, but we'd have done this" or "We were 13-0, won the sec, beat five top ten teams, three of them on the road, and became the only undefeated SEC team in the BCS's history to NOT play for the title." And coincidentally, Oklahoma's played in it this year, last year, and in 2000. You won in 2000, but you got punked the last two years, and last year you got to play for the title even though you didn't even win your conference. You got embarassed by Kansas State and LSU last year, and humiliated this year. But the little 12 is SOOO much better than the SEC. That's why you played a four-loss team for your conference title this year. And, considering the BCS is run by the head of the Big 12, it's no wonder you get the benefit of the doubt.

Don't think so? How about the fact that only two teams that didn't win their conference have ever played for the national title? You and Nebraska in 2001. Oh, the other thing you two Big 12 teams have in common is that you got your asses kicked on the sport's biggest stage.

So yes, we are disappointed. But all you can do is play the schedule in front of you. Auburn did that with class and dignity, they won every game, faced down every challenge, and made every Auburn fan in the country and the world proud.

War Eagle!!

Now, can someone get the Auburn cheerleaders some of those USC sweaters? Meow!

Please answer the questions about the post on the right

Is it on the same topic as the post on the left?

[✓ Yes] A little No I don't know

If your answer to the previous question is "Yes" or "A little", is it from the same perspective as the post on the left?

[✓ Yes] Somewhat No I don't know n/a

Do you think the author of the post on the left would be interested in the post on the right?

[✓ Yes] No I don't know

Figure 3.2: Screenshot of our web-based annotation tool

Chapter 4

Computing Post Similarity Using LSM Cosine Similarity

4.1 Latent Semantic Mapping

Latent Semantic Mapping (LSM) is a way to find "semantic" similarities between words and documents. Two words are "similar" if they appear with the same or similar other words in a set of training documents (without regard to word order). So for example, *baseball* and *pitcher* are similar, while *baseball* and *orchestra* are probably quite different. Two documents are "similar" if they contain lots of similar words.

Bellegarda (2005) describes Latent Semantic Mapping in detail. The first step is to construct a two-dimensional matrix W of dimensionality $M \times N$, where column j $(< N)$ is a vector for a single document, row i $(< M)$ stands for a single word, and cell i, j contains the frequency (absolute or log frequency) with which word i appears in document j. As most words will appear in only a few documents, W is typically very sparse. As there will be many unique words in any reasonably large collection of documents, W is typically very large.

The second step is to perform singular value decomposition on matrix W. This step decomposes W into three matrices so that:

$$W \approx \hat{W} = USV^T \tag{4.1}$$

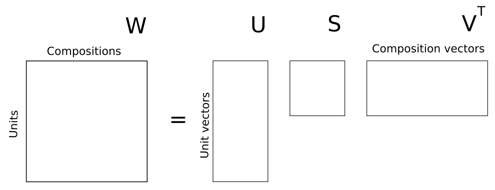


Figure 4.1: Singular Value Decomposition

where U is the $(M \times R)$ left singular matrix with row vectors $u_i (1 \leq i \leq M)$, S is the $(R \times R)$, $R < min(M, N))$ diagonal matrix of singular values $s_1 \geq s_2 \geq ... \geq s_R > 0$ and V is the $(N \times R)$ right singular matrix with row vectors $v_j (1 \leq j \leq N)$. $R < min(M, N)$ is the order of the decomposition. Both of left and right matrices U and V are column orthonormal, i.e.,

$$U^T U = V^T V = I_R$$

where I_R is the identity matrix of order R. Thus the column vectors of U and V are column orthonormal basis for the vector space of dimension R spanned by the vector u_i and v_j, referred as the LSM vector space.

In this vector space, unit-unit comparisons(word closeness), composition-composition comparisons(document closeness), and unit-composition comparisons(word-document closeness) can be measured by cosine similarity of a pair of vectors(word-word, document-document or word-document, respectively), which reflects the latent semantic closeness between a pair of words, a pair of documents, and a word in the context of the document.

4.2 Building Matrices

When computing document similarity, we hypothesize that some word sequences should be *lexicalized*, or treated as if they are one word. In particular, word sequences that are names (e.g. *New York, Dr. Sarah Jones, Stony Brook University*) should be treated as single words. Also, we hypothesize that some word types are more important for computing document similarity than others: in particular, named entities and verbs are important, since they indicate the types of actions and objects in a document. So we construct three different matrices for our LSM experiment: a matrix whose cells represent document-word frequencies, a matrix whose cells represent either document-word frequencies (for words not part of named entities) or document-entity frequencies, and a matrix whose cells represent either document-verb frequencies or document-entity frequencies. In this section we describe the process of building these matrices.

4.2.1 Pre-processing

First, we pre-processed our training data as follows:

- To construct the input to building the word-based matrix we extracted the text of each blog post. We segmented the text into tokens, removing punctuation. A "token" in this case was a word, a sequence of non-space characters with a space on each side. We lowercased every token and stemmed it (using the Porter Stemmer (van Rijsbergen, Robertson, and Porter, 1980)). Stemming reduces a word to its root: for example, *girls* becomes girl, *sadly* becomes *sad* and *tricked* becomes *trick*. We then removed stop words (e.g. *a, the, from, in*). Finally, we saved the plain text of the post (without part-of-speech or named-entity tags and without the XML markup that appeared in the original post). For the input sentence *Harry Potter flew around London on a broomstick*, the saved sequence of tokens from this process would be {*harry, potter, fly, london, broomstick*}.

- To construct the input to building the entity-non entity matrix we extracted the text of each blog post. We again segmented the text into tokens, removing punctuation. However, in this case a "token" could be a named entity (a sequence of words marked by Lydia as a name, with the category of the name) or a word. We lowercase and stemmed all the tokens as before, and saved the resulting plain text. The list of named entity categories labeled by Lydia is given in Table 4.1. For the input sentence *Harry Potter flew around London on a broomstick*, the saved sequence of tokens from this process would be {*harry_potter, fly, london, broomstick*}.

- To construct the input to building the verb-entity matrix, we extracted the text of each blog post. We processed the text as for the entity-non entity matrix, but removed all tokens

LAST_NAME	PARK	MONTH_PERIOD
RAILROAD	U_ORGANIZATION	WAR
RELIGION	WORSHIP_PLACE	PORT
NATIONALITY	YEAR_PERIOD	ROAD
PERSON	MEDIA_TYPE	WATERFORM
SCHOOL	BODY_OF_WATER	REGION
MONTH	SPORT	GOVERNMENT_AGENCY
ADDRESS	FIRST_NAME	GOVERNMENT
COUNTRY	SPORT_TEAM	U_LOCATION
NATIONALITY_ETHNIC_GROUP	MARKET	WEBSITE
RESTAURANT	COMPANY	MILITARY
MALE_FIRSTNAME	MEDIA	PLACE
NEWSPAPER	WATERWAY	CATEGORY
DATE_PERIOD	DISTRICT	CAR_STOP
FEMALE_FIRSTNAME	PROVINCE	TITLE
INSTITUTE	AIRPORT	STATION
HOLIDAY	TUNNEL	DAY
ETHNIC_GROUP	LANDFORM	SEA
ORGANIZATION	MONUMENT	STATE
MUSEUM	EVENT	BRIDGE
POLITICAL_PARTY	ERA	TIME_PERIOD
WEEK_PERIOD	DAM	UNIVERSITY
AMUSEMENT_PARK	TIME_ZONE	ACADEMIC
MAGAZINE	CONTINENT	CONFERENCE
ETHNIC_GROUP_NATIONALITY	COUNTY	CITY
COMMUNITY_COLLEGE	THEATER	

Table 4.1: The values of attribute CATEGORY used to extract name entities

that were not either a named entity or a verb[1]. For the input sentence *Harry Potter flew around London on a broomstick*, the saved sequence of tokens from this process would be {*harry_potter, fly, london*}.

4.2.2 Matrix construction

We first tried to build the word-based matrix, the entity-non entity matrix and the verb-entity matrix using jLSI (Giuliano, 2007). However, jLSI keeps the whole matrix under construction in memory; it also uses a dense matrix format rather than a sparse matrix format. So we kept running out of

[1]We said a *verb* was a word part-of-speech tagged as VB (verb, base form), VBD (verb, past tense), VBG (verb, present participle),VBN (verb, past participle), VBP (verb, present tense, not 3rd person singular) or VBZ (verb, present tense, 3rd person singular).

memory. To overcome this problem, we implemented our own matrix builder in C. Our matrix builder uses two steps: first, it uses the document set to build a dictionary containing word-index mappings. Then, it uses the dictionary and document set to build the document vector for each document.

Dictionary construction There are two main reasons for us to build dictionaries before building the matrix:

1. We decided to store the matrices in a binary sparse-matrix format, as LSM matrices are mostly very sparse. The binary sparse-matrix format stores only non-zero elements of the matrix.
2. For efficient matrix construction we would like the dictionary to be sorted so that finding the row index for a word is quick. Although this sorting can be done by inserting a word into the dictionary while constructing document vectors, this requires either repeated linear searches through the dictionary or repeated modification of the document vectors previously constructed. As our program adds a document vector by directly writing it into a matrix file (to reduce main memory usage), the performance penalty for simultaneously constructing the dictionary and matrix would be very high because of the bottleneck of disk I/O.

When the dictionaries were built, we realized that our vocabulary size was unreasonably large, mainly due to fake words (either from markup, from spam posts, or from blogger exclamations). The dictionary for the word-based matrix had 5029848 terms, the dictionary for the entity-non entity matrix had 5697269 terms, and the dictionary for the verb-entity matrix had 1300772 terms. We wanted to keep the vocabulary size at no more than 50000 terms, so we removed terms from the dictionaries using the following heuristics:

1. Valid terms can only be composed of alpha-numeric characters
2. No valid term can be composed of numeric characters only
3. No valid term can contain a repeated substring of length greater than or equal to 2. For example, *haha* is not a valid term.

To further restrict the vocabulary size, we also eliminated terms with relatively low occurrence in the entire corpus. For the dictionary for the word-based matrix we eliminated terms that occurred fewer than 85 times. For the dictionary for the entity-non entity matrix we eliminated terms that occurred fewer than 82 ties. For the dictionary for the verb-entity matrix we eliminated terms that occurred fewer than 9 times.

As a result of these changes, we ended up with the following dictionary sizes: for the word-based matrix, 49958 terms; for the entity-non entity matrix, 49869 terms; and for the verb-entity matrix, 49047 terms.

Document vector construction Our matrix construction program first reads a dictionary and loads it into memroy as a hashtable. Then it reads builds the matrix one document vector at a time as folows:

1. While there are terms in the document:

   Read the next term from the document.

   If the term is in the dictionary, get the index of the term. If the term is already in the document vector, add 1 to its occurrence. If not, add it to the document vector and set its occurrence to 1.

   It the term is not in the dictionary, then do not augment the document vector.

2. Finally, change all frequencies in the document to log frequencies using the formula

$$1 + \log(tf)$$

   where tf denotes term frequency. This helps to reduce later computation effort.

3. Append the document vector to the matrix and release immediately the memory space occupied by the document vector. This is to minimize memory consumption.

The matrix-building process finishes when all documents in the corpus are included in the matrix file. We followed this procedure for the word-based matrix, the entity-non entity matrix and the

```
numRows numCols totalNonZeroValues
for each column:
  numNonZeroValues
    for each non-zero value in the column:
        rowIndex value

All values are 4-byte integers except value, which is a
4-byte float. All are in network byte order.
```

Figure 4.2: Binary sparse-matrix format description

verb-entity matrix.

4.2.3 Singular Value Decomposition

Finally, we performed Singular Value Decomposition(SVD) on each matrix. We used Doug Rohde's SVD C Library (Rohde, 2009). The main parameter for SVD is the dimensionality. We limited the dimensionality to 50, which is the minimum value suggested in (Landauer, Foltz, and Laham, 1998). We ran SVD on all of our matrices to get their corresponding diagonal S matrix, transposed V matrix and transposed U matrix. Now we are ready to compute cosine similarity using these matrices.

4.3 LSM Cosine Similarity

Our goal in this experiment is to test whether the composition-composition cosine similarities between a pair of documents reflects the actual human judgments of document similarity. We applied the LSM Framework Extension described in (Bellegarda, 2005; Mei and Zhai, 2001) to our annotated ACE document pair corpus. Suppose our training data has M words and N posts, then the dimension of matrix W, U, S and V^T in equation (4.1) is $(M \times N)$, $(M \times R)$, $(R \times R)$ and $(R \times N)$, respectively.

Now suppose we have a new post d with a term frequency vector p of dimension M. Equation (4.1) implies

$$p = USv^T \tag{4.2}$$

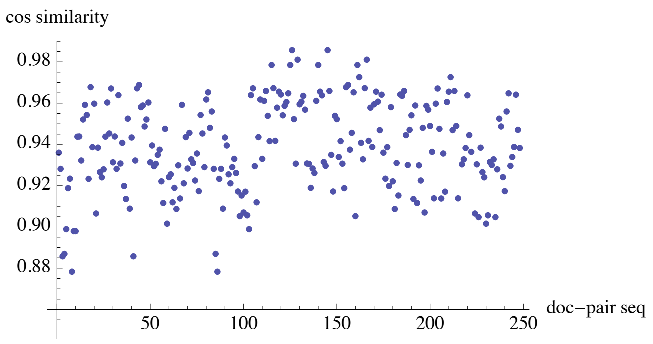


Figure 4.3: LSM cosine similarities of annotated document pairs using word-based matrix

where the R-dimensional vector v^T acts as an additional column of the matrix V^T. This leads to the definition

$$v = vS = p^T U \tag{4.3}$$

We constructed this *composition vector* v for each post in our annotated post pairs.

We then computed cosine similarity between the composition vectors for each post pair. Suppose there is an annotated post pair (p, q). Let v_p be the composition vector of post p and v_q be the composition vector of post q. Thus the composition-composition cosine similarity between posts p and q is defined as:

$$K(p, q) = cos(v_p S, v_q S) = \frac{v_p S^2 v_q^T}{\|v_p S\| \|v_q S\|} \tag{4.4}$$

Essentially, we passed the posts in each post pair through the reduced-dimensionality matrix, and then computed similarity between the transformed posts.

The cosine similarities for our annotated post pairs using the word-based matrix, entity-non entity matrix and verb-entity matrix are shown in Figure 4.3, Figure 4.4, and Figure 4.5.

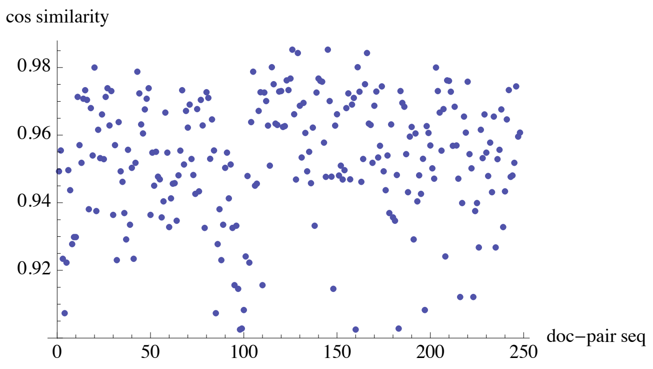


Figure 4.4: LSM cosine similarities of annotated document pairs using entity-non entity matrix

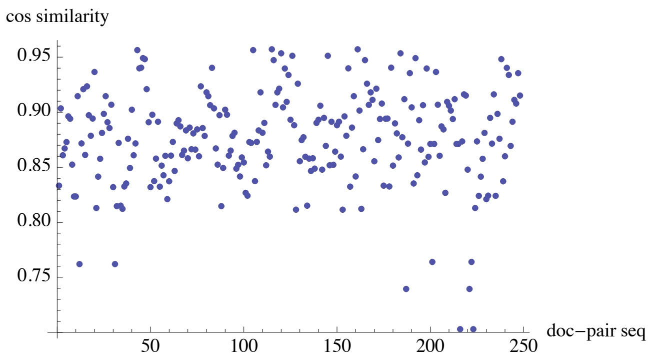


Figure 4.5: LSM cosine similarities of annotated document pairs using verb-entity matrix

Chapter 5

Computing Post Similarity Using Simple Cosine Similarity

We also calculated cosine similarity between the annotated post pairs in our testing data directly (without "passing through" the matrix). This is a widely used method for computing document similarity; however, we thought it would be less accurate than our other method because there is no "latent semantics".

This method of calculating document similarity is simple and efficient. For a post pair d_p and d_q, there are two term frequency vectors p and q based on a dictionary containing terms from d_p and d_q. For this approach as for our first approach, we mapped all term frequencies to log space by converting every non-zero term frequency tf to $1 + log(tf)$. Once these two term frequency vectors are constructed, we compute cosine similarity:

$$K(p, q) = cos(p, q) = \frac{p \cdot q^T}{||p|| \, ||q||} \tag{5.1}$$

For each post pair, we constructed word-based, entity-non entity, and verb-entity term frequency vector pairs and computed cosine similarity between the vectors in each pair. Word-based term frequency vectors used lowercased, stemmed words from each post. Entity-non entity term frequency vectors used words and named entities from the hand-annotated ACE corpus files (ACE corpus files

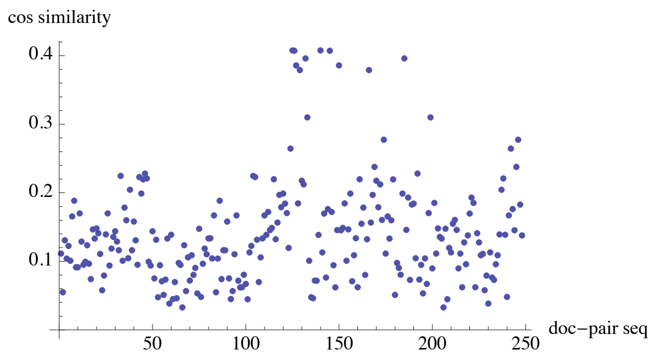


Figure 5.1: Simple cosine similarities of annotated document pairs by word

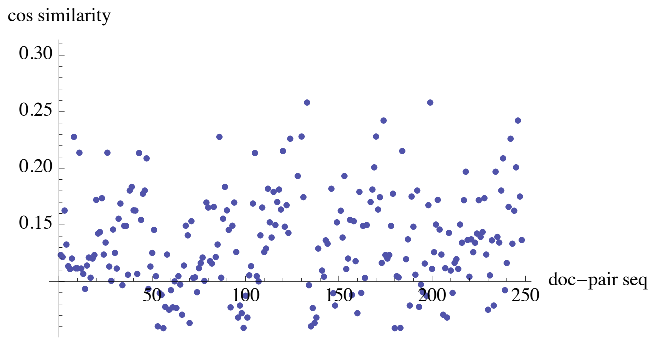


Figure 5.2: Simple cosine similarities of annotated document pairs by entity-non entity

are labeled for named entities). The words were stemmed, and all tokens were lowercased. Verb-entity term frequency vectors used the hand-annotated ACE corpus files and versions of the plain text files part-of-speech tagged using the MXPOST tagger (Ratnaparkhi, 1996). Once again, the verbs were stemmed and all tokens were lowercased. Finally, all stopwords were filtered out of each document before term frequency vector construction.

The cosine similarities for our annotated post pairs are shown in Figure 5.1, Figure 5.2 and Figure 5.3.

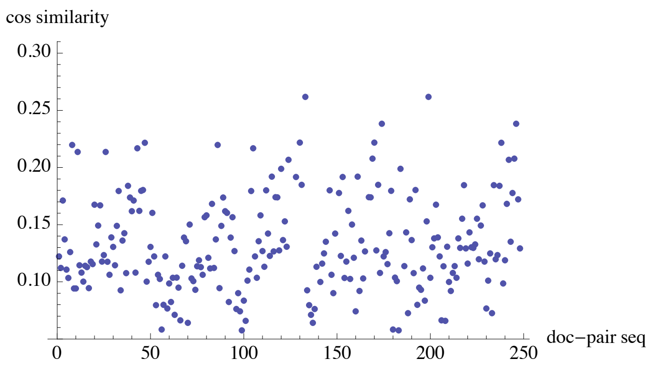


Figure 5.3: Simple cosine similarities of annotated document pairs by verb-entity

Chapter 6

Evaluation

We want to know how our approaches to computing document similarity compare to human judgments of document similarity. We separated our testing post pairs into three groups:

- **No - No** – post pairs for which both human annotators said the posts were not on similar topics (both annotators answered "No" to question 1)
- **No - !No** – post pairs for which exact one human annotator said the posts were on similar topics
- **!No - !No** – post pairs for which both human annotators said the posts were on similar topics

6.1 Exploratory Analysis

We started by computing the mean and standard deviation of document similarities for each approach over our testing data. These are shown in Table 6.1 and Table 6.2.

We observed that all three straight cosine similarity approaches showed an interesting tendency: the mean cosine similarity for No - No is smaller than the mean cosine similarity for No - !No, which again is smaller than the mean cosine similarity for !No - !No. That is

$$\mu_{No-No} < \mu_{No-!No} < \mu_{!No-!No}$$

	LSM cos			**Straight cos**		
	No - No	**No - !No**	**!No - !No**	**No - No**	**No - !No**	**!No - !No**
Word	0.939212	0.917143	0.941667	0.130690	0.139524	0.226667
Entity Nonentity	0.953695	0.950000	0.955417	0.130443	0.175714	0.245417
Verb Entity	0.879015	0.857143	0.875000	0.128473	0.166667	0.231667

Table 6.1: Means of LSM cosine and simple cosine similarity, grouped by human judgments No - No, No - !No, !No - !No

	LSM cos			**Straight cos**		
	No - No	**No - !No**	**!No - !No**	**No - No**	**No - !No**	**!No - !No**
Word	0.020496	0.057582	0.032102	0.056442	0.097857	0.127105
Entity Nonentity	0.017353	0.019760	0.024149	0.041892	0.082032	0.117153
Verb Entity	0.042217	0.059132	0.040927	0.040987	0.076428	0.098939

Table 6.2: Standard deviations of LSM cosine and simple cosine similarity, grouped by human judgments No - No, No - !No, !No - !No

So we went a step further, to see if these differences were statistically significant.

6.2 Statistical Significance

We considered using Analysis of variance (ANOVA), but unfortunately our cosine similarities are not normally distributed, so ANOVA is not applicable (see Figure 6.1, Figure 6.2, Figure 6.3, Figure 6.4, Figure 6.5 and Figure 6.6).

Instead, we used the Kruskal-Wallis rank sum test to compare cosine similarity scores of our different approaches (Boslaugh and Watters, 2008). This is a non-parametric test for repeated measures, which fits our data well. We found significant differences in cosine similarities across the six approaches ($p < 0.01$). So we ran post-hoc Wilcoxon rank sum tests to compare pairs of approaches (Boslaugh and Watters, 2008). We found significant differences between all pairs of approaches (at $p < 0.01$) *except* for:

- Entity-non entity straight cos and verb-entity straight cos

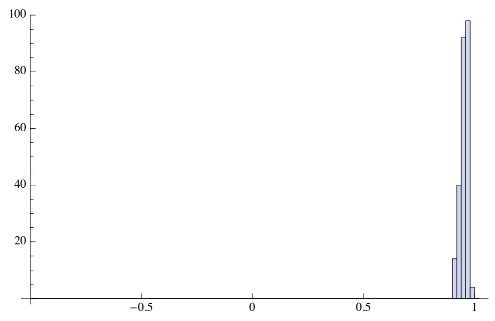


Figure 6.1: Entity non-entity LSM cos score histogram

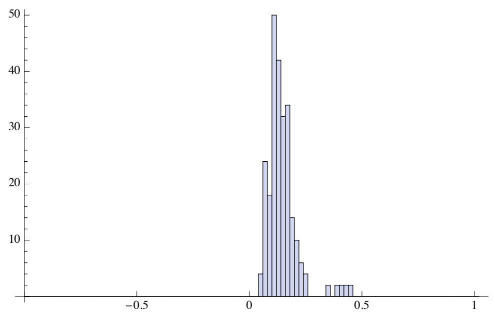


Figure 6.2: Entity non-entity straight cos score histogram

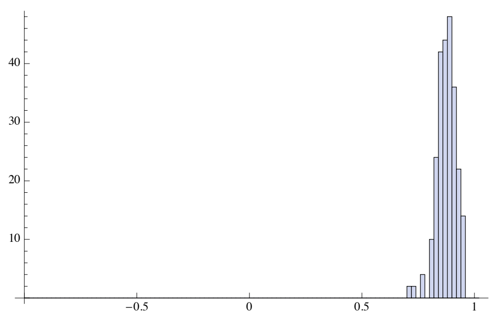


Figure 6.3: Verb-entity LSM cos score histogram

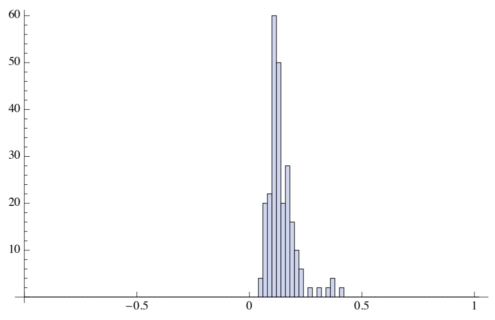


Figure 6.4: Verb-entity straight cos score histogram

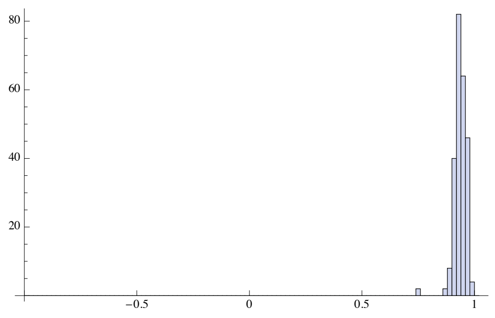


Figure 6.5: Word LSM cos score histogram

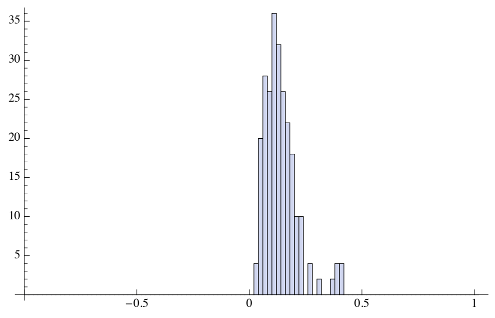


Figure 6.6: Word straight cos histogram

Dataset pair	**p value of Wilcoxon's rank sum test**
Entity non-entity LSM cos, Entity non-entity straight cos	0
Entity non-entity LSM cos, Verb-entity LSM cos	0
Entity non-entity LSM cos, Verb-entity straight cos	0
Entity non-entity LSM cos, Word LSM cos	1.154632×10^{-14}
Entity non-entity LSM cos, Word straight cos	0
Entity non-entity straight cos, Verb-entity LSM cos	1.003104×10^{-82}
Entity non-entity straight cos, Verb-entity straight cos	0.5477166
Entity non-entity straight cos, Word LSM cos	1.00310×10^{-82}
Entity non-entity straight cos, Word straight cos	0.1327274
Verb-entity LSM cos, Verb-entity straight cos	0
Verb-entity LSM cos, Word LSM cos	2.047253×10^{-53}
Verb-entity LSM cos, Word straight cos	0
Verb-entity straight cos, Word LSM cos	1.003104×10^{-82}
Verb-entity straight cos, Word straight cos	1.003104×10^{-82}
Word LSM cos, Word straight cos	0

Table 6.3: The results of Wilcoxon's rank sum test

- Entity-non entity straight cos and word straight cos

There were some interesting observations from the result:

1. Any LSM cos has significant difference to any straight cos
2. Entity non-entity straight cos has no significant different with either verb-entity straight cos or word straight cos
3. Verb-entity straight cos has significant difference to word straight cos

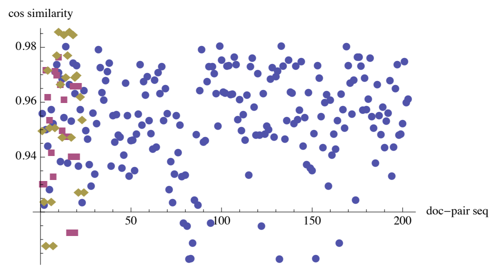


Figure 6.7: Entity non-entity LSM cos score, grouped by human annotation categories.(dot: no-no; box: no-not no; diamond: not no-not no)

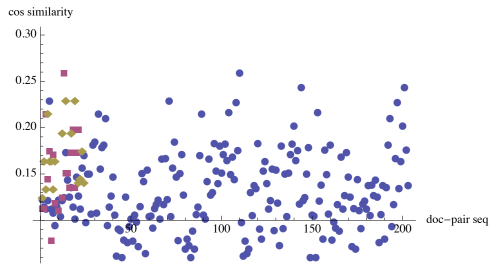


Figure 6.8: Entity non-entity straight cos score, grouped by human annotation categories(dot: no-no; box: no-not no; diamond: not no-not no)

We plotted the cosine similarity values (Figure 6.7, Figure 6.8, Figure 6.9, Figure 6.10, Figure 6.11 and Figure 6.12) and the corresponding Box-Whisker plots (Figure 6.13, Figure 6.14, Figure 6.15, Figure 6.16, Figure 6.17 and Figure 6.18) from our experiment results based on human annotation categories (No - No, No - !No, and !No - !No).

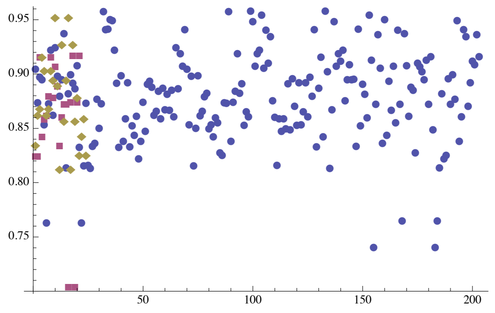


Figure 6.9: Verb-entity LSM cos score, grouped by human annotation categories(dot: no-no; box: no-not no; diamond: not no-not no)

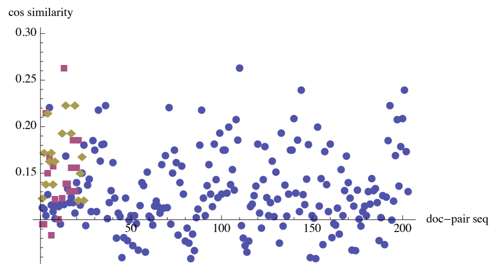


Figure 6.10: Verb-entity straight cos score, grouped by human annotation categories(dot: no-no; box: no-not no; diamond: not no-not no)

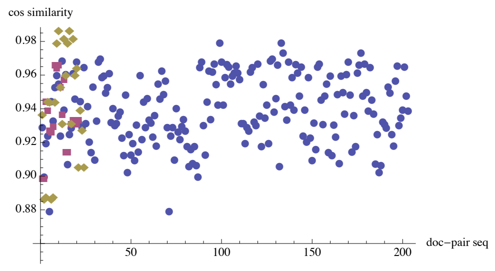


Figure 6.11: Word LSM cos score, grouped by human annotation categories(dot: no-no; box: no-not no; diamond: not no-not no)

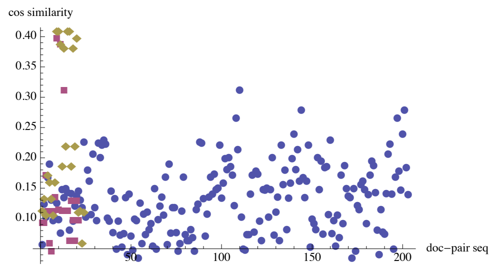


Figure 6.12: Word straight cos score, grouped by human annotation categories(dot: no-no; box: no-not no; diamond: not no-not no)

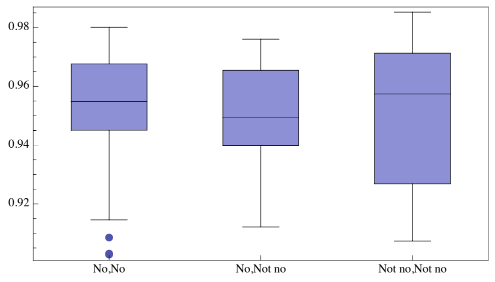


Figure 6.13: The box plot of entity non-entity LSM cos, based on human annotation categories

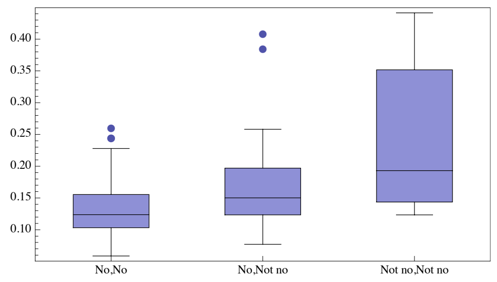


Figure 6.14: The box plot of entity non-entity straight cos score, based on human annotation categories

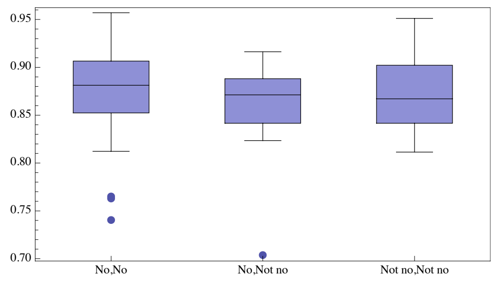


Figure 6.15: The box plot of verb-entity LSM cos score , based on human annotation categories

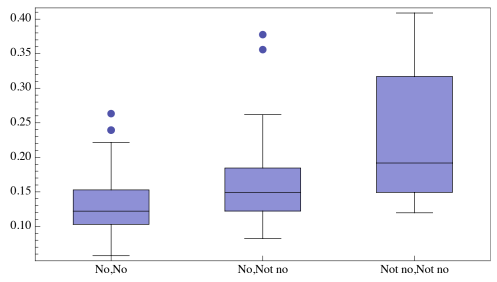


Figure 6.16: The box plot of verb-entity straight cos score, based on human annotation categories

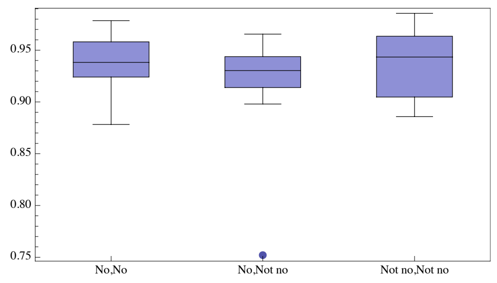


Figure 6.17: The box plot of word LSM cos score, based on human annotation categories

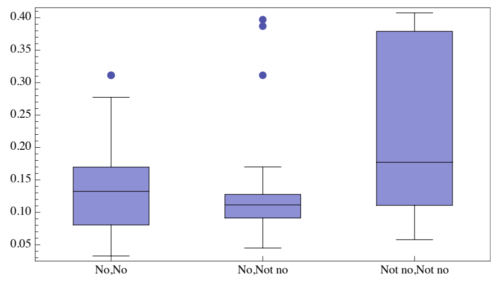


Figure 6.18: The box plot of word straight cos score, based on human annotation categories

6.3 Finding Cutoff Levels for Document Similarity

Finally, we tried to use the annotated data to identify cutoff cosine similarity levels for computing document similarity for each approach. As before, we split the test data into three categories, No - No, No - !No, and !No - !No.

Next we split our test data at random into five partitions of equal size.

Then, we ran a five-round experiment. In each round we took one partition as testing data and the remaining four partitions as training data. We used the training data to calculate the cutoff cosine similarity numbers that would give the highest proportion of correct assignments to test data categories.

Finally, we used these cutoff numbers to assign post pairs in the testing partition to test data categories. We then computed the fraction of these assignments that were correct. Our results are shown in Table 6.4.

Round	**Entity-non entity LSM**	**Entity-non entity simple**	**Verb-entity LSM**	**Verb-entity simple**	**Word LSM**	**Word simple**
1	0.163265	0.755102	0.510204	0.714186	0.367347	0.367347
2	0.163265	0.755102	0.489796	0.653061	0.244898	0.489796
3	0.448980	0.612245	0.448980	0.591837	0.224490	0.591837
4	0.163265	0.510204	0.489796	0.714286	0.244898	0.469388
5	0.153846	0.692308	0.519231	0.673077	0.230769	0.557692
average accuracy	0.218524	0.664992	0.491601	0.669309	0.262480	0.495212

Table 6.4: Accuracy of each round and the average accuracy

From these results we observe:

1. All simple cosine approaches perform better than LSM cosine approaches.

2. Among the simple cosine approaches, the verb-entity approach performs better than the others.

We can conclude that there is some preliminary evidence that the verb-entity simple cosine approach is more accurate than the others at determining topic similarity. However, it cannot determine similarity of sentiment. Further research needs to be done to identify an automatic approach for topic and sentiment similarity.

Chapter 7

NLP Pipeline Evaluation

In our post similarity experiments, we used data that was labeled (automatically or by hand) with part-of-speech tags and named entities. We would like to add to PLOG a Natural Language Processing pipeline that can automatically perform these and other labeling tasks. So we evaluated existing systems that we may use to assemble a pipeline for PLOG. We started with three systems

- LingPipe (Alias-i, 2008)
- OpenNLP (OpenNLP, 2009)
- Mallet (McCallum, 2002)

All three are freely-available systems that can segment text into sentences, assign part-of-speech tags, perform named entity annotation, and do other Natural Language Processing tasks. Also, all three are trainable to some extent so they can be adapted to a particular kind of language.

Mallet uses Conditional Random Fields to train its models (Mann and McCallum, 2007; Tsai, Wu, and Hsu, 2005; Deerwester et al., 1990). We found that this made it very slow. For example, we were not able to finish training Mallet to do part-of-speech tagging given the Penn Treebank Wall Street Journal corpus within a day. Furthermore, Mallet ran out of memory heap while training. So we excluded it from consideration.

7.1 Method

We evaluated the two remaining systems on each task for which they have a component. Most of the tasks are classification tasks, so we report classification accuracy.

The data we used for our evaluation is mainly the Penn Treebank Wall Street Journal corpus[1], but we used the CoNLL-2000 corpus[2] to evaluate the systems' performance on chunking text (finding basic phrases), the ACE 2005 Multilingual Training Corpus[3] to evaluate named-entity tagging and a fourNewsGroups corpus available from LingPipe's source code archive[4] to evaluate topic classification.

For each component, we ran an evaluation using five-fold cross validation. That means we split our data at random into five partitions of roughly equal size. We then ran five rounds of training/testing. In each round, one partition was used for testing and the other five for training. We averaged classification accuracy and speed (time to train and test) across the five rounds.

7.2 Results

Average accuracy and average speed of each system for each task are shown in Table 7.1. We observe that OpenNLP can do one task that LingPipe cannot (chunking). For two tasks (sentence segmentation and named entity tagging) it is more accurate than LingPipe but slower. For one task (part-of-speech tagging) it is about as accurate as LingPipe (but still slower). LingPipe can do two tasks that OpenNLP cannot (interesting phrase detection and topic classification): however, there is no testing data for interesting phrase detection, and topic detection is not particularly useful for PLOG since in PLOG we cannot write down a closed set of topics a priori.

We decided to adopt OpenNLP to be our NLP pipeline because:

- OpenNLP has shown higher accuracy on sentence segmentation, POS tagging and named entity tagging.

[1]Available from the Linguistic Data Consortium; catalog number LDC99T42
[2]Available from the website of the Conference on Computational Natural Language Learning 2000
[3]Available from the Linguistic Data Consortium; catalog number LDC2006T06
[4]demos/fourNewsGroups/

	LingPipe		**OpenNLP**	
	Accuracy	**Speed**	**Accuracy**	**Speed**
Sentence segmentation	0.5467128%	11886 msec	0.7792478%	11266 msec
Part-of-speech tagging	0.9638384%	29212 msec	0.9657274%	1982865 msec
NP/VP chunking	n/a	n/a	0.9249919%	47567 msec
Named entity tagging	0.7332214%	2624 msec	0.9117132%	14949 msec
Interesting phrase detection	n/a	13303 msec	n/a	n/a
Topic classification	0.9861111%	8191 msec	n/a	n/a

Table 7.1: Performance of different NLP toolkits

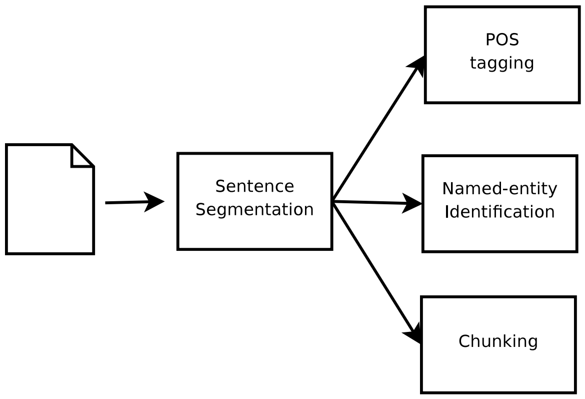


Figure 7.1: NLP pipeline

- Sentence segmentation of LingPipe is not trainable.
- The effort to conduct training for OpenNLP is less than LingPipe. OpenNLP has already well-written training programs. What users should do is only feed training data in the indicated format.

Our proposed NLP pipeline to process a PLOG post is shown in Figure 7.1. Note that the input post is assumed to be in plaintext format.

Chapter 8

Conclusions and Future Work

The contributions of this thesis include:

- A comparison of LSM-based cosine and simple cosine methods for computing topic similarity between document pairs
- A comparison of word-based, entity-non entity and verb-entity methods for computing topic similarity between document pairs
- A comparison of several NLP pipelines for preliminary processing of documents

From our experiment results, we suggest using simple cosine similarity may be more favorable than using LSM-based cosine similarity:

1. Simple cosine requires significantly less computation effort than LSM-based cosine.
2. Simple cosine may be significantly more accurate than LSM-based cosine for this task (if the LSM-based cosine training data is large).

We found that using simple cosine we can achieve moderate accuracy on identifying topic similarity, which is very essential to PLOG. We hope this result would bring PLOG into practical implementation.

Moreover, among the straight cosine similarities, the verb-entity and entity-non entity methods have the highest accuracy in topic similarity identification.

Finally, we suggest the use of OpenNLP for preliminary document processing as it is easily trainable and achieves high accuracy on several NLP tasks important for PLOG (even though it can be slow).

In future work, we think it is worthwhile to continue experimenting with more sophisticated ways to compute topic similarity between document pairs, and with ways to compute topic and sentiment similarity. Also, we need to do an evaluation of our methods in PLOG, to see if they are useful for PLOG users.

Bibliography

Alias-i. 2008. Lingpipe. http://alias-i.com/lingpipe/.

Artstein, R., and Poesio, M. 2008. Inter-coder agreement for computational linguistics. *Computational Linguistics* 34(4).

Banerjee, S.; Ramanathan, K.; and Gupta, A. 2007. Clustering short texts using Wikipedia. In *Proceedings of the international ACM conference on Research and development in information retrieval (SIGIR)*, 787–788.

Bellegarda, J. R. 2005. Latent semantic mapping. *IEEE Signal Processing Magazine* 22(5):70.

Boslaugh, S., and Watters, P. A. 2008. *Statistics in a Nutshell.* O'Reilly Media.

Deerwester, S.; Dumais, S.; Furnas, G.; Landauer, T.; and Harshman, R. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41:391–407.

Doran, W.; Stokes, N.; Carty, J.; and Dunnion, J. 2004. Assessing the impact of lexical chain scoring methods and sentence extraction schemes on summarization. *Lecture Notes in Computer Science* 2945/2004:627–635.

Giuliano, C. 2007. jLSI a for latent semantic indexing. Software available at http://tcc.itc.it/research/textec/tools-resources/jLSI.html.

Hart, M.; Johnson, R.; and Stent, A. 2007. Content-based access control. In *Proceedings of the IEEE Symposium on Privacy and Security*.

Hayes, C., and Avesani, P. 2007. Using tags and clustering to identify topic-relevant blogs. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*.

Kil, J. H.; Lloyd, L.; and Skiena, S. 2005. Question answering with Lydia. In *Proceedings of the Text Retrieval and Extraction Corpus (TREC) 2005*.

Landauer, T. K.; Foltz, P. W.; and Laham, D. 1998. An introduction to latent semantic analysis. *Discourse Processes* 25:259–284.

Lloyd, L.; Kechagias, D.; and Skiena, S. 2005. Lydia: A system for large-scale news analysis. *Proceedings of the International Conference on String Processing and Information Retrieval (SPIRE)* 161–166.

Mann, G., and McCallum, A. 2007. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Proceedings of the Meeting of the North American chapter of the Association for Computational Linguistics (NAACL)*.

McCallum, A. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Mei, Q., and Zhai, C. 2001. A note on EM algorithm for probabilistic latent semantic analysis. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*.

Mei, Q.; Ling, X.; Wondra, M.; Su, H.; and Zhai, C. 2007. Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the International World Wide Web Conference (WWW)*, 171–180.

Mishne, G. 2006. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *Proceedings of the International World Wide Web Conference (WWW)*, 953–954.

Nallapati, R., and Cohen, W. 2008. Link-PLSA-LDA: A new unsupervised model for topics and influence of blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*.

OpenNLP. 2009. Opennlp. http://opennlp.sourceforge.net/.

Ratnaparkhi, A. 1996. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 133–142.

Rohde, D. 2009. Doug Rohde's SVD C library. http://tedlab.mit.edu/ dr/svdlibc.

Schutze, H., and Manning, C. D. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Silber, H., and McCoy, K. 2002. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics* 28(4):487.

Sood, S. C., and Hammond, K. J. 2007. TagAssist: Automatic tag suggestion for blog posts.

Tsai, T.-H.; Wu, S.-H.; and Hsu, W.-L. 2005. Exploitation of linguistic features using a CRF-Based biomedical named entity recognizer. In *Proceedings of the ACL Workshop on Linking Biological Literature, Ontologies and Databases: Mining Biological Semantics*.

van Rijsbergen, C.; Robertson, S.; and Porter, M. 1980. New models in probabilistic information retrieval. Technical report, British Library. British Library Research and Development Report, no. 5587.