Proceedings of the 5th WSEAS International Conference on Applied Computer Science, Hangzhou, China, April 16-18, 2006 (pp573-578)

Augmentation of a Term/Document Matrix with Part-of-Speech Tags to Improve Accuracy of Latent Semantic Analysis

TOM RISHEL, A. LOUISE PERKINS, SUMANTH YENDURI, FARNAZ ZAND
Computer Science Dept.,
University of Southern Mississippi
730 East Beach Blvd, Long Beach, MS 39560
USA

S.S. IYENGAR
Computer Science Dept.,
Louisiana State University
298 Coates Hall, Tower Drive, Baton Rouge, LA 70802
USA

Abstract: - We consider the improvement in accuracy of latent semantic analysis when a part of speech tagger is used to augment a term/document matrix. We first construct an augmented term/document matrix as input into singular value decomposition (SVD). The singular values then serve as principal components for a cosine projection. The results show that the addition of POS tags can decrease ambiguities significantly.

Key-Words: - Latent Semantic Analysis, Documents, Tags, Singular Value Decomposition

1 Introduction

Latent semantic analysis (LSA) has been used for several years to improve the performance of document library searches. For example, [4] showed that a SVD based projection, also referred to as latent semantic analysis or latent semantic indexing, improved document library searches. LSA uses scaled word frequencies across a set of candidate documents, which we refer to as a library, to discover and quantify semantic similarities among documents and, in so doing, to locate documents that are most similar to the original query document. We use Brill's POS tagging software [1, 2, 3] in conjunction with the Infomap natural language processing software [7] to generate a POS tagged term/document matrix.

POS taggers have only recently been used to augment a term/document matrix as input for LSA [8]. In that paper they used very brief documents and queries that are sentence-length or shorter. They report a decrease in accuracy when using a POS-augmented word/document matrix. In this paper we present a study that improves the accuracy, contrary to their results. We believe that part of the reason for their negative result is due directly to their short document lengths. Another issue may be that the granularity level of the POS tagger can affect query results by posting both false positive and false negative results.

The procedure we use requires three major steps. First, we format two libraries of identical documents, one POS tagged, the other untagged. Next we formulate a tagged query and an untagged query. Then we query the tagged library with the tagged term and the untagged library with the untagged term.

To tag our texts, we used Brill's automatic part-of-speech tagger. It assigns tags that indicate the part-of-speech of a word within the context provided. The tagger we use was trained using the Wall Street Journal Penn Treebank tagged corpus [5].

In the Wall Street Journal Penn Treebank corpus, the words were drawn from a variety of sources and tagged using an automated process. The tags were then corrected by hand to achieve a "gold standard" document for POS tagging. The tagset from the Wall Street Journal Penn Treebank corpus [6] shows the granularity of our augmentation.

2 Term Document Matrix

A term/document matrix is a matrix composed of columns of documents and rows of the terms that occur in each of the documents. Figure 1 is the term/document matrix generated as a result of the following three-document set (These short examples are for illustrative purposes only).

Document 1: After the first day I felt a spring in my step.
Document 2: The first day of Spring was a beautiful day.
Document 3: When she first comes in, spring up and shout, "Surprise!"

Preprocessing of these three documents included removing the punctuation (we did not remove the "stop words" because of the length of the examples).

		Document 1	Document 2	Document 3
T1	after	0.09	0	0
T2	the	0.09	0.11	0
T3	first	0.09	0.11	0.1
T4	day	0.09	0.22	0
T5	I	0.09	0	0
T6	felt	0.09	0	0
T7	a	0.09	0.11	0
T8	spring	0.09	0.11	0.1
T9	in	0.09	0	0.1
T10	my	0.09	0	0
T11	step	0.09	0	0
T12	of	0	0.11	0
T13	was	0	0.11	0
T14	beautiful	0	0.11	0
T15	when	0	0	0.1
T16	she	0	0	0.1
T17	comes	0	0	0.1
T18	up	0	0	0.1
T19	and	0	0	0.1
T20	shout	0	0	0.1
T21	surprise	0	0	0.1
		0.99	0.99	1

Fig. 1 The term-document matrix generated from the three example sentences. The numerical cell values are scaled frequencies of words within documents. The last row is the sum of the scaled frequencies.

The tagset from the Wall Street Journal Penn Treebank corpus is given. Words that are very common, known as "stop words" are usually excluded from the matrix (e. g. see Table 1). The value in each cell of the matrix is the scaled frequency of the term in the document. To reduce the cost of query comparisons, the singular value decomposition is truncated after an *a priori* limited number of matrix entries (typically about 100). From this truncated matrix we extract the most significant 100 orthogonal factors from which the original matrix can be approximated by linear combination [4].

After this reduction, a vector of factor weights represents each document. The number of items in each vector equals the number of factors into which the original matrix was decomposed (100 in the above discussion). Queries are represented in a manner similar to documents. Query vectors are built from the scaled combination of the terms within the query.

Table 1. Some examples of common "stop words"

the	or	with	at	that
be	as	by	for	and
from	under	such	there	of
other	whether	also	than	which
now	where	these	when	we
an	to	but	upon	then
if	is	it	can	this

3 The Parts of Speech Tagger

POS taggers have been available for use since 1963 beginning with the seminal work of Klein and Simmons. We use Eric Brill's POS tagger [1, 2, 3]**.** A POS tagger defines syntactic categories such as noun, verb, etc. and then associates a category with each word in a document.

Before a document may be tagged with Brill's tagger, the tagger must be trained. In the training phase of Brill's tagger, the tagger fills in a set of rule templates that define how to apply POS tags. Hence, Brill's tagger is a rule-based artificially intelligence algorithm. These rule templates may be provided *a priori* [1, 2] (Table 2), or learned from an untagged corpus together with a dictionary [3].

Table 2. Rule templates from the Brill tagger

Change tag ***a*** to tag ***b*** when:
The preceding (following) word is tagged ***z***.
The word two before (after) is tagged ***z***.
One of the two preceding (following) words is tagged ***z***.
One of the three preceding (following) words is tagged ***z***.
The preceding word is tagged ***z*** and the following word is tagged ***w***.

The preceding (following) word is tagged ***z*** and the word two before (after) is tagged ***w***.
The preceding (following) word is ***q***.
The word two before (after) is ***q***.
One of the two preceding (following) words is ***q***.
The current word is ***q*** and the preceding (following) word is ***k***.
The current word is ***q*** and the preceding (following) word is tagged ***z***.
where ***a, b, z,*** and ***w*** are variables over the set of parts of speech and ***q*** and ***k*** are variables over all words in the training corpus.

The algorithms used in the Brill tagger with rule templates are based on comparison of the results of each pass of the tagger to the most common tag for a word in the training corpus. The tagger generates a list of tag errors by counting the number of times a word was tagged with tag A when it should have been tagged with tag B as indicated by the most common tag assigned to the word in the training corpus. The tagger calculates the net improvement of applying each rule template in the rule template set to each error in the list. The rule that results in the greatest net improvement is added to the rule set. This process continues until the net improvement from acquiring and applying new rules falls below a pre-set threshold [2]. To tag a sentence with the trained tagger, the sentence is first tagged with the start-state tagger and then each of the rules are applied in order. (We do not use the learning tagger.)

The Brill tagger begins by making an initial guess at the part of speech for each word as determined by examination of a large manually tagged corpus (Context is not considered in assigning this first tag). We use the Wall Street Journal Penn Treebank Corpus (When the tagged corpus does not include a word, the word is first tagged as a noun [2]. When a tagged corpus is available, the decision as to when a rule should be applied is based on decreasing the error-rate as compared to the tagged corpus (Brill, 1992). When no manually tagged standard is available, the procedure is:

1. Each word in the training set is tagged with all tags allowed for that word, as indicated in the dictionary.
2. Tags are altered by applying previously learned transformations.

Consider the transformation:
Change the tag of a word from X to Y in context C.

This transformation is scored as follows: Where $Y \in X$ for each tag $Z \in X$, $Z \neq Y$ compute *freq(Y)/freq(Z) * incontext(Z,C)* where *freq(Y)* is the number of occurrences of words unambiguously tagged with tag *Y* in the corpus, *freq(Z)* is the number or occurrences of words unambiguously tagged with tag *Z* in the corpus, and *incontext(Z,C)* is the number of times a word unambiguously tagged with tag *Z* occurs in context *C* in the training corpus.

Let $R = argmax_Z$ *freq(Y)/freq(Z) *incontext(Z,C).*

Then the score for the transformation *Change the tag of a word from X to Y in context C* is

*incontext(Y,C) – freq(Y)/freq(R) * incontext(R,C).*

We use a Windows port of Eric Brill's tagger (version 1.14). The port to Windows was performed by Sina Ghadirian and is available at http://www.readingenglish.net/software.

4 Formatting Tools

We developed a set of utilities using Python to assist in formatting and processing the text. The first adds whitespace around all symbols in a text. This formatting is required for the Brill tagger to properly process text. The second takes as input a corpus that includes multiple possible tags for each word and generates as output the corpus with each possible tag for a word appended to a separate copy of the word. The code for these utilities is available upon request.

5 Latent Semantic Analysis

The Infomap software (which was made available in 2004 by the Computational Semantics Laboratory at Stanford University) was used to perform the latent semantic analysis. Infomap is a freely available software program (http://infomap.stanford.edu). This software builds term-by-document matrices, performs single value decomposition (SVD) on those matrices, and runs queries against those matrices. Various other programs and more detailed information is available at the site noted above.

LSA begins with a large, sparse, term-by-document matrix containing scaled word frequencies computed from a document library (see Table 1 for a brief example). SVD is performed on this matrix to generate a set of orthogonal factors from which the original matrix can be approximated by linear combination.

Each of these orthogonal factors represents a dimension of similarity among the terms and documents of the original library. Thus it is possible to construct a vector of factor weights to approximately represent a document from the library. Furthermore, a vector may be constructed to

represent any subset of the terms that were included in original library. In constructing the vector the factors are scaled appropriately to accurately represent the terms of the document or query.

To find documents similar to a query, a vector is constructed from the terms of the query and compared to the vectors of all documents in the library. The documents whose vectors most closely match the query vector are returned as possible matches. Note that the match is not based on the query and documents containing the same terms, but rather on the query and documents containing similar weighting factors. We consider each principal component vector as an abstraction of a concept or idea that is represented in the original document library. Hence a document's vector represents a combination of one or more of these concepts (i.e. a semantic entity). LSA provides a handle to search a document library without direct text matching.

6 Procedure

On both the BrillWindows and Infomap software websites (noted above) documentation describing installation and use is included. For our textual library, we utilized news documents from www.cnn.com. Specifically, we ran two searches on www.cnn.com. First we searched for "china 2000". The results of this search were sorted by relevance. News documents were chosen from the year 2000 (No specific criteria were used in selecting documents other than that they listed the year 2000 as the publication date). We only included one document per newsworthy event. The text of the selected documents was copied and all formatting and graphics removed.

We then repeated the process using "russia 2000" as our search criteria. Again, results were sorted by relevance, documents were chosen from 2000, and we avoided selecting multiple documents dealing with the same topic.

Ten documents were selected from the china search and ten documents were selected from the russia search. Each set of documents was sorted into alphabetical order by the first word of their titles and they were numbered from one to ten. The result was two Sets of document files named china1, china2 . . .china10 and russia1, russia2 . . .russia10. Eighteen of the documents, nine from each set, were used to build the term/document matrix.

Next the Python utility described above was used to add white space around all symbols in the text files. So, for example, "(U.S. April 15, 2005)" is changed to " (U . S . April 15 , 2005) " . The files were saved in this format. They were then tagged using the BrillWindows port of the Brill tagger trained on the Wall Street Journal Penn Treebank corpus. (The Brill tagger uses backslash characters as a delimiter between words and tags. To improve readability, we replaced the backslash characters with underscores.)

The tagged and untagged files were then stored in separate directories. Each directory included an index file that listed the names of each of the member text files. The index file was used as the input to the Infomap software. The Infomap software was used in the default configuration. Two "models" were built. One was named "tagged_model" and the other was named "untagged_model".

7 Results

As configured, the Infomap software returns the twenty term or document vectors with the highest cosine similarity score to the query vector. Table 3 is the result of using the word "china" as a query against the untagged matrix and the word "china_nnp" as a query against the tagged matrix. NNP is the part-of-speech tag for proper nouns and is the tag most frequently associated with the word "China" in the document library (N = 220). Table 3 shows how adding POS tags changes the set of documents returned by the Infomap software. Table 3 shows how adding POS tags changes the set of words returned by the Infomap software.

Table 3. Results of queries "china" and "china_nnp" requesting document similarities

Untagged		Tagged	
Document Name	Cosine Similarity	Document Name	Cosine Similarity
china1_untagged.txt	0.463657	china3_tagged.txt	0.434622
china3_untagged.txt	0.457149	china2_tagged.txt	0.431527
china7_untagged.txt	0.435218	china7_tagged.txt	0.376409
china6_untagged.txt	0.405293	china6_tagged.txt	0.373664
china5_untagged.txt	0.400212	china5_tagged.txt	0.353992
china2_untagged.txt	0.386189	china1_tagged.txt	0.336986
china8_untagged.txt	0.376046	china8_tagged.txt	0.299148
china4_untagged.txt	0.329631	china4_tagged.txt	0.297588
china9_untagged.txt	0.210723	china9_tagged.txt	0.110808
russia7_untagged.txt	0.039227		
russia1_untagged.txt	0.002823		

Table 4. Results of queries "china" and "china_nnp" requesting word similarities

Tagged		Untagged	
Word	Cosine Similarity	Word	Cosine Similarity
china_nnp	1.0000	china	1.0000
dropping_vbg	0.5617	existence	0.5237
bans_nns	0.5610	republic	0.5154
flights_nns	0.5474	creation	0.5038
shipping_nn	0.5474	appease	0.4834
define_vb	0.5416	dropping	0.4784
economy_nn	0.5345	bans	0.4734
components_nns	0.5243	flights	0.4667
comprised_vbn	0.5243	shipping	0.4667
senate_nnp	0.5213	demonstrates	0.4620
demonstrates_vbz	0.5136	providing	0.4620
providing_vbg	0.5136	alternative	0.4525
passage_nn	0.5047	beijing	0.4448
market-driven_jj	0.4995	happen	0.4242
83-15_jj	0.4968	delivery	0.4079
voted_vbd	0.4968	taiwanese	0.4072
normal_jj	0.4903	fighters	0.3995
permanent_jj	0.4680	feet	0.3977
convince_vb	0.4642	indefinitely	0.3977
state-planned_jj	0.4587	lean	0.3970

8 Discussion

These results show that the addition of POS tags can change our results significantly. The addition of a POS tag is not sufficient to completely remove ambiguity from a given word. However, it did decrease the ambiguity of the word for our documents and therefore improves the accuracy of LSA.

The set of documents returned by the tagged search is smaller than the set of documents returned by the untagged search. False positives have been discarded. This suggests, even though the cosine similarities are smaller in some cases, that the accuracy of the matches is greater.

The range of cosine similarities for selected tagged documents is 0.323814. The range of cosine similarities for selected untagged documents is 0.460834. The smaller range of similarities of the selected documents demonstrates the increased accuracy of the tagged model. Apart from accuracy, a query algorithm that reduces the number of matches returned is of importance. Trimming the users search is a desirable goal.

When we change the query to return words instead of documents, we see a similar pattern in the response sets. The range of the cosine similarities of returned words is smaller in the tagged set than in the untagged set. The range of cosine similarities in the untagged set is 0.5413 while the range of cosine similarities in the untagged set is 0.6030. This smaller range of cosine similarities again demonstrates the increased accuracy of the tagged model.

For word results, the cosine similarities are generally higher in the tagged response set. In the tagged set the maximum cosine similarity, excluding the query word itself, is 0.5617. In the untagged set the maximum cosine similarity is 0.5237. The average cosine similarity of the tagged set is 0.5385. The average cosine similarity of the untagged set is 0.4782. The higher average and smaller range of cosine similarities of the tagged set show the increased accuracy of the LSA when selecting words that are semantically similar to the query.

9 Conclusions

The possibilities of LSA are only beginning to be explored. There remains much work to be done in finding possible improvements to and applications of LSA. One change that may result in further improvements in accuracy is decreasing the granularity of the tags used in POS tagging. Because some words may be tagged slightly differently and yet be semantically similar, a less fine-grained tag set may result in more accurate semantic groupings. For example, in the Wall Street Journal Penn Treebank corpus, the word "put" may be tagged VB – base form verb, VBN – past participle verb, JJ - adjective, NN – common noun, or VBD – past tense verb. In the current implementation of the POS-tagged LSA, each of these would be considered a distinct word. While grammatically the base form verb is different from the past participle verb and the past tense verb, semantically they are all similar. Dropping the last character from the three-character tags would allow the LSA software to treat each of these separate verb uses as a single idea, thus potentially improving the accuracy of the results. Note that this change maintains the accuracy gained by adding POS tags. The granularity of the parts-of-speech is decreased to noun, verb, determiner, adjective, adverb, etc.

Another area for possible improvement is adding pronoun disambiguation software to LSA. In standard LSA applications, pronouns are included on "stop lists" of words that are removed from consideration when constructing the original term-document matrix. Pronoun disambiguation software attempts to replace pronouns with the nouns to

which they refer. So the sentence, "Mary said she was going to the store." becomes, "Mary said Mary was going to the store." Most nouns are not included on the "stop lists". If pronoun disambiguation software were used in a pre-processing stage the scaled frequencies of the nouns in the term-document matrix would more accurately reflect the semantic content of the document and the LSA would also be more accurate.

We are currently combining this work with natural language processing. Specifically, we utilize the LSA analysis to select the most probable context for a text. Based on the context we then disambiguate English descriptions, re-write them in unambiguous English, and present them to the user for verification of our understanding. This can iterate until the English is unambiguous. We then use context dependent Context Free Grammar's to compile instructions given within the English text into a program. In this way we utilize English as a programming language, assisted by our context identifier.

References:

[1] Brill, E., "A simple rule-based part of speech tagger", Proceedings of the Third Annual Conference on Applied Natural Language Processing, Trento, Italy, 1992.

[2] Brill, E., "Some advances in rule-based part of speech tagging", Proceedings of the Twelfth National Conference on Artificial Intelligence, Seattle, Wa, 1994.

[3] Brill, E. & Pop, M., "Unsupervised learning of disambiguation rules for part of speech tagging", In Armstrong, S., Church, K. W., Isabelle, P., Manzi, S., Tzoukermann, E., & Yarowsky, D. (Eds.), Natural Language Processing Using Very Large Corpora. Kluwer Academic Press, 1999.

[4] Deerwester, S., Dumais, S. T., Furrnas, G. W., Landauer, T. K., & Harshman, R., "Indexing by latent semantic analysis", Journal of the American Society For Information Science, 41, 391-407, 1990.

[5] Marcus, M., Santorini, B., & Marcinkiewicz, M., "Building a large annotated corpus of English: the Penn Treebank", Computational Linguistics, Volume 19, 1993.

[6] Santorini, B., "Part-of-speech tagging guidelines for the Penn Treebank Project", Technical report MS-CIS-90-47, Dept., of Computer & Info. Science, Univ of Pennsylvania, 1990.

[7] Stanford University, "Infomap NLP software: An open source package for natural language processing", Computational Semantics Lab: Center for the Study of Language and Information, Stanford University, 2004.

[8] Wiemer-Hastings and Zipitria, "Adding syntactic information to LSA", Proceedings of the 22nd Annual Conference of the Cognitive Science Society. Mahwah, NJ. Erlbaum, 2000.