Opinion Analysis in Document Databases

C.Cesarano, A.Picariello
University of Naples Federico II
Dipartimento di Informatica e Sistemistica
{cacesara,picus}@unina.it

D.Reforgiato, A.Sagoff, V.S.Subrahmanian, B.Dorr
Univerity of Maryland - College Park
Department of Computer Science
{diegoref,amelia,vs,bonnie}@cs.umd.edu

Abstract

There are numerous applications in which we would like to assess what opinions are being expressed in text documents. Forr example, Martha Stewart's company may have wished to assess the degree of harshness of news articles about her in the recent past. Likewise, a World Bank official may wish to assess the degree of criticism of a proposed dam in Bangladesh. The ability to gauge opinion on a given topic is therefore of critical interest. In this paper, we develop a suite of algorithms which take as input, a set D of documents as well as a topic t, and gauge the degree of opinion expressed about topic t in the set D of documents. Our algorithms can return both a number (larger the number, more positive the opinion) as well as a qualitative opinion (e.g. harsh, complimentary). We assess the accuracy of these algorithms via human experiments and show that the best of these algorithms can accurately reflect human opinions. We have also conducted performance experiments showing that our algorithms are computationally fast.

Introduction

There are numerous applications where the ability to understand opinions expressed in documents is critical. Political campaigns may wish to understand public sentiment about a romantic affair by a candidate running for office. Likewise, the US government may wish to gauge the strength of public sentiment about the Abu Ghraib prison abuse scandal — this will serve as our running example in this paper. There are numerous techniques in the literature to analyze opinions - clearly the best known techniques are those followed by polling organizations that directly canvass people for opinions. Unfortunately, this is a very expensive proposition. An alternative has been to study opinions expressed in various kinds of document collections such as movie reviews(F. Salvetti 2004) and news reports, and so on. In this paper, we focus on the problem of analyzing opinions reported in news articles. We provide a general opinion analysis architecture in which many different algorithms to score opinions can be "plugged in."

Several papers(B. Pang 2002; F. Salvetti 2004; Turney 2002) in the "opinion analysis" genre come up with binary scores. In the case of movies, a binary score is a "recommend" "don't recommend" score. On the other hand, if we are interested in knowing the strength of opinion in Saudi Arabia about the Abu Ghraib scandal, just a "yes/no" binary score seems insufficient. It would be vastly preferable if we could give it a numeric score (e.g. 0 means very positive, 1 means very harsh). Alternatively, we could at least grade it from a list of qualitative ratings (e.g. very positive, positive, neutral, negative, very negative). In this paper, we present general methods for both *quantitative and qualitative scoring* of the opinions expressed in a document about a particular topic. Note that documents do not get a score all by themselves — it is documents *in conjunction with a topic of interest that get scores.*

We first lay out a general architecture for analyzing opinions. This architecture has the advantage that almost anyone's algorithm for opinion analysis can be "plugged in." The architecture has several parts, only one of which we focus on in this paper (due to space constraints) – namely the scoring methods used. We develop multiple quantitative scoring functions as well as a *hybrid scoring method* that can be used to integrate together, the results of multiple scoring methods (not just ours). We provide an experimental analysis of our method using an archive of over 350 news articles about the Abu Ghraib scandal.

(T. Wilson 2004) present a learning based method to classify opinion based on identifying and leveraging a rich set of clues and some novel syntactic features specifically developed for this purpose. (V. Hatzivassiloglou 1997) worked extensively on the semantic orientation (positive or negative) of adjectives. Like the binary description of (F. Salvetti 2004) words are defined only as positive or negative, while we additionally provide rankings based on intensity. (Turney 2002) developed an algorithm for classifying a document as positive or negative. His algorithm also only provides the rankings recommended or "not recommended" instead of the continuum of rankings we have. Turney's algorithm is built almost exclusively for reviews (such as movie reviews and car reviews). For instance "bad" and "god-awful" denote different negative intensities that our model would capture, while the above models would not. Furthermore, their paper defines adjectives in terms each other, finding opposites and synonyms depending on the conjunctions used when the adjectives apply to the same noun, and uses sub-

jective human ratings to test the accuracy of their method. (B. Pang 2002) also worked on classifying reviews as positive or negative, experimenting with Naive Bayes, maximum entropy, and support vector machine based algorithms. Again, these algorithms are more appropriate for finding a two-category polar relationship rather than ranking an adjective (or document's) intensity across a continuum of values as we do

The main differences between our work and these efforts are that:

(i) we provide a continuum of ratings for words (adjectives as well as non-adjectives),

(ii) our scores for the opinions expressed in a document are likewise continuous, not binary,

(iii) we develop multiple scoring methods including qualitative scoring methods and

(iv) we develop a model to combine multiple scoring methods together (including many developed by others) whereas none of the earlier efforts seems to do so.

Opinion analysis architecture

Suppose a user wants to assess the opinions expressed about topic t in a set $\mathcal{D}$ of documents. Our architecture consists of the following components:

- **User specification:** The user specifies a set of sources (e.g. directories,domain names,URLs), a topic t of interest, and a time interval of interest.
- **Web spider:** A web spider that we have built will retrieve all documents in any of the descendant directories of the specified locations that are relevant to topic t and that were authored/updated during the time frame of interest. This is the set $\mathcal{D}$ of documents of interest to the user. [1] There are numerous algorithms to find documents about a given topic t (S. Deerwester 1990) - hence, we do not address this problem in our paper.
- **Scored opinion expressing word bank:** We have created a *scored opinion expressing word bank* in which words (e.g. appalling, desirable, mistreatment etc.) that directly or indirectly express an opinion are assigned a *score*. The lower the score, the more positive the word is.

  Two methods to score words are described in this paper - our system uses an extensible library of opinion expressing word scoring methods that can easily be expanded. Note that it is possible to restrict the scored opinion expressing word bank to specific types of words (e.g. adjectives as done by Salvetti et. al. (F. Salvetti 2004) and Hatzivassiloglou et. al.(V. Hatzivassiloglou 1997)), rather than to use all the words in the scored opinion expressing word bank.
- **Quantitative opinion analysis algorithms:** We have developed several algorithms that take each document $d \in \mathcal{D}$ and assess the harshness of d w.r.t. topic t. The scored opinion expressing word bank is used in deriving a quantitative score for the document. We provide three *families* of algorithms to assign a score to the opinions expressed about a given topic t in a given document d. In addition, we provide a *hybrid algorithm* that can take the scores returned by *any* algorithms, not just ours, and merge them.[2] The time taken by these algorithms, and the accuracy of these algorithms in gauging opinions held by humans depends not only on the algorithms themselves, but also the methods used to score words.
- **Qualitative scoring module:** The system can either return the *raw quantitative score* to the user, or can return a *qualitative score*. A qualitative score is derived from the quantitative score by assigning an adjective (e.g. positive, harsh, very harsh, and so on) to various ranges of qualitative scores. We show that we can automatically learn such ranges. Our system can also lay out the scores for documents on a spatio-temporal basis (e.g. show how opinions about the Abu Ghraib scandal changed with time in Saudi Arabia vs. Belgium) — but for space reasons, we do not go into details of this here. We describe methods to assess the qualitative score of document.

Due to space constraints, this paper will primarily focus on the last three modules rather than on the user interface and the web spider.

The Scored Opinion-Expressing Word Bank

We created a scored opinion expressing word bank by selecting a collection $\mathcal{D}_{\text{test}}$ of 100 randomly selected "training" documents. Each document was read by 16 subjects, each of whom gave the document a harshness score from 0 to 1 - a high score is a very harsh document, while a low score is a very positive document.

Here is a paragraph from one example document[3] document. We will use this example to illustrate some of the concepts introduced in this paper.

Example 1

"As news of the disgraceful mistreatment of prisoners by American soldiers sweeps the world, our enemies celebrate a major propaganda gift," writes Ralph Peters in the New York Post. "Even our friends cannot defend the indefensible."

Suppose now that w is a word and d is a document. A *word scoring function* wsf is any mapping from the set of all opinion expressing words to the unit interval $[0, 1]$. Of course, there are infinitely many wsf's — our task is to find a few good ones. In addition, note that we may want to restrict the set of words to which a score is assigned (e.g. just adjectives as done by (F. Salvetti 2004;

[1] Similar principles can be applied to find all postings to newsgroups, email archives, etc.

[2] We assume all scoring methods are normalized to a single scoring scale and that they all assume higher scores mean more harsh documents. Note that if a scoring method s makes the opposite assumption, i.e. that a low score reflects harshness while a harsh score means the document is positive, then we can merely use $\frac{1}{s(d)}$ or $1 - s(d)$ as our harshness metric (in the latter case we assume scoring is on a 0 to 1 scale).

[3] http://www.opinionjournal.com/best/?id=110005034

V. Hatzivassiloglou 1997)). Of course, in the above quote, words like "mistreatment" and "propaganda" that have a negative connotation would be missed out if we restrict ourselves to adjectives alone.

Definition 1 ($\mathsf{numb}(a, d)$**)** *We use the notation* $\mathsf{numb}(w, d)$ *to denote the number of occurrences of either* w *or a synonym of* w *in document* d*.*

For instance, if we consider the paragraph in Example 1 as a document d_0, the value $\mathsf{numb}(\text{"disgraceful"}, d_0)) = 1$.

Definition 2 ($\mathsf{avsc}^k(d)$**)** *Suppose* $\mathcal{D}_{\text{test}}$ *is a set of test documents, and* $H = \{h_1, \ldots, h_m\}$ *is a set of human users, each of whom renders a non-negative score* $h_i(d)$ *about the document* d*. Suppose we order the scores in the multiset* $\{h_1(d), \ldots, h_m(d)\}$ *in ascending order and delete the top* k *scores and the bottom* k *scores. We use the notation* $\mathsf{avsc}^k(d)$ *to denote the average of the remaining scores.*

For instance, suppose 10 human subjects read a given document and assigned scores $0.8, 0.7, 0.2, 1, 0.75, 0.6, 0.9, 0.8, 0.6, 0.7$ to the document. When we order this set of scores in ascending order, we get $0.2, 0.6, 0.6, 0.7, 0.7, 0.75, 0.8, 0.8, 0.9, 1$. If $k = 2$, we eliminate the two lowest and the two highest numbers in this sorted list to get $0.6, 0.7, 0.7, 0.75, 0.8, 0.8$. The average of these numbers is 0.725 which is the value returned by $\mathsf{avsc}^2(d)$. By setting $k > 0$, we get rid of outliers. In the above example, one person assigned a very low score (0.2) which seems to be an outlier compared to all other scores. One must be careful in the selection of k as we clearly want to keep a reasonable selection of scores to average over.

Definition 3 *Given any document* d *and any collection* $\mathcal{D}$ *of documents, we use the notation* $\mathsf{oew}(d)$ *and* $\mathsf{oew}(\mathcal{D})$ *to respectively denote the set of all opinion-expressing words (and their synonyms) occurring in document* d *and* $\mathcal{D}_{\text{test}}$*.*

If we wish to restrict interest to adjectives (e.g. (F. Salvetti 2004)), we may use a function $\mathsf{oew}_{adj}(d_0)$ which returns all adjectives occurring in the document of Example 1 — in this case, the set returned is: { disgraceful,indefensible }.

Pseudo-expected value word scoring

We now introduce our first method to score opinion expressing words. This method draws its inspiration from the concept of expected values in statistics(Ross 2001).

Using the notation described above, we observe that the expression

$$\frac{\mathsf{numb}(w, d)}{\Sigma_{w' \in \mathsf{oew}(\mathcal{D}_{\text{test}})} \mathsf{numb}(w', d)}$$

denotes the proportion of occurrences of an opinion expressing word w and its synonyms compared to the total number occurrences of adjectives in $\mathcal{D}_{\text{test}}$. This expression therefore expresses the relative proportion of w and its synonyms in the document.

The expression

$$\frac{\mathsf{numb}(w, d)}{\Sigma_{w' \in \mathsf{oew}(\mathcal{D}_{\text{test}})} \mathsf{numb}(w', d)}$$

is like an expected value computation in statistics - if we multiply it by $\mathsf{avsc}^k(d)$, we would have a measure of the contribution of the score $\mathsf{avsc}(d)$ of d contributed by opinion expressing word a and its synonyms. We can set the score, $\mathsf{pevs}^k(w)$ of word w by averaging the contribution of the score of w across all the documents in the test set. That is:

$$\mathsf{pevs}^k(w) \quad = \quad \frac{\Sigma_{d \in \mathcal{D}_{\text{test}}} \left(\mathsf{avsc}^k(d) \times \frac{n(w,d)}{\Sigma_{w' \in \mathsf{oew}(\mathcal{D}_{\text{test}})} n(w',d)} \right)}{\Sigma_{d \in \mathcal{D}_{\text{test}}} \mathsf{avsc}^k(d)}.$$

It is important to note that the above definition gives us a whole family of methods to score opinion expressing words based on the selection of k*.* We will experiment with different versions of k throughout this paper.

It is important to note that many variants of this strategy can also be considered. We have, for example, considered the case where instead of counting the total number of occurrences of an opinion expressing word (or its synonyms), we only count the number of occurrences that occur in either a direct expression of opinion (e.g. "Amnesty International stated that conditions in Abu Ghraib were appalling") or an indirect one (e.g. "CNN reported that they were alarmed by reports of abuse...."). Many other variations about which opinion expressing words are counted are also possible but cannot be explored in detail here due to space constraints.

Pseudo Standard-Deviation Adjective Scoring

An alternative strategy is to use a standard deviation based strategy. Here, we start by considering the scores assigned to each test document (on some fixed scale, e.g. 0 to 1 or 1 to 10) by the human users.

Definition 4 ($\mathsf{sdsc}^k(d)$**)** *Suppose* $\mathcal{D}_{\text{test}}$ *is a set of test documents, and* $H = \{h_1, \ldots, h_m\}$ *is a set of human users, each of whom renders a quantitative score* $h_i(d)$ *about document* d*. Let* μ *be the mean of all these scores and let* σ *be the standard deviation. Let* $k \geq 1$ *be any integer. We set* $\mathsf{sdsc}^k(d)$ *to be the mean of the multiset* $\{h_i(d) \,|\, abs(h_i(d) - \mu) \leq k \cdot \sigma\}$*.*

In other words, when assigning a score to an opinion expressing word, we start by evaluating the scores assigned to test documents by human subjects. We compute the mean and standard deviation of these scores. We then throw away all scores that are more than k standard deviations away from the mean, and take the average of the remaining scores. This strategy has the advantage of eliminating outliers on a sound statistical basis(Ross 2001). For example, it is statistically known that for normal distributions, about 97% of all values in a set lie within three standard deviations of the mean. So $k = 3$ above would be a good choice.

We then assign a score to each opinion expressing word w in exactly the same way as we did with pseudo-expected value scoring - the only difference is that sdsc^k is now used in the formula instead of avsc^k, i.e. the score assigned is given by the formula below.

$$\mathsf{psds}^k(w) \quad = \quad \frac{\Sigma_{d \in \mathcal{D}_{\text{test}}} \left(\mathsf{sdsc}^k(d) \times \frac{n(a,d)}{\Sigma_{w' \in \mathsf{oew}(\mathcal{D}_{\text{test}})} n(w',d)} \right)}{\Sigma_{d \in \mathcal{D}_{\text{test}}} \mathsf{sdsc}^k(d)}.$$

Scoring documents

Suppose now that we have a scored opiniion expressing word bank using any arbitrary word scoring function wsf (such as the pseudo expected value scoring method or the pseudo standard deviation scoring method — of course, any other method can be used as well) and we wish to score documents d in some collection $\mathcal{D}$ of documents. We now present a suite of algorithms to score opinions expressed in a given document d.

Topic-Focused (TF$^{\text{wsf}}$) Algorithm

The Topic-Focused (TF$^{\text{wsf}}$) algorithm finds all sentences involving either a direct or indirect expression of opinion about the topic t of interest. It then assigns a score $\text{wsf}(s)$ to each sentence s by summing up the scores (using $\text{wsf}(a)$) of all opinion-expressing words a occurring in s. It returns the average sentence score of all such sentences. Notice that TF^{wsf} returns different answers based on the selected word scoring method. We can many, many variants of TF$^{\text{wsf}}$ based on precisely which word scoring function wsf is used.

function TF$^{\text{wsf}}$(d,t,extractRelSent)
  d is a document
  t is the topic of interest
begin
  Result←0 //no result so far
  NSentences←0// no sentences processed so far
  NWDS←0// no words processed so far
  Sentences ← extractRelSent(d,t)// find relevant set of sentences in d
  foreach $s \in Sentences$ **do**
    NSentences ← NSentences+1
    $OEW \leftarrow findOEW(s)$// find multiset of OEWs in s
      foreach $w \in OEW$ **do**
        $Syn \leftarrow findsyn(w)$// $Synonyms$ is the set of synonyms of w
        foreach $w' \in Syn$ **do**
          if $w' \in SWB$**then**
            $NOEW \leftarrow NOEW + 1$
            $Result \leftarrow \text{wsf}(w') + Result$
          end if
        end foreach
      end foreach
  end foreach
  if ($NSentences > 0$**) then**
    $Result \leftarrow Result/NSentences$
  else
    $Result \leftarrow 0$
  end if
  return $Result$
end

Distance-weighted topic focused (DWTF$^{\text{wsf}}$) Algorithm

The Distance-weighted topic focused (DWTF$^{\text{wsf}}$) algorithm examines all sentences in the document and assigns an "initial" score to them (e.g. by finding the average scores of the adjectives in the sentence or by taking the sum of the scores of adjectives in the sentence, and so on). In fact, *any* method, sc, to score a sentence can be plugged in and this could include TF$^{\text{wsf}}$ applied to a document containing just one sentence). It then splits the document into those that express either a direct or indirect expression of opinion about the topic t or interest (this set is denoted by $OpinionS$, and those sentences in the document that do not express an opinion about t (denoted by $NotOpinionS$). For each sentence s that expresses an opinion about t and each sentence s_n that does not, it finds the distance between the sentences and then *multiplies* this distance by a constant $\beta \geq 1$ that can be selected in any way desired. We then multiply the score of sentence s_n by $e^{-\beta Distance(s,s\_n)}$ — this modulates the impact of s_n's score on s. Note that instead of using $e^{-\beta Distance(s,s\_n)}$ we could have used any similar function — e.g., $2^{-\beta Distance(s,s\_n)}$. In other words, if harsh adjectives are used in a sentence s_n that does not express an opinion about t and s_n is very near s, then the impact is large — otherwise it is small. [4]

function DWTF(d,t,β,sc)
  d is a document
  t,topic of interest
  Result←0
  $OpinionS \leftarrow GETOpinionSentences(d,t)$
    //array containing sentences about t
  $NOTOpinionS \leftarrow GETNOTOpinionSentences(d,t)$
    //list of sentences that do not express an opinion about t
  foreach $s \in OpinionS$ **do**
  val←0
    foreach $s\_n \in NOTOpinionS$ **do**
      val ← $e^{-\beta Distance(s,s\_n)} * sc(s\_n) + val$
      weight ← $e^{-\beta Distance(s,s\_n)}$
    end for
  Result← $Result + sc(s) + \frac{val}{weight}$
  end for
  Result ← $\frac{Result}{m}$
return Result
end

Template-based (TB$^{\text{wsf}}$) Algorithm

This algorithm uses a set of templates and only examines sentences that "match" a template. It then uses the same approach as the TF$^{\text{wsf}}$ algorithm to assign a score. As in the case of the DWTF algorithm, *any* scoring function for sentences can be used.

function TB(d,$Templates$,t,sc)
  d is a document
  $Templates$ is a list of templates
begin
  Result←0
  $value \leftarrow 0$
  $Relevant$ = set of sentences i d about topic t
  foreach $s \in d$ **do**
    foreach $temp \in Templates$ **do**
      if (s $match$ $temp$)**then**
        $PartofSentence \leftarrow GETPartofSentence(s,t)$
          //string containing part of sentence that matches a template
        $value \leftarrow value + sc(PartofSentence)$
        $n \leftarrow n + 1$
      end if
    end foreach
  end foreach
  Result← $\frac{value}{n}$
return $Result$
end

Hybrid Evaluation Method (HEM)

The HEM$^{\vec{ds},r,m}$ algorithm is far more general. It associates with each document d, a vector of length m for some integer m. The vector consists of functions $\vec{ds} = \langle ds_1, \ldots, ds_m \rangle$ to assign scores to the document. For example, suppose we could use the three methods listed above (i.e. TF$^{\text{wsf}}$,DWTF$^{\text{wsf}}$,TB$^{\text{wsf}}$) (with any choices for wsf that we like) to assign scores s_1, s_2, s_3 to some document d. In this case, we associate the vector (s_1, s_2, s_3) with d. The same procedure is also applied to all documents in $\mathcal{D}_{\text{test}}$.

The HEM$^{\vec{ds},r,m}$ algorithm looks at the vectors associated with documents in $\mathcal{D}_{\text{test}}$ and finds the r-nearest neighbors of d's associated vector for some number $r > 0$. Suppose these documents in $\mathcal{D}_{\text{test}}$ are $d_1, \ldots, d_r$. The score returned for document d is the average of the scores assigned to documents $d_1, \ldots, d_k$ by the human subjects who evaluated these documents. *Note that* HEM^{r,m} *is shorthand for a slew of algorithms based on using m different scoring functions and different values of r.*

[4] We do not need to compare s with all sentences just those that do not express an opinion about t.

```
function HEM(d,r,ds⃗,sc⃗_{D_test})
    d is a document
    r is the number of nearest neighbors we want to find
    ds⃗ is a vector containing a set of scoring algorithms
    sc_{D_test} is a matrix containing the vectors of the scores of D_test using the
                any algorithms and the score assigned by human subjects
begin
    Result←0
    foreach ds ∈ ds⃗ do
        value ← ds(d)//array of scores of docs using algorithsm in ds⃗
    end foreach
    for i = 1 to r do
        ResultDoc[i][1] ← score[i]
            //matrix containing the score vector and the index of d ∈ D_test
        ResultDoc[i][2] ← number_of_the_Document_in_D_test
    end for
    foreach score ∈ score⃗ do
        if (Distance(value, score) < each v ∈ ResultDoc[][1])th en
            ResultDoc[i][1] ← score[i]
            ResultDoc[i][2] ← index_of_the_Document_in_D_test
        end if
    end foreach
    foreach element ∈ ResultDoc do
        Score_given_by_human_subjects ← sc_{D_test}(element[][2], 2)
        Result← Score_given_by_human_subjects + Result
    end foreach
    Result ← Result/r
return Result
end
```

Qualitative Scoring: The QualScore Algorithm

All the preceding algorithms provide a quantitative score reflecting the opinions expressed about topic t by a given document. The The *qualitative adjective scoring* algorithm Qualscore assumes that the rating scale is a list of adjectives $a_1, \ldots, a_n$ in increasing order of "positiveness" of the adjective (e.g. POSITIVE, HARSH, VERY HARSH, together with a sequence of $n-1$ threshold values $0 \leq t_1 \leq \cdots \leq t_{n-1} \leq 1$. Note that these adjectives are just used to rate a document — the actual words used in deciding what the rating is come from the list of score opinion expressing words described earlier.

For instance, if our rating scale is POSITIVE, HARSH, VERY HARSH, we may use the threshold values $0.4, 0.7$ to indicate that any document with a score between 0 and 0.4 is considered POSITIVE, anything with a score between 0.4 and 0.7 is considered HARSH and anything between 0.7 and 1 is considered VERY HARSH.

The obvious question the reader will have here is: "where do these ranges come from?" Ranges may be automatically discovered as follows: we use any of the document scoring methods described in the preceding sections to score the document d. We then find the nearest neighbor d_{nn} of the document d in the set $\mathcal{D}_{\text{test}}$ of test documents and use the qualitative score assigned by the user to that document.

```
function Qualscore(d,r,ds⃗,sc⃗_{D_test},Templates ,t)
begin
    Result←0
    value←0
    foreach ds ∈ ds⃗ do
        value ← ds(parametes)//array containing the score of a
                                //document according with the algorithm in ds
    end foreach
    IDinD_test ← findNearest(value)
    Result ← getQualScore(IDinD_test)
return Result
end
```

Implementation and Experiments

We have implemented a prototype opinion scoring system - Java was used to implement the user specification module, web spider, and all the scoring algorithms. Oracle was used to store and index news articles. The system runs on a 2GHz Linux box with 1 GB of RAM.

Building the Scored Opinion Expressing Word Bank. We trained our algorithms using a corpus of 352 news articles on 12 topics drawn from various US news sources such as New York Times, the Denver Post, ABC News and the Houston Chronicle. 100 of these news articles were evaluated by 16 students to build up the Scored Opinion Expressing Word Bank. We used the adjective scoring methods described earlier. The following table shows the scores assigned to certain adjectives (with $k = 2$ for the two scoring methods described in this paper.

Adjective	pevs$^2(word)$	psds$^2(word)$
ludicrous	0.45	0.51
misleading	0.38	0.48
rabid	0.75	0.71
gruesome	0.78	0.81
hideous	0.08	0.13
...	...	...

We see that in general, words like "rabid" and "gruesome" are considered harsher than "misleading." Not surprisingly, there can be some odd ratings (e.g. should hideous really be rated 0.8 which sounds fairly positive?).

Measuring Accuracy. We used a set of 28 students — these students did not overlap with the 16 who created the scored adjective bank — to each evaluate a set of 34 news articles on "Alberto Gonzalez" who at the time was undergoing hearings in the US Senate for appointment as the US Attorney General. The students rated each article from 0 to 1.

We ran our algorithms and evaluated the precision of each of our algorithms as follows. Let $Ret(thr, alg)$ be the set of all news articles retrieved by the algorithm alg which has a score that exceeds a given threshold score. Let $User(thr)$ be the set of all documents that the users say has a score over thr. We then set the precision of an algorithm for a given threshold to be:

$$precision^{thr}(alg) \quad = \quad \frac{|Ret(thr, alg) \cap User(thr)| \times 100}{|Ret(thr, alg)|}.$$

Figure 1 shows how the precision of our algorithms changes as the threshold is increased. In our experiments, HEM used DWTF, TB and TF all with the opinion expressing word scoring function psds^2. The figures for different values of k are not very different.

Results. When the threshold is between 0.5 and 0.8 approximately, the best algorithm in terms of precision is $DWTF$ with $k = 2$. However, for any other threshold, HEM yields the best performance. Nonetheless, both these algorithms approximate each other closely and are never more than 10% away from each other in terms of precision.

Measuring Performance. We measured the performance of our algorithms in terms of computation time. The same conditions used in the preceding experiment apply. We varied the number of news articles relevant to t from 0 to 100. Figu 2 shows the results. The reader can easily see that TB takes the least time, while DWTF takes the most time. Note that HEM of courses takes even more time as it must compute all of the three functions (DWTF, TB and TF). In

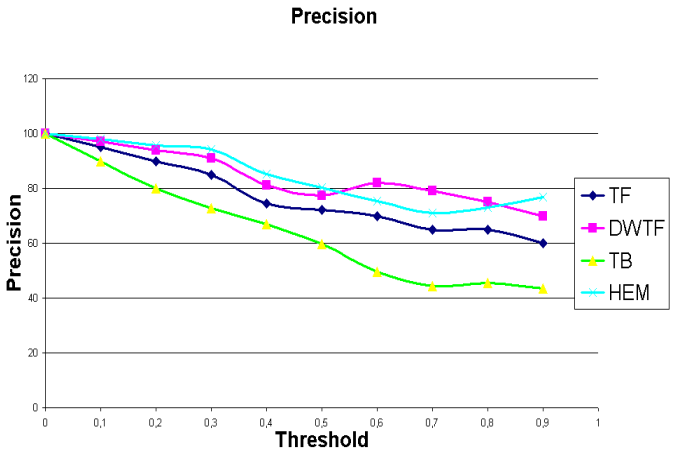

Figure 1: Precision trend

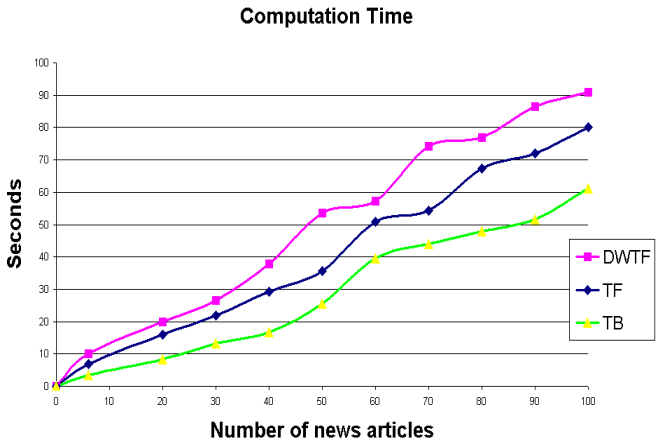

Figure 2: Computation Time

general, assessing the harshness of a set of news documents seems to increase linearly with the number of documents (not surprising) and seems to take approximately 0.9 seconds per document considered.

Related Work

There has been a significant amount of work on assessing opinion or sentiment in documents. Salvetti et. al. (F. Salvetti 2004) determine "opinion" polarity" (e.g., for classifying movie reviews). The key differences between our work and theirs is that: (i) they have no analog of our scored Word Banks, (ii) the return 0 or 1 rather than a continuum of values as we do and hence they cannot capture varying levels of opinion, just good or bad, (iii) their work scoring models are based on Naive Bayes and Markov Models rather than expected values and standard deviation – this falls out of the fact that they are trying to get just good/bad vs. a continuum of values as we do.

Weibe and her colleagues(T. Wilson 2004) have been doing a tremendous amount of linguistic subjectivity analyses for many years.

Conclusions

There is growing interest in the ability to extract opinions from documents. A tremendous amount of work has been done on extracting binary opinions (yes/no, recommend/don't recommend) (B. Pang 2002; F. Salvetti 2004; Turney 2002; V. Hatzivassiloglou 1997).

In this paper, our focus is on delivering a measure of the intensity of opinion that a document expresses about a given topic. To do this, we provide a general purpose architecture that can neatly embed many scoring functions other than ours. We show how to use human assessments of test data to learn the "intensities" of words in an opinion expressing word bank. We then provide a set of quantitative models to compute the intensity of opinion expressed by a document on a given topic. We also develop a hybrid quantitative scoring model that can be used to score the harshness of a document w.r.t. a specific topic. Finally, we develop a qualitative scoring model that classifies documents according to qualitative ratings. Our experiments indicate that the algorithms work efficiently and that the ratings they produce match human ratings closely.

Acknowledgements. Work funded by AFOSR contract FA95500510298, ARO contract DAAD190310202, and by DARPA/UC Berkeley contract number 03-000219. (V. Hatzivassiloglou 1997)

References

B. Pang, L. Lee, S. V. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 79–86.

F. Salvetti, S. Lewis, C. R. 2004. Automatic opinion polarity classification of movie reviews. *Colorado Research in Linguistics* 17(1).

Ross, S. 2001. *A first course in probability*. Prentice Hall.

S. Deerwester, S. Dumais, T. L. G. F. R. H. 1990. Indexing by latent semantic analysis. *Journal of the american society of information science* 41(6):391–407.

T. Wilson, J. Wiebe, R. H. 2004. Just how mad are you? finding strong and weak opinion clauses. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI)*, 761–769.

Turney, P. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, 417–424.

V. Hatzivassiloglou, K. M. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of ACL*, 174–181.