Framework for Sentiment Aware Queries and Results in Search Using Oracle Text

Shubhro Jyoti Roy, Alexandra Czarlinska, and Asha Tarachandani

***Abstract*—Users today need to express their informational need in a way such that the search results can be further analyzed to directly address the need, instead of merely returning a list of lexical hits. In this paper we address the problem of extracting sentiment metadata related to the user's topic or entity of interest along with the search results. We propose a framework that provides the flexibility of specifying a sentiment related informational need associated with a particular topic of interest, along with the standard keyword query. This information is then used to extract sentiment scores for the expressed entity of interest based on the "hitlist" of most relevant documents returned by the main search query. Our experimental results based on product and movie review datasets, demonstrate the advantages of embedding the sentiment processing within the search engine framework.**

***Index Terms*—Sentiment analysis, information retrieval, query framework, text analytics, sentiment aware search.**

I. Introduction

In the age of exabytes of instantly accessible online data and given the pervasiveness of social media, users and companies alike need more insight rather than mere "hits" to be returned from search operations. Users need a way to form queries that not only return a list of lexical hits, but that also return some additional metadata that directly addresses the user's "informational need". An example of this trend can be seen in simple weather queries. A search query like "Weather San Francisco" today returns a widget showing the current temperature and weather predictions for the day followed by the top-n weather related websites (from which the widget data may have been populated). This is in contrast with prior search engine designs where only the top-n weather related websites were provided without any direct answer addressing the informational need of the user. In general, this type of functionality is provided for queries on weather, recent events (political, financial and sports-related), and famous people. Indeed, advances have been made towards a richer user interface for search results by exploiting metadata extracted from result-set documents [1].

Yet, one particular kind of informational need that has been overlooked by modern search engines is "sentiment". For example, let us consider that we want to understand "What is likely to happen to US stock prices in the next few months given subsequent applications of Quantitative Easing by the US Federal Reserve?" Note that the question is a real-world question and is factual in nature, yet it entails an element of opinion since no one can fully predict the future. How would we address this informational need today using web search engines?

In one approach, we could issue the query "predictions for stock prices". This will return a list of related web sites or articles gathered from a variety of online sources. We will then have to skim through the summaries returned for each result to see if the context is relevant. For example, some results might be discussing predictions based on factors that do not include "Quantitative Easing", but we are really interested in this relationship. We can keep adding more keywords to the query to constraint it further. This may result in too much constraint and not return any relevant results. However, assuming that we have obtained a relevant result set, we now have to post-process the hitlist documents to determine what the prediction actually is (stock prices will go up, will go down or will stay the same). It would be really useful if the search engine performed this analysis for us when returning the results. Even modern search engines do not provide such a mechanism for users to explicitly indicate that they want to extract sentiment/opinion in addition to receiving the raw "lexical hit" search results. The needs of such users are acute enough that often painstaking and largely manual efforts are undertaken in order to extract sentiment scores from lexical hit results. Worse, often the user's informational need is ultimately satisfied from mined results that derive from terms, concepts or entities that are not present within the keywords specified in the query.

In this paper we present a framework for addressing this issue of specifying and processing sentiment related informational need in search engines. The framework is implemented as part of the Oracle Text product, which is an in-database information retrieval system [2].

II. Related Work

Sentiment Analysis of documents is a growing field that has seen a lot of research in the recent years. Previous work particularly relevant to our task falls naturally into two groups. The first is related to techniques for sentiment classification of natural language texts at document and topic level. The second relates to automatic extraction of sentiment metadata in information retrieval systems.

A. Sentiment Classification Techniques

Approaches for sentiment classification can broadly be divided into machine learning based approaches and lexicon based approaches. Pang *et al.* [3] compared Naïve Bayes, Maximum Entropy and Support Vector Machines (SVM)

Manuscript received February 10, 2015; revised June 2, 2015. This work is part of patent US-14/032,587

The authors are with Oracle Text, USA (e-mail: shubhro.roy@oracle.com, alexandra.czarlinska@oracle.com, asha.tarachandani@oracle.com).

DOI: 10.7763/IJKE.2015.V1.15

using combinations of unigrams, bigrams and part of speech tags etc. as features. Their work concluded that the best performance in terms of precision-recall is achieved by SVM using unigram features. Manning and Wang further compared Naïve Bayes and SVMs and concluded that Naïve Bayes outperforms SVMs for short snippets of text whereas SVMs are better for longer reviews [4]. On the other hand Turney's work explored an unsupervised approach where he extracted adjectives and adverbs from text and calculated sentiment score based on Pointwise Mutual Information (PMI) using predefined seed-words [5]. Other unsupervised approaches have used sentiment lexicons to extract opinion bearing words and compute their relative strengths [6], [7].

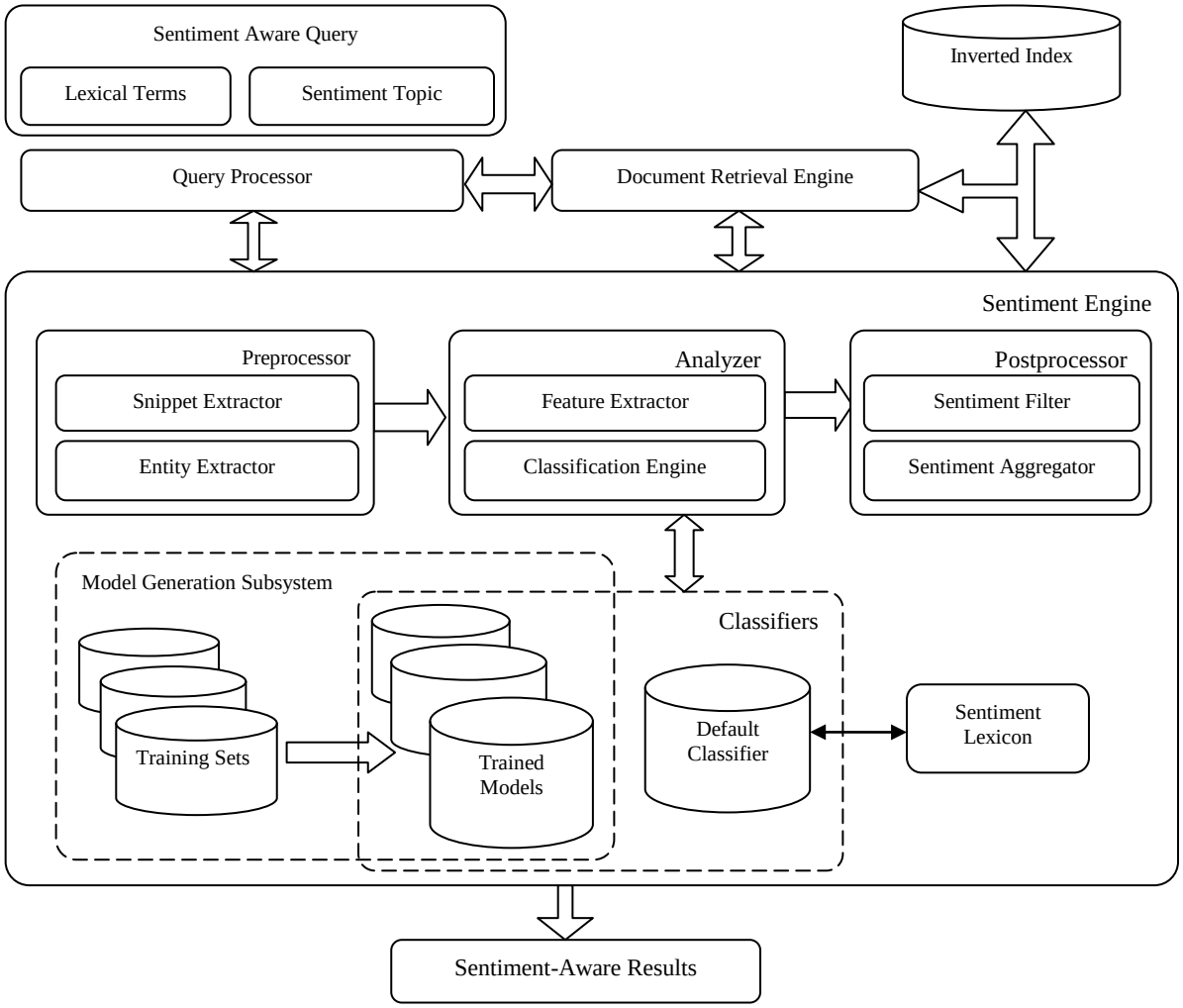


Fig. 1. System Architecture.

Most of the above mentioned work though, has focused on calculating sentiment score for the entire document. However, it is often the case that a single document has multiple subjects and the author's sentiment towards each one is markedly different. Bing Liu shows in his work that product reviews contain different sentiments associated with different product features [8]-[10]. Hovy and Kim's work involves extracting opinion bearing words from a predefined sentiment region around a specified topic and then aggregating the score for the individual words to compute the overall sentiment related to the topic [11]. Nasukawa *et al*. also use similar approach to extract sentiment scores for a given topic and then use NLP techniques to associate the extracted sentiment to the topic [12].We draw on these approaches to extract sentiment scores for the user-specified topic of interest.

B. Application in Information Retrieval Systems

In the Information Retrieval domain, there has been limited work on sentiment analysis of search results. Chelaru *et al*. [13] employed a lexicon based approach for identifying sentiment associated with web-search queries. Pera *et al*. [14] used a multiclass-SVM based approach to extract the user's informational need from the query in order to generate summaries tailored to the identified informational need. The informational need is expressed in terms of products, facets and sentiment terms identified in the user's query. Our work differs from this approach as the framework provides a mechanism for the user to directly specify that sentiment analysis is being requested for a specified topic or entity. This is critical for companies and agencies such as marketing and government that want to analyze the sentiment for specific entities.

In the past, work has been done on enriching search interfaces with additional metadata such as locations, person names, dates etc. [1]. Demartini and Siersdorfer also discuss an approach for extracting document-level sentiment metadata from search results to study diversity of opinion across different search engines [15]. Our work provides a mechanism for specifying sentiment-based informational needs via sentiment aware queries to generate

sentiment-aware results from result-set documents returned by the main query.

III. Background on Oracle Text

Oracle Text is Oracle's integrated full-text retrieval technology available in Standard and Enterprise Editions. Oracle Text provides PL/SQL APIs to allow the user to index, search, and analyze text and documents stored in the Oracle database, in files, and on the Web. Oracle Text can perform linguistic analysis of documents and search text using a variety of strategies including keyword search, contextual queries, boolean operations, pattern matching, mixed thematic queries, HTML/XML section searching and more. The search results can be rendered in various formats including unformatted text, HTML with term highlighting, and original document format. Oracle Text also supports multiple languages and uses advanced relevance-ranking technology to improve search quality [2]. The framework described in this paper is built on the existing information retrieval framework of Oracle Text.

IV. The Proposed Framework

Fig. 1 gives an architectural overview of the proposed framework. The Document Retrieval Engine subsystem is not further discussed in this paper. It is assumed to be a standard information retrieval engine that processes the user's query by looking up the inverted index to get posting-lists of non-stopword tokens present in the query. It then uses the docid and offset information in the posting-list to process any query operators present in the query and accordingly return a ranked hitlist of documents that are the best possible match to the user's query. The following sections further explain the other components of this framework.

A. Sentiment Aware Queries

Document-level sentiment scores fail to detect individual sentiments directed towards different subjects or topics in the document [12]. Rarely does a document contain opinion regarding just one subject, and even then, the opinion of the author may vary largely across the entire document. It is often this topic- specific sentiment information that the user is most interested in. For example, consider the below segments from a single camera review for Nikon:

1) The Nikon S3 has proven to be a good buy. It is affordable yet has great picture quality.
2) I hate the Lens Cap!!! There, I said it. It's probably just one of those small annoyances that keep frustrating you.
3) I think LX2 is probably one of the best, most innovative cameras on the market today, but it is too expensive.

As we can see from the above excerpts, a single review can have multiple sentiments. When we classified the entire review the document got a sentiment score of +69 (on a scale of -100 to 100) showing it as a positive review about Nikon S3 camera. But we do see that the user has some negative sentiment associated with the Lens Cap. Also the positive sentiment in the third excerpt is actually about a different camera, which may have boosted the sentiment score of the entire document.

Now consider that the user want to understand "What is the feedback on the picture quality and lens of the new Nikon S3 camera?" Specifically, the user wants the camera reviews for "Nikon S3" but the sentiment information requested is related to the "picture quality" and the "lens". There is no way to identify this directly. Using an automated query analyzer like Pera *et al.* [14] in this case, may incorrectly identify the sentiment topic as "Nikon S3" and return sentiment scores related to it rather than the "picture quality". Hence it is important in such use-cases to enable the user to provide this information directly to the search engine so that appropriate sentiment metadata can be provided as a part of the search results. Sentiment Aware Queries provide the user with this mechanism to specify a lexical query based on which the documents will be retrieved and a sentiment query based on which the sentiment information will be processed. Fig. 2 shows a sample sentiment-aware query that can be used to express the above specified informational need.

```
ctx_query.result_set('idx',`Camera AND
Nikon S3 AND SA(Nikon)=POS','
<ctx_result_set_descriptor>
  <hitlist order="SCORE DESC">
    <sentiment classifier="camera">
      <item topic="picture quality"/>
      <item topic="ABOUT(Lens)"
      radius=200/>
    </sentiment>
  </hitlist>
  <group SA="picture quality">
</ctx_result_set_descriptor>
', :rs);
```

Query Element	Description
ctx_result_set_ descriptor	Describes the user query being submitted. This is the top-level query element.
hitlist	Specifies that the top hits for the query should be returned
sentiment	Indicates that sentiment information related to the search results should be returned.
item	Specifies topic for which sentiment information should be returned.
group	Specifies group counts should be returned.

Fig. 2. Sentiment-aware query.

The query on the inverted index 'idx' is specified using the existing Oracle Text ResultSet Interface (ctx_result_set_descriptor element), which is an XML based query specification framework [2]. The new sentiment element (bolded) enables the user to specify that 1) sentiment information is being requested 2) topics or entities for which sentiment information should be extracted. For example, the search query used above for document retrieval is "Camera AND Nikon S3" and the sentiment topics are "picture quality" and "lens". Using the ABOUT operator for the term "lens" the user specifies that synonyms and stemmed versions of the term should also be used to extract sentiment information. The sentiment element also takes an attribute "classifier", which can be optionally used to provide a specially trained model for sentiment prediction. If the classifier is not specified, the sentiment engine uses the default classifier. This shall be discussed in more detail in the next section. In addition to specifying a sentiment topic, the framework also enables the user to specify post-filtering options related to

sentiment information such as group-counts and filters. In Fig. 2, the phrase "SA (Nikon) =POS" in the main query indicates that only those documents for which the sentiment score for "Nikon" is positive will be returned. The element "<group SA='picture quality'>" indicates that group counts for sentiment (positive and negative) will also be returned for the topic specified. The Query Processor parses this XML query to extract the different query terms and options specified by the user and accordingly directs the Document Retrieval Engine and the Sentiment Engine.

B. Sentiment Engine

The sentiment engine provides the core sentiment processing functionality at query time. The input to the system includes the sentiment topic extracted from the sentiment-aware query by the query parser as well as the documents retrieved by the Document Retrieval Engine for the main search query provided by the user. These documents are further processed by the Sentiment Engine to extract sentiment metadata. Finally the sentiment engine also uses the existing Inverted-Index for snippet generation and feature extraction.

1) Preprocessor

```
Procedure ExtractSnippets(docid_list, topic,
radius)
begin
for each docid in docid_list
{
 while (docid contains more topic-offsets)
 {
  toff = getoffset(docid, topic);
  start_off = toff - radius;
  end_off = toff + radius;
  AdjustBoundary(start_off, end_off);
  Add (start_off,end_off) to snippet_list
 }
 Sort snippet_list by start_off;
 for each snippet s_i in snippet_list
  {
   if (s_i.end_off > s_i+1.start_off)
     merge s_i and s_i+1
  }
}
end;

Procedure AdjustBoundary(stoff, eoff)
begin
while (gettoken(stoff) <> EOS && stoff>0)
 stoff--;
while (gettoken(eoff) <> EOS &&
gettoken(eoff)<>EOF)
 eoff++;
end;
EOS: End of Sentence token
EOF: End of File token
```

Fig. 3. Snippet extraction.

As discussed in Section II, the sentiment related to a particular topic / entity is dictated by opinion bearing terms present in the text surrounding the topic [3]. In [11] we also see that in terms of accuracy, a region of text works better than just the sentence containing the topic, for sentiment prediction. Hence, the preprocessor extracts text segments called "snippets", surrounding the topic terms specified by the user, from the documents returned by the Document Retrieval Engine. For this purpose the snippet extractor performs inverted index lookup to check if the document contains the topic keywords and if so it calculates the snippet start and end offsets based on the snippet radius and sentence boundaries. As shown in Fig. 2, the snippet radius is a parameter that the user can control do dictate how much of text surrounding a given topic term should be analyzed for sentiment processing. If this parameter is not specified we use a default value that has shown best performance in terms of accuracy on experiment datasets. The extractor also respects sentence boundaries i.e. the start and end offsets of the snippet are selected such that sentences are not chopped off in the middle. This ensures all informative sentiment bearing terms are included in the extracted snippet. The snippet extraction algorithm shown in Fig. 3 is computationally inexpensive as it has access to pre-computed offset information stored in the inverted index. This avoids the cost of actually parsing the document to extract snippets.

If a sentiment topic is not specified by the user we use a Named Entity Extractor to identify frequently occurring entities in the result-set documents and then extract snippets for those entities. Finally the extracted snippets are processed by the Sentiment Analyzer to generate sentiment scores for each snippet.

2) Analyzer

The sentiment analyzer is responsible for generating prediction scores for the snippets. If the user already specified a model to be used for sentiment prediction, the feature extractor transforms the extracted snippet into a feature vector of tf-idf scores for all non-stopword unigram tokens present in the snippet. This information can be directly computed using token-frequency information stored in the inverted index. Once the feature vectors are generated, the user-specified SVM model is used to generate probabilistic scores for both positive and negative sentiment categories. The raw probability scores are then used to generate a normalized sentiment score for the snippet, ranging for +100 to -100, indicating the degree of positivity / negativity. If the user does not specify a model, the framework uses the default classifier to compute sentiment scores.

3) Model generation subsystem

Sentiment polarities, intuitively, are dependent on domains / topics. As discussed in [5], the word "unpredictable" in a phrase like "unpredictable steering" in an automobile review indicates negative connotation while the same word in the phrase "unpredictable plot" in a movie review has positive orientation. Hence a supervised model trained on movie reviews data will not perform well for documents related to automobile reviews or financial reviews. The model generation sub-system enables the user to train domain specific models using pre-labeled training data, which can then be used during the sentiment query processing phase. The user can train multiple such models and specify the one to be used at query time using the sentiment-aware query syntax shown in Fig. 2.

For our purpose we select Support Vector Machines (SVM) to build the domain-specific models as it has been shown to have the best performance in terms of precision-recall for sentiment analysis using unigram features [3]. Since the size of the snippet can vary based on the radius value provided at

query time, we cannot use Naïve Bayes as it has been shown to have good performance only for very short segments of text [4]. The in-database SVM implementation used here is part of the Oracle Data Mining Product [16]. It uses an Active Learning approach to speedup convergence and hence model training time. The algorithm uses a small sample set of randomly selected data points from the training-data to create an initial model and then incrementally refines the decision hyperplane using the rest of the data until the model converges or the maximum allowed number of support vectors is reached. It also uses a computationally inexpensive, data-driven approach to estimates Complexity Factor and Epsilon values that result in a model with competitive accuracy and supports rapid convergence [17]. As a result the user does not have to specify these SVM parameters at training time. Using the model generation API, the user just needs to specify the database tables containing the training data and associated training-labels along with the model-name. The framework also takes care of stratified sampling and cross-validation during training phase and stores the resulting model in the database itself. This abstracts away complexities related to training an SVM model making it more accessible to users lacking data mining expertise. Please refer to the Appendix A for an example of model generation.

4) Default classifier

If the user did not specify a model to be used for sentiment prediction we used a lexicon based approach for sentiment scoring. Previous work on unsupervised sentiment classification has shown that adjectives and adverbs are good indicators of sentiment (Hatzivassiloglou, 1997, 2000 [18], [19], Turney 2002 [5]). It has also been shown that adjectives present around a given topic are indicative of sentiment related to the particular topic [11], [20]. Hence we first use a Part of Speech (POS) tagger to identify adjectives present in the snippet. We then extract the semantic score for the identified adjectives using SentiWordNet (Esuli and Sebastiani, 2005 [21]), which is a WordNet [22] based sentiment lexicon. These individual scores for the adjectives are then aggregated to compute the score for the entire snippet. Now let's consider the snippet below:

"The Picture Quality is Very Good. Yes, it does take some time getting used to, but once you familiarize yourself with everything this camera is capable of, you can achieve spectacular results"

As we can see the phrase "very good" closest to the topic "picture quality" defines the sentiment related to it. Other sentiment- bearing terms like "spectacular" actually refer to the camera in general and not just the picture quality. Hence we hypothesize that adjectives closest to the topic phrase have greater probability of bearing sentiment information related to the topic than those further away from it. This is also in agreement with the rationale behind extracting text segments surrounding the topic terms. To incorporate this factor into the aggregate sentiment score we weight the score of the individual adjectives by the distance of the adjective from the topic as shown in Equation 1.

$$score(snippet) = \frac{1}{n}\sum_{\forall adj \in snippet} \frac{SWN(adj)}{dist(adj, topic)} \quad (1)$$

Here *dist(adj, topic)* is simply the absolute difference of the token offset of the adjective and the topic fetched from the inverted index. *SWN(adj)* is the score returned by SentiWordNet for that adjective and '*n*' is the number of adjectives in the snippet. The above approach shows performance comparable to the pre-trained supervised models, in terms of accuracy.

5) Postprocessor

Finally the extracted snippets and the calculated sentiment scores for those snippets are passed to the postprocessor. This component of the framework is responsible for generating the sentiment-aware results based on the computation performed in the previous steps. If the user query had specified any filtering criteria based on sentiment, such as return only those results that have positive sentiment related to the topic, then such processing is performed at this stage by the Sentiment Filter. If the user requested for document-level sentiment scores for the topic, the post-processor aggregates the snippet level scores for the topic across the entire document. Finally if sentiment group counts are requested, then the number of positive and negative documents for the specified topic is computed from the hitlist document returned by the main query, using the document level aggregate scores.

C. Sentiment Aware Results

```
<ctx_result_set>
<hitlist>
  <hit>
    <sentiment>
       <item topic="picture quality">
        <segment>
           I was actually quite impressed with
          it. Powerful zoom, sharp lens,
          decent <b>picture quality</b>. I
          also played with some other
          Panasonic models in various stores
          just to get a better feel.
        </segment>
         <score>67</score>
        <segment>
          <b>Picture Quality</b> is Very Good.
          Yes, it does take some time getting
          used to, but once you familiarize
          yourself with everything this is a
          great camera.
        </segment>
        <score>88</score>
      </item>
    </sentiment>
  </hit>
  …
</hitlist>
<groups SA="picture quality">
  <group value="positive">
    <count> 17 </count>
  </group>
  <group value="negative">
    <count> 4</count>
  </group>
</groups>
</ctx_result_set>
```

Fig. 4. Sentiment-aware result.

The sentiment-aware results generated in response to the user query are in XML format which can be easily consumed by any front-end application to appropriately display the results. Fig. 4 shows a sample response generated by the framework for the sentiment-aware query in Fig. 2.

The hitlist element contains the "hits" for the main query submitted by the user. Since sentiment results were also requested, each hit contains a "sentiment" element which further contains the text segments extracted from that hit for the given topic and the corresponding sentiment score. In Fig. 4 we have just shown 2 of the 21 hits returned. The result also has a "groups" element which has the group counts for positive and negative sentiment related to "picture quality".

V. Experimental Evaluation

In order to assess how well the framework meets the sentiment-related informational needs of the user, experiments were performed based on 3 criteria: a) overall document-level accuracy of sentiment models, b) runtime performance of the sentiment queries c) quality of results returned by the sentiment queries. For this purpose we first describe the datasets used for this evaluation and the traditional approach based system used for comparative analysis. We then discuss the performance of the framework described in Section IV based on the above specified evaluation criteria.

A. Dataset

We have used two distinctly different datasets for evaluation. The movie reviews dataset consist of 1000 positive and 1000 negative movie reviews from IMDB first used by Bo Pang and Lillian Lee, 2004 [23]. The reviews in this dataset are significantly long. This is good for analyzing how document level sentiment can differ from snippet level sentiment related to a particular topic, as multiple distinct snippets can be extracted from each review. The other dataset used contains 1000 positive and 1000 negative camera reviews from Amazon Product Reviews dataset [24]. The reviews in this dataset are much smaller as compared to the movie reviews dataset, often containing only a few sentences. The performance on this dataset is a good indicator of how the algorithm will perform on short segments of text.

B. Traditional Approach

We discussed earlier that sentiment-aware search enables us to embed the sentiment processing within the document retrieval framework itself, which results in better performance and more accurate response to user's informational need. This is as opposed to the traditional approach where the search query returns a set of hitlist documents which are then parsed and post-processed to extract sentiment information external to the information retrieval system. Here we compare the proposed framework to a similar traditional approach in terms of processing time and quality of results. For this purpose we built a PL/SQL based external system which accepts a search query and a sentiment query. It then uses Oracle Text to fetch the hitlist documents for the search query and the sentiment topic and finally uses the same pre-trained SVM models used by the framework to predict the document level sentiment scores. Note that scores in this approach are computed for the whole document and not at the topic level.

C. Document Level Accuracy

For validating document level performance of the supervised models (SVM) we trained the model on 800 movie reviews (400 positive and 400 negative) and then submitted sentiment queries on the remaining 200 reviews (100 positive 100 negative) requesting document level sentiment scores. A similar process was repeated for the camera reviews dataset.

TABLE I: Classifier Performance

	Supervised - Movie Reviews	Supervised - Camera Reviews	Unsupervised - Camera Reviews
Precision	74.07%	87.25%	78.48%
Recall	87.76%	76.02%	63.93%
Accuracy	78.65%	75.61%	73.33%

From the performance results in Table I, we can see that the accuracy does fall a bit in the case of camera reviews, as the reviews themselves are very small (only a couple of sentences in some cases). Hence a few misleading phrases using negative words to express positive sentiment or using sarcasm could lead to incorrect prediction. This is consistent with the findings of Wang and Manning, 2012 [4].

For the unsupervised model no training was required. To test this model we indexed the same 200 camera review documents on which the supervised model was tested and submitted sentiment queries without specifying a classifier. Hence the framework uses the default SentiWordNet based classifier to generate the sentiment scores. As we can see from Table I, the unsupervised approach shows comparable performance to the supervised SVM-based model, which was specifically trained on camera reviews. We also note that currently the unsupervised approach simply aggregates the scores of adjectives extracted from the text segment. Hence in case of phrases like "the lens is not so good", the adjective "good" will still contribute a positive semantic orientation score to the overall sentiment score. To handle such cases we would also need to look at adverbs, which may boost or invert the sense of the adjective. We also do not handle sentiment inversion based on keywords like "but", "yet" etc. as can be seen in the below review:

"First I tried the 28-135 IS-pretty good. But I wasn't totally happy so returned it and ordered the 24-105 L."

The above review is actually negative but positive adjectives like "good" and "happy" would result in a positive sentiment score. Yet the model serves the purpose of providing decent out-of-the-box performance in the case where the user does not specify a domain-specific model. Adding more linguistic based approaches, similar to [25], would definitely improve performance in terms of accuracy but may have significant impact on query runtime. Finally we note that it was not possible to compare model accuracy at the snippet level as true labels are not available for the snippets (they are generated by the framework).

D. Runtime Performance of Sentiment Queries

Fig. 5 shows the query runtime for sentiment aware queries

using the framework for both the Movie Reviews and Camera Reviews datasets. In both cases we see an almost linear increase in time with an increase in the number of sentiment topics processed. In the case of movie reviews the processing time is slightly higher since those reviews are much longer compared to the camera reviews. As a result the number of snippets extracted and analyzed per document is much higher than in the case of movie reviews.

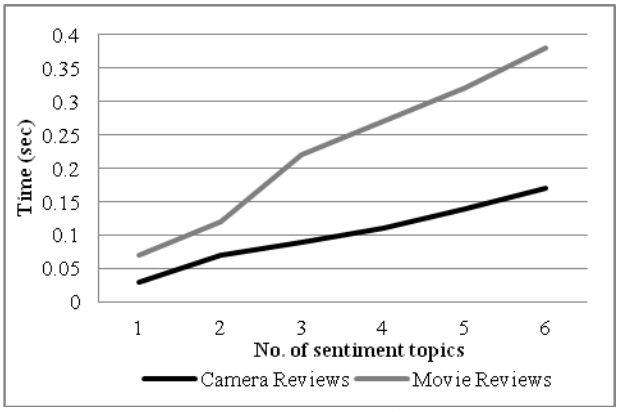
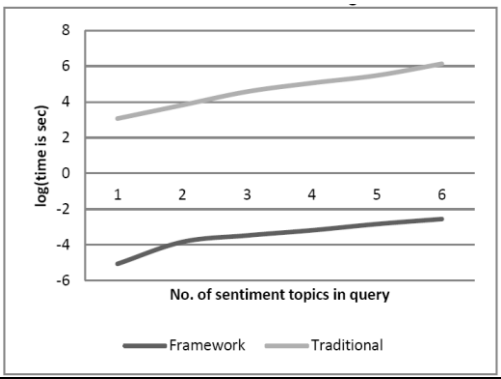


Fig. 5. Framework query performance.

Furthermore, in the case of camera reviews the driving search query returns 3257 documents whereas for the movie reviews the driving query returns only 15253 documents which also contributes to this difference in query time. Nevertheless in both cases the framework is able to process up to 6 sentiment topics under 0.5 seconds.



No. of sentiment topics	1	2	3	4	5	6
Framework (Time in sec)	0.03	0.07	0.09	0.11	0.14	0.17
Traditional (Time in sec)	8.38	14.1	23.75	33.14	44.39	69.6

Fig. 6. Traditional vs. framework – Runtime comparison.

Next we compare the camera review query processing time to the traditional approach. Fig. 6 shows the query runtime for the traditional approach using the PL/SQL based API discussed earlier. As we can see, the framework based queries are orders of magnitude faster than the traditional approach. This is mainly because in the traditional approach we first need to fetch the documents for the search query and then re-parse the hitlist documents to extract relevant features. Then the SVM model is used to generate sentiment scores. The framework approach also has direct access to offset and frequency information of keywords from the inverted index which aids in fast computation of feature vectors. Hence we see that processing the sentiment information as part of the document retrieval framework leads to better query performance.

E. Quality of Results

The main aim of the framework is to directly address the user's informational need in terms of sentiment information. Here we have compared the topic level sentiment scores returned by the framework and the document level sentiment scores computed using the PL/SQL based traditional approach.

TABLE II: FRAMEWORK SCORE (FS) VS. TRADITIONAL SCORE (TS)

Topic	Snippet	FS	TS	Outcome
script	the **script** is simply a silly and unresolved story, which is artificially stretched into a three hour long motion picture	-65	68	Much Better
Brad Pitt	the other outstanding perf is given by none other than **brad pitt** , the main actor the movie's popularity may hang upon , at least initially	88	68	Better
Lens	Why buy an expensive camera and then use a cheap **lens** like this? It's silly to spend big money on the camera and have only blurry pictures to show from it.	-75	77	Much Better
Lens	The lighting must be almost PERFECT. The **lens** does not respond well to artificial lighting.	64*	-58	Worse

*See Future Work for improvements

As seen in Table II, the document level sentiment score may not always agree with the sentiment associated with the topic of interest for the user. The first two snippets in the table are from the same document with an overall sentiment score of 68 indicating a positive orientation. But if the user was actually interested in sentiment related to the "script" of the movie, the document level score would actually be misleading as the author expresses negative sentiment towards the script. On the other hand the review speaks highly of "Brad Pitt". But this is not very evident from the document level score as it is also influenced by other negative sentiments present in the document, such as the sentiment towards the script. A similar trend can be observed in the third example as well. Hence we see that without knowledge of the sentiment topic ("script" or "brad pitt"), it is difficult to appropriately address the user's informational need using document level sentiment information. Using automated approaches to extract the informational need from the query, similar to [14], may also fail to identify the correct sentiment topic in this scenario, as user queries can have multiple entities. But the user may not be necessarily interested in sentiment related to all the entities present in the query.

The last example in the table shows a negative case, where analyzing the text segment around the specified topic actually leads to incorrect sentiment scores. The presence of a positive word "perfect" close to the topic term "lens" results in a positive score for the topic when actually the reviewer's sentiment is negative towards the lens. The word "perfect" refers to the "lighting" and not the "lens". In such cases, using subject-sentiment relationship analysis as in [12] can be helpful.

VI. Conclusion

Traditional query systems do not provide a way to express sentiment aware informational need. This is because they do not distinguish between user-supplied keywords used for document retrieval and those used to perform sentiment analysis. Yet this distinction is critical to satisfy many real-world analysis needs as shown in this paper. To address this issue, we presented an integrated framework which implements sentiment analysis on search results directly in the Information Retrieval engine at a snippet-level rather than on a document-level. Uniquely, this system allows the user to specify both query terms and separate sentiment terms, rendering the query a Sentiment-Aware query. This specification allows the engine to more efficiently and accurately analyze the sentiment request while retrieving the search hits. Furthermore, the framework gives the user control of how the results are aggregated by sentiment via Sentiment-Aware results. It also provides the flexibility to train multiple domain-specific models and to select the most appropriate one at query time. Based on the experimental results of a real-world Information Retrieval engine, we see how Sentiment-Aware queries and Results enable the user to accurately express the true informational need. We also observe that embedding the sentiment processing engine within the information retrieval system leads to great advantages in terms of query performance.

VII. Future Work

Here we discuss some enhancements that can be made to the existing framework to improve ease-of-use and query performance. The current implementation requires the user to specify which sentiment model should be used at query time. This is often not very intuitive for the user. Instead we could use an automated approach where the model is selected based on the topic or domain identified from the query or hitlist documents returned by the query. This can be achieved by using topic models similar to those discussed by Xing Yi and James Allen, 2009 [26]. In case the topic model is unable to identify a topic from the query results or if no trained model is found that matches the identified topic/domain, we can always fallback to the unsupervised model. We could also use a voting mechanism based ensemble approach, using all existing domain models, to score document of unknown domain. Such approaches have been shown to have better accuracy as compared to standalone SVM models [27]. The only caveat here is that such approaches are often computationally very expensive and hence may not be suitable for use during query processing.

The SentiWordNet based default classifier could also be replaced by a PMI based scoring mechanism as shown by Turney [28]. This approach would be specifically well suited for our framework as the PMI scores could be directly calculated using the existing inverted index. This would also ensure that the keyword sentiment scores are revenant to the domain of the document since the PMI scores are calculated based on the indexed documents.

We have also seen that sentiment expressed in the text surrounding the topic terms may not always be directed to that topic. This can often lead to false positives. Hence we intend to explore better ways to associate the extracted sentiment score to the topic specified by the user [12]. Finally as discussed in [29], adverb-adjective combinations work better than just adjectives for lexicon based sentiment scoring approaches. Hence this is also a possible enhancement that we are looking at currently.

Appendix

Fig. 7 shows an example of how the framework can be used to train domain specific models prior to query time.

```
create index docx on training_set(docs)
indextype is ctxsys.context;

exec
ctx_cls.sa_train_model( 'my_classifier',
'docx', 'id', 'train_category', 'doc_id',
'cat_id');
```

Fig. 7. Model training.

In the first step the user needs to create a standard inverted index on the training set document which can then be used to extract features. In the second step the user calls the training API with appropriate parameters to generate the Sentiment Model. Table III explains the parameters specified in the example. Once the model is created it can be used at query time using sentiment-aware queries.

TABLE III: Parameter Descriptions

Parameter	Description
my_classifier	Name of the Model
docx	Name of the index created on the training set documents.
Id	Name of the primary key of training_set table
train_category	Mapping table containing true labels for the training data
doc_id	Name of Foreign Key on train_category table
cat_id	Name of column in train_category table containing true labels.

Acknowledgment

We wish to thank Roger Ford, who is the Product Manager for the Oracle Text product for his contributions. We would also like to thank Zhen Liu for his help in conceptualizing the framework.

References

[1] P. Mika, "Microsearch: An interface for semantic search," presented at the Workshop on Semantic Search Sem Search, 2008.
[2] Oracle text: An oracle technical white paper. [Online]. Available: http://www.oracle.com/technetwork/database/enterprise-edition/11goracletexttwp-133192.pdf
[3] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in *Proc. the 2002 ACL EMNLP Conf.*, 2002, pp. 79–86.
[4] S. Wang and C. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proc. the 50th Annual Meeting of the Association for Computational Linguistics*, pp. 90–94, 2012.
[5] P. D. Turney, "Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews," in *Proc. the 40th ACL Conf.*, pp. 417–424, 2002.
[6] N. Godbole, M. Srinivasaiah, and S. Skiena, "Large-scale sentiment analysis of news and blogs," presented at the International Conference on Weblogs and Social Media, 2007.

[7] J. M. Wiebe, "Learning subjective adjectives from corpora," presented at the 17th AAAI Conf., 2000.
[8] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proc. the 10th ACM SIGKDD Conf.*, pp. 168-177, 2004.
[9] L. Zhang and B. Liu, "Identifying noun product features that imply opinions," in *Proc. the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 575-580, 2011.
[10] M. Hu and B. Liu, "Mining opinion features in customer reviews," in *Proc. the 19th National Conference on Artificial Intelligence*, pp. 755-760, 2004.
[11] S. Kim and E. Hovy, "Determining the sentiment of opinions," in *Proc the 20th COLING Conf.*, pp. 1367-1373, 2006.
[12] J. Yi, T. Nasukawa, R. Bunescu, and W. Niblack, "Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques," in *Proc. the 3rd IEEE Conf. on Data Mining (ICDM'03)*, pp. 423-434, 2003.
[13] S. Chelaru, I. Altingovde, S. Siersdorfer, and W. Nejdl, "Analyzing, detecting and exploiting sentiment in web queries," *ACM Transactions on the Web Journal*, vol. 8, no. 1, 2013.
[14] M. Pera, R. Qumsiyeh, and Y. Ng, "A query-based multi-document sentiment summarizer," in *Proc the 20th ACM-CIKM Conf*., pp. 1071-1076, 2011.
[15] G. Demartini and S. Siersdorfer, "Dear search engine: What'S your opinion about...? Sentiment analysis for semantic enrichment of web search results," presented at the SEMSEARCH Conf., 2010.
[16] Oracle data mining 11g release 2: Competing on in-database analytics. [Online]. Available: http://www.oracle.com/technetwork/database/options/advanced-analytics/odm/twp-data-mining-11gr2-160025.pdf
[17] B. Milenova, J. Yarmus, and M. Campos, "SVM in oracle database 10g: Removing the barriers to widespread adoption of support vector machines," in *Proc. the 31st VLDB Conf*., pp. 1152-1163, 2005.
[18] V. Hatzivassiloglou and K. R. McKeown, "Predicting the semantic orientation of adjectives," in *Proc. the 35th ACL Conf.*, pp. 174–181, 1997.
[19] V. Hatzivassiloglou and J. Wiebe, "Effects of adjective orientation and gradability on sentence subjectivity," in *Proc the 18th Conference on Computational Linguistics*, pp. 299-305, 2000.
[20] A. Popescu and O. Etzioni, "Extracting product features and opinions from reviews," in *Proc. the Human Language Technology and Empirical Methods in Natural Language Processing (HLT'05) Conf.*, pp. 339-346, 2005.
[21] S. Baccianella, A. Esuli, and F. Sebastiani, "SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," in *Proc. the 7th LREC Conf.*, pp. 2200-2204, 2010.
[22] G. A. Miller, "WordNet: A lexical database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39-41, 1995.
[23] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proc. the 42nd ACL Conf.*, pp. 271-278, 2004.
[24] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification," in *Proc. the 45th Annual Meeting of the Association for Computational Linguistics*, pp. 440–447, 2007.
[25] K. Dave, S. Lawrence, and D. M. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," presented at the 12th Int.WWW Conf., 2003.
[26] X. Yi and J. Allan, "A comparative study of utilizing topic models for information retrieval," in *Proc. the 31th European Conference on IR Research on Advances in Information Retrieval*, pp. 29-41, 2009.
[27] M. Whitehead and L. Yaeger, "Sentiment mining using ensemble classification models," in *Proc. the Int'l Conf. on Systems, Computing Sciences and Software Engineering (CISSE)*, pp. 509–514, 2010.
[28] P. D. Turney, "Mining the web for synonyms: PMI-IR versus LSA on TOEFL," in *Proc. the 12th European Conference on Machine Learning*, pp. 491-502, 2001.
[29] F. Benamara, C. Cesarano, A. Picariello, D. Reforgiato, and V. S. Subrahmanian, "Sentiment analysis: Adjectives and adverbs are better than adjectives alone," presented at the Int'l Conf. on Web Logs and Social Media, 2007.

Shubhro Jyoti Roy was born in Kolkata, India. He has completed his master in information systems from Carnegie Mellon University in 2013, and the bachelor in computer engineering from University of Pune, India in 2009. He is currently a senior software engineer at Oracle in Redwood Shores, California where he is part of the Oracle Text team within the Database Server Technologies Group. Previously he also worked as a subject matter expert (SME) for Amdocs Development Center India as a part of the Business Intelligence Data-Warehousing (BID) team. His prior research interest, while at Carnegie Mellon, was application of machine learning techniques for analyzing multivariate time-series data to predicting critical health conditions in patients. Currently, his primary research focus is towards application of machine learning in information retrieval systems to enhance quality of search results.

Alexandra Czarlinska was born in Warsaw, Poland. She received her Ph.D. from the Wireless Communications Group (WCL) in the Department of Electrical and Computer Engineering at Texas A&M University in 2008 and the B.A.Sc. degree in engineering science in 2002 from the University Of Toronto, ON, Canada. She is currently a primary member of Technical Staff (SMTS) at Oracle in Redwood Shores, California where she works on the Oracle Text in-database search engine. Previously she worked as a software developer for Open Text, Waterloo, Canada in the area of search engine design for enterprise content management where she patented a novel method of managing the capacity of index partitions. Her prior research focus was in the area of probabilistic modeling of wireless sensor and actuator networks and on applications of game theory to engineering systems leading to more than a dozen publications in journals and conferences. Her current focus is on improving the analytic capabilities of information retrieval systems such as through integrated sentiment analysis of the results. While at Texas A&M University, Dr. Czarlinska was a member of the IEEE and a student coordinator of the National Science Foundation's Research Experience for Undergraduates program in Electrical Engineering. While at the University of Toronto, she was the recipient of the National Scholarship Award.

Asha Tarachandani was born in New Delhi, India. She has obtained a master in computer science from UC-Berkeley in 2002, and the bachelor of technology in computer science and engineering from IIT Kanpur in 2000. She is currently a senior development Manager AT Oracle in Redwood Shores, California. Previously she has worked at Oracle as a software engineer for Oracle XML Database. Her expertise is in database techniques for semi-structured and unstructured text, including information retrieval, text analysis, document stores, XQuery, and JSON. She has received 14 patents and a Best Paper Award for DBKDA 2011 for her work at Oracle.