CONSENTO: A New Framework for Opinion Based Entity Search and Summarization

Jaehoon Choi Donghyeon Kim Seongsoon Kim Junkyu Lee
Sangrak Lim Sunwon Lee Jaewoo Kang*
Korea University, Seoul, Korea
{ jaehoon, donghyeon, seongkim, onleejk, limsangrak, sunwonl, kangj }@korea.ac.kr

ABSTRACT

Search engines have become an important decision making tool today. Decision making queries are often subjective, such as "a good birthday present for my girlfriend," "best action movies in 2010," to name a few. Unfortunately, such queries may not be answered properly by conventional search systems. In order to address this problem, we introduce CONSENTO, a consensus search engine designed to answer subjective queries. CONSENTO performs segment indexing, as opposed to document indexing, to capture semantics from user opinions more precisely. In particular, we define a new indexing unit, Maximal Coherent Semantic Unit (MCSU). An MCSU represents a segment of a document, which captures a single coherent semantic. We also introduce a new ranking method, called ConsensusRank that counts online comments referring to an entity as a weighted vote. In order to validate the efficacy of the proposed framework, we compare CONSENTO with standard retrieval models and their recent extensions for opinion based entity ranking. Experiments using movie and hotel data show the effectiveness of our framework.

Categories and Subject Descriptors

H.3.1 [**Content Analysis and Indexing**]: Indexing methods, Linguistic processing; H.3.3 [**Information Storage and Retrieval**]: Retrieval models

Keywords

entity search, consensus search, ConsensusRank, sentiment analysis, maximal coherent semantic unit

1. INTRODUCTION

Web search has become ubiquitous today. Commercial search engines have been highly effective for factual queries like "iPhone 4S release date." Users can find the answers to such queries in one of the top ranked documents. However, the current search engines fall short of giving relevant answers to subjective queries. For example, queries like "best action movies in 2010" and "thrillers with a plot twist" are not properly answered by the current engines. There, the top ranked documents may contain a list of best action movies or plot-twisting thrillers. That list, however, reflects only the opinions of document authors, rather than public sentiment.

Vertical search engines, such as Google Product Search[1] and Bing Video[2], provide domain specific query results. However, they just produce the entities whose descriptions match the query terms. Apart from the conventional search systems, there exist entity search and question-answering(QA) systems [2, 5, 4]. These systems produce direct answers to queries such as "who is the president of the House of Chanel?" and "airlines flying Boeing 747." Here, the most popular approach is to query a search engine to retrieve passages that are likely to contain candidate answers, and then extract the answers from those passages using natural language processing and machine learning techniques. Still, they just focus on factual queries that contain finite sets of true answers. Thus, these systems are not appropriate, either, for our problem context.

Opinion QA (OQA), as opposed to "factual" QA, focuses on answering opinion queries, such as "what negative opinions do people have on Hilary Benn?" and "what are the reasons for the success of the Kyoto Protocol?"[3] [1]. However, such systems still follow the standard QA processes, retrieving relevant information pieces and extracting answers from them. With only a small number of top ranked documents, it would be difficult to ensure reliable performance for the consensus search problem.

Opinion finding tasks have been run in the TREC blog track (2006-2009) in the form of retrieving blog posts that contain the most relevant opinions to the query [8, 9]. A typical query in this task is "what do people think about X?" This task focuses on finding opinion-bearing blog posts and not opinions themselves. Most participants approach this task as a re-ranking problem. In the first stage, top-k blog posts are retrieved independent of their opinionated nature using a conventional retrieval system, and in the second stage, the top-k list is reordered using an opinion detection technique. This re-ranking approach suffers from the same shortcomings as the OQA systems. It is difficult to ensure

*Corresponding author.

CIKM'12, October 29–November 2, 2012, Maui, HI, USA.

[1]http://www.google.com/shopping
[2]http://www.bing.com/videos
[3]TAC 2008 Opinion Question Answering. http://www.nist.gov/tac/

that the top-k blog posts reflect the natural distribution of opinions on the query in the blogosphere.

As explained so far, most of the previous approaches in the related problem domains are inappropriate for our problem context. For example, given a query, "good value hotels," we need to quantify positive and negative opinions on the value aspect of each hotel and rank the hotels based on a combination of them. The key challenges here are how to quantitatively summarize the opinions on an aspect of a target entity such that the quantitative summary reflects the consensus in the user opinions as closely as possible, and more importantly, how to do it all online in query time. The previous approaches fall short of answering this kind of query mainly because they postpone semantic analysis of indexed documents until query time. Relevant documents are retrieved first and then they are further processed to extract entities, opinions and so on.

Let us now turn to the formal definition of the search problem that we are to address.

Definition 1. **(Consensus Search Problem)** Given an entity set $E = \{e_1, e_2, \cdots, e_u\}$ and a query Q, suppose there exists an ideal ranking function $CR(Q, e_i)$ that would return a rank of e_i reflecting the amount of votes that e_i would have received on the web with respect to Q. The consensus search problem is then defined as the problem of locating the ranked list of entities $L = [e_{k_1}, e_{k_2}, \cdots, e_{k_u}]$ such that $CR(Q, e_{k_i}) \geq CR(Q, e_{k_j})$ for all $1 \leq i \leq j \leq u$.

In order to address this problem, we introduce CONSENTO, a consensus search engine designed to answer subjective queries. Unlike standard text retrieval systems, CONSENTO indexes Maximal Coherent Semantic Unit (MCSU), which is a maximal subsequence of words containing a single coherent meaning within a document. For example, "excellent performance, but plot was hard to follow" implies two different sentiments, or one positive sentiment (performance) and the other negative (plot). However, for a query "excellent plot," conventional text retrieval systems would find the above review relevant to the query, since their indexing unit is a document and the review document contains both of the two query terms in proximity. In order to capture the user's sentiment correctly, CONSENTO splits the review comment into two MCSU segments, and indexes them separately.

Among others, our search subsystem significantly differs, in terms of ranking, from conventional standard text retrieval models. Conventional models would produce segments that best match query terms. On the other hand, CONSENTO is designed to return entities that are most agreed upon by users with respect to the query context. In order to implement this, we introduce a new ranking model, ConsensusRank, which takes into account a user opinion, or an MCSU segment, matching a particular query as a weighted vote to the "referred to" entity.

Figure 1 shows the conceptual diagram illustrating how ConsensusRank works. When given a query, all matching segments are retrieved from the index. The retrieved segments are then grouped by their referencing entities. Finally, the scores of the segments are aggregated to compute the scores of the corresponding entities. The score of each segment is determined by multiple factors including similarity to the query terms, sentiment orientation, strength, and review quality.

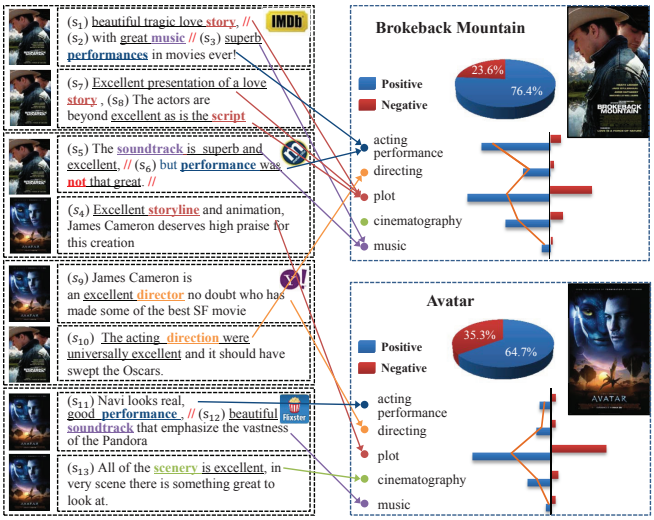

Figure 1: ConsensusRank: An Illustration.

The working prototype of CONSENTO is available at `http://CONSENTO.korea.ac.kr`.

2. RELATED WORK

In order to process consensus queries online, CONSENTO performs most of the semantic analysis early in the indexing stage. The MCSU has been introduced for this purpose[4].

An MCSU segment captures a user's opinion on an entity's aspect. Hence, CONSENTO's query processing is closely related, in essence, to aspect-based opinion mining [7].

Aspect-based opinion mining is typically carried out through four phases: aspect extraction [11], search for associated opinions [10], classification of sentiments on the aspect [12], and summarization of sentiments on the aspect [14].

However, virtually all previous works were developed with emphasis on off-line processing. Contrastingly, CONSENTO is designed to perform all of the four sentiment analysis phases online.

3. CONSENTO APPROACH

3.1 MCSU Extraction

MCSU is formally defined as follows:

Definition 2. **(Maximal Coherent Semantic Unit)** Given a review $r_i = [w_1, w_2, \cdots, w_n]$, an MCSU segment in r_i is $s_j = [w_{k_1}, w_{k_2}, \cdots, w_{k_r}]$ where $1 \leq k_1 < k_2 < \cdots < k_r \leq n$, a subsequence of r_i containing a single coherent semantic.

The task of extracting MCSU is non-trivial as it requires sophisticated natural language processing. We start with a dependency parser to analyze the sentence structure. Figure 2 shows resulting parsing examples produced with the FANSE parser [13]. We explain the segmentation processes using the examples below.

Example 1. **"The performance is excellent but the camera work is terrible."**

[4]A brief sketch of CONSENTO was presented in [3].

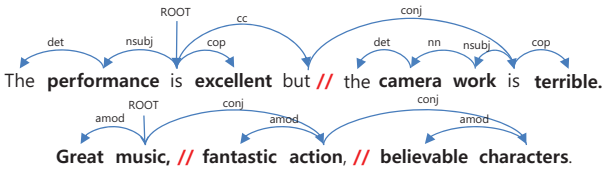

Figure 2: Segmentation Examples: Sentence (top) and Phrases (bottom).

This comment contains two distinct opinions on two different aspects of the target movie, one for performance (positive) and the other for camera work (negative). Our MCSU extraction algorithm works in two steps as follows:

1. **Locating root-level verbs.** We identify the root verb and follow coordination (cc) or conjunction (conj) dependencies to locate all root-level verbs. As shown in Figure 2 (top), two *is*'s are detected.

2. **MCSU expansion.** The first phase produces two singleton segments with each containing the word "is." In this phase, we iteratively expand each segment to MCSUs. Starting with the first "is," we conduct the repetitive expansion by including words having immediate dependencies on it, such as "performance," "excellent," and "but." In the next iteration, we further expand the segment to include "The." However, the *conj* link associated with "but" is not to be explored here, as it crosses the second segment. As there are no more links to follow up, the expansion ends, resulting in "The performance is excellent but" as the first MCSU. We repeat the process for the second segment to obtain "the camera work is terrible."

Example 2. **"Great music, fantastic action, believable characters."**

Not all user posts are well-formed sentences. Users may express their opinions freely in any format they like. The most popular form, other than sentences, is similar to the above examples. Figure 2 (bottom) shows the dependency relations obtained from the parser. The segmentation process is almost identical to the previous example. The difference is that we locate nouns instead of verbs at the root level in the first phase. The second step is identical. The resulting three MCSUs are illustrated in Figure 2 (bottom).

The details of our MCSU segmentation algorithm is given in Algorithm 1.

Once MCSUs are extracted, CONSENTO indexes them with a standard inverted index. The only change is the additional information included in each *posting*, or an entry in an inverted list. Given a document corpus, $D = \{d_1, d_2, \cdots, d_n\}$, a standard inverted index consists of two parts: the *lexicon* L containing all the distinct terms in the documents in D and the *inverted lists* of postings, with each associated to a term t_i in L. Normally, a posting consists of $< id_j, f_{ij}, [o_1, o_2, \cdots, o_{|f_{ij}|}] >$, where id_j is the ID of the document d_j that contains the term t_i, while f_{ij} represents the frequency count of t_i in d_j, and o_k's indicate the offsets of t_i in d_j. On the other hand, for the purpose of MCSU indexing, we store additional information in each posting

Algorithm 1 MCSU Segmentation Algorithm

Input:sentence #the input sentence
Output:MCSU #set of segments

procedure SEGMENT_MCSU(*sentence*)
  $V = S = G = MCSU = \phi$ // initialization
  $lastSubject \leftarrow \phi$
  $tokentable \leftarrow parse(sentence)$
  if $isVerb(ROOT)$ **then** // for a sentence
    $V \leftarrow$ EXPAND($ROOT, [\,], ALL - [cc, conj]$)
      // get the root-level verbs by following cc, conj
      // EXPAND(token, tokenExclude, depExclude) returns a phrase expanded from token by following all links except those in tokenExclude and depExclude
    for all $v \in V$ **do**
      // extract segments by expanding each root verb
      if $\neg isCC(v)$ **then** // only when dependency is conj
        $clause\ c \leftarrow$ EXPAND($v, V, [\,]$)
        if $hasSubject(c)$ **then**
          $lastSubject \leftarrow getSubject(c)$
        else
          $c \leftarrow c \cup lastSubject$ // inject missing subject
        end if
        $MCSU \leftarrow MCSU\ \cup c$
      end if
    end for
    return $MCSU$
  else if $isNoun(ROOT)$ **then** // for phrases
    $S \leftarrow getAmods(sentence)$
    $G \leftarrow getGovernors(S)$ // get tokens modified by S
    for all $g \in G$ **do**
      $clause\ c \leftarrow$ EXPAND($g, G, [\,]$)
      $MCSU \leftarrow MCSU\ \cup c$
    end for
    return $MCSU$
  end if
end procedure

such as review ID and sentiment words (hereinafter, "sentiwords"), in order to use it in the ranking stage. We formally define the structure of MCSU postings below.

Definition 3. **(MCSU Posting)**
Given a data source set $W = \{w_1, w_2, \cdots, w_y\}$, a review set $R = \{r_1, r_2, \cdots, r_x\}$, an MCSU segment set $S = \{s_1, s_2, \cdots, s_r\}$, an aspect set $A = \{a_1, a_2, \cdots, a_s\}$, a sentiword set $M = \{m_1, m_2, \cdots, m_t\}$, and a target entity set $E = \{e_1, e_2, \cdots, e_u\}$, an MCSU posting consists of $< sid_j, e_k, f_{ij}, [o_1, \cdots, o_{|f_{ij}|}], (aid_l, [mid_1, \cdots, mid_q]), rid_v, wid_z >$, where sid_j is the ID of the MCSU segment s_j that contains the term t_i; e_k is the target entity that s_i refers to; f_{ij} represents the frequency count of t_i in s_j; o_k's indicate the offsets of t_i in s_j; aid_l is the ID of the aspect a_l appearing in s_j; mid_p refers to the ID of the sentiword m_p modifying a_l; rid_v means the ID of the review r_v containing s_j; and wid_z is the ID of the data source w_z containing the review r_v.

For example, suppose IMDB (w_1) has two reviews (r_1, r_2) on Titanic (e_1). The second review (r_2) is split into three segments (s_4-s_6). The MCSU posting on the first segment (s_4) that says "touching soundtrack," (e.g., m_1 a_2, respectively) is constructed as $< s_4, e_1, (a_2, [m_1]), r_2, w_1 >$ and indexed to "touching" and "soundtrack" posting lists. Negation is also handled in our model. Our segmentation algorithm picks up negations and encodes them by inverting the polarity of the sentiword in the corresponding aspect-sentiment pair.

3.2 Scoring and Ranking

Given a query, once the matching segments are retrieved, we group them by entity and aggregate the segment scores per entity to compute the entity scores. The segment score

and the entity score (i.e., ConsensusRank) are defined formally as follows:

Definition 4. **(Segment Score)**
Given the sets W, R, S, A, M in *Definition 3*, and a query Q, the score of the segment s is defined by:

$$S_{SEG}(s,Q) = \sum_{t \in Q \cap s} \log \frac{n+1}{n_t} \times (1+q(r))^{k_1} \times sign(p(s)) \times (1+|p(s)|)^{k_2} \quad (1)$$

where n represents the total number of the segments in the corpus; n_t indicates the number of the segments containing the query term t; $q(r)$ means the review quality of the review r containing s; $p(s)$ is the mean polarity of the sentiwords in s; $sign(p(s))$ is the sign of $p(s)$; k_1 and k_2 are the parameters typically set between 0 and 5.

Definition 5. **(ConsensusRank)** Given a query Q, an entity set E, a segment set S, $S_Q = \{s_i | s_i \in S$ that match $Q\}$, $S_e = \{s_i | s_i \in S$ that refer to entity $e \in E\}$, and $S_{Q \cap e} = \{s_i | s_i \in S_Q \cap S_e\}$, the score of entity e is defined as:

$$S_{ENT}(e,Q) = \sum_{s_i \in S_{Q \cap e}} S_{SEG}(s_i,Q)/|S_e|^K \quad (2)$$

where $|S_e|$ represents the size of S_e and K is the parameter typically set between 0 and 1.

The term $|S_e|^K$ is a factor intended to normalize the imbalance in the review volume among entities. As K decreases, the scoring function prefers the popular entities that tend to acquire more reviews.

4. VALIDATION

4.1 Experimental Setup

We validate the framework using the review data sets from two different domains, movies and hotels.

Data Sets. For the movie data set, we crawled from six major sources including Amazon, IMDB, Metacritic, Flixster, Rotten Tomatoes and Yahoo Movies. For experiments, we used the reviews for the movies from 2008 to 2010, which accumulate to 130MB.

We obtained the hotel data set from Ganesan and Zhai, which they used for evaluating their work in [6]. The data set contains the reviews for the hotels in 10 major cities, which are crawled from TripAdvisor[5]. The authors also kindly provided us the corrected judgement set for our test because the original one they used in [6] has some missing hotels. Following them, we removed the hotels with less than 10 reviews from the evaluation. For more details about the data set, please refer to [6].

Relevance Judgment Generation. As no gold standard is available for consensus movie ranking, we constructed one for validation purposes. Although it would be ideal to use human judgments to construct the gold standards, it would be a very time consuming and labor intensive process to have human judges read the reviews and score how accurately the review descriptions match the returned entities. To alleviate this problem, we opted to use some authoritative lists of movies that may reflect the cinematic quality and popularity of each film.

[5]http://www.tripadvisor.com

As to cinematic quality, we resorted to the last three years' movie awards and festivals. We looked up the award histories of top 12 award ceremonies and festivals such as the Academy Awards and the Cannes Film Festival. The winner in an event receives 2 points and a nominee receives 1 point. By aggregating the points, we generated the movie rankings for five award categories for each year from 2008 to 2010. The five categories include "best movies" (e.g., best picture award), "best performance," "best cinematography," "best music," and "best screenplay." For constructing multi-aspect query judgement sets, we normalized the scores in each ranking by dividing by max score.

With regard to the popularity list, we used the statistics on box office revenues. We generated the box office rankings for three categories, including "action," "comedy," and "drama," for each year from 2008 to 2010.

Query Generation. Following the Ganesan and Zhai's approach, we asked three average users to provide three short seed queries per aspect [6]. We then generated all possible combinations of the seed queries across the five cinematic quality aspects. A query "best performance, good direction" is one of the example two-aspect queries overarching performance and direction aspects. The shortest query contains only one aspect while the longest touches all five aspects. A total of 6298 queries are generated in this process. We do not construct a multi-aspect query from the genre aspects because it is not natural. For example, there are not many movies that span over all three genres. Hence, the popularity queries are all single aspect queries. All datasets, the judgment sets, and the seed query sets are available for download at `http://infos.korea.ac.kr/consento`.

Baselines. As the baseline, we used Ganesan and Zhai's OE and QAM methods, which are the current state-of-the-art opinion-based entity ranking methods [6]. They concatenate all reviews on an entity in a single document, and index the document using a standard text retrieval system. As to query time, the OE expands the user query with a predefined set of synonyms of opinion words, and processes the expanded query as usual. The QAM is an additional improvement. It splits a query based on aspects, processes each subquery separately, and aggregates the scores from the subqueries for a final computation of rankings. Finally, the ranks of the returned documents represent the ranks of the corresponding entities. We implemented the OE and the QAM on Lucene 3.1 for evaluation.

Two standard text retrieval models are also compared: BM25 and VSM+BM (default Lucene scorer). Since they are all based on the conventional text retrieval model, we had to concatenate all reviews for a movie in one document, and index the documents, in order to test them for the purpose of entity search. The current version of CONSENTO is implemented on Lucene 3.1 and runs on an 8-node HP ProLiant DL320 cluster running Ubuntu server v10. Each node is equipped with a 4-core 2.13GHz Intel Xeon CPU, 12 GB memory and 1TB of disks.

4.2 Comparison with Baselines

Table 1 shows the nDCG@10 scores for the queries concerning cinematic quality averaged over the 2008-2010 results. The "single" and "double" means single-aspect and double-aspect queries, respectively. The "long" means queries span over 3 to 5 aspects. As illustrated in the table, CONSENTO(CR) significantly outperformed the baselines includ-

Table 1: nDCG@10 Scores of Cinematic Quality Queries.

Query	VSM+BM		BM25		CR(gain)
	base	OE(+QAM)	base	OE(+QAM)	
single	0.10	0.20(0.20)	0.20	**0.24**(0.24)	**0.40**(**70%**)
double	0.10	0.22(0.23)	0.17	0.23(**0.28**)	**0.41**(**49%**)
long	0.11	0.26(0.31)	0.20	0.26(**0.34**)	**0.48**(**39%**)
avg.	0.10	0.22(0.25)	0.19	0.24(**0.29**)	**0.43**(**53%**)

Table 2: nDCG@10 Scores of Popularity Queries.

Query	VSM+BM		BM25		CR(gain)
	base	OE	base	OE	
action	0.20	0.15	**0.25**	0.14	**0.71**(**190%**)
drama	0.07	0.10	0.1	**0.12**	**0.37**(**203%**)
comedy	0.20	0.18	**0.31**	0.21	**0.41**(**33%**)
avg.	0.16	0.14	**0.22**	0.16	**0.50**(**141%**)

ing VSM+BM (Lucene default scorer), BM25, and their OE and QAM extensions. The improvements range from 39% to 70% in the three test cases. We used the default model parameters for CR ($k_1 = k_2 = 1$ and $K = 0$) for all of our evaluation.

Table 2 shows the nDCG@10 scores for the popularity queries averaged over the 2008-2010 results. In this test, we only evaluate the single aspect queries due to the aforementioned reason. Thus, no QAM results are reported in this test. Similar to the previous test, CR outperformed all baselines with substantial margins, achieving 33% to 203% gains in the three test cases.

Table 3 shows the nDCG@10 scores for the hotel queries averaged over all 10 cities. In this case, CR underperformed the OE and QAM baselines for the long queries by 1.4%. In order to comprehend the result, we investigated what facilitates this drastic performance improvement of the OE and QAM in the hotel data set.

This happens because the OE expands an opinion word in the query such as "good" and "nice" to a predefined set of 35 positive sentiment words and an intensifier such as "very" to a collection of 21 similar adverbs. It appears that the expanded words completely dominate the aspect (or any context) words in the matching process, which leads to a production of a generic ranked result where hotels that are generally good in all aspects are always put on the top. It implies that OE may not work in domains where the variances in per-aspect preferences are high. It is certainly not a desirable property for consensus search.

In fact, the Spearman's rank correlation coefficients of hotel rankings in the judgment set range from 0.44 to 0.93 while those of cinematic quality range from -0.28 to 0.37. The rank correlations of the popularity rankings cannot be computed as they do not have much overlap.

5. CONCLUSION AND FUTURE WORK

In this study, we introduced a new search problem, which we have termed consensus search. In order to address the problem, we proposed a new search framework, and attested to its validity by implementing a prototype search engine, Consento. Consento is unique in the sense that it is the first consensus search engine that ranks entities based on the implicit votes extracted from users' online posts. In future work, we plan to expand Consento to other domains including, for example, products, events, organizations, and social issues, upon which users express their opinions. We also plan to explore broader options for ranking parameters to further enhance the performance of our system.

Table 3: nDCG@10 Scores of Hotel Queries.

Query	VSM+BM		BM25		CR(gain)
	base	OE(+QAM)	base	OE(+QAM)	
single	0.83	0.86(0.86)	0.86	**0.88**(0.88)	**0.89**(**2.0%**)
double	0.84	0.89(0.86)	0.86	**0.89**(0.89)	**0.90**(**0.4%**)
long	0.86	**0.92**(0.88)	0.850	0.91(0.91)	**0.91**(**-1.4%**)
avg.	0.85	0.89(0.87)	0.86	**0.89**(0.89)	**0.90**(**0.33%**)

Acknowledgments

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST)(No.2012R1A2A2A01014729).

6. REFERENCES

[1] A. Balahur, E. Boldrini, A. Montoyo, and P. Martínez-Barco. Going beyond traditional QA systems: challenges and keys in opinion question answering. In *COLING '10*, pages 27–35, 2010.

[2] K. Balog, P. Serdyukov, and A. de Vries. Overview of the trec 2011 entity track. In *NIST Special Publication : TREC 2011*, 2011.

[3] J. Choi, D. Kim, S. Kim, J. Lee, S. Lim, S. Lee, and J. Kang. Consento: a consensus search engine for answering subjective queries. In *WWW 2012*, pages 481–482. ACM, 2012.

[4] H. T. Dang, D. Kelly, and J. Lin. Overview of the trec 2007 question answering track. In *NIST Special Publication : TREC 2007*, 2007.

[5] G. Demartini, T. Iofciu, and A. P. de Vries. Overview of the inex 2009 entity ranking track. *LNCS*.

[6] K. Ganesan and C. Zhai. Opinion-based entity ranking. *Information Retrieval*, 15, 2012.

[7] B. Liu. Sentiment analysis and subjectivity. *Handbook of Natural Language Processing*, 2010.

[8] C. Macdonald, R. L. Santos, I. Ounis, and I. Soboroff. Blog track research at trec. *SIGIR Forum*, 44(1):57–74, 2010.

[9] I. Ounis, C. Macdonald, and I. Soboroff. Overview of the trec 2008 blog track. In *NIST Special Publication : TREC 2008*, 2008.

[10] A.-M. Popescu and O. Etzioni. Extracting product features and opinions from reviews. In *HLT '05*, pages 339–346, 2005.

[11] G. Qiu, B. Liu, J. Bu, and C. Chen. Opinion word expansion and target extraction through double propagation. *Comput. Linguist.*, 37:9–27, 2011.

[12] T. T. Thet, J.-C. Na, and C. S. Khoo. Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of Information Science*, 2010.

[13] S. Tratz and E. Hovy. A fast, accurate, non-projective, semantically-enriched parser. In *EMNLP '11*, pages 1257–1268, July 2011.

[14] J. Zhu, M. Zhu, H. Wang, and B. K. Tsou. Aspect-based sentence segmentation for sentiment summarization. In *TSA '09*, pages 65–72, New York, NY, USA, 2009. ACM.