Mining social media: tracking content and predicting behavior
Tsagkias, E.

Citation for published version (APA):
Tsagkias, M. (2012). Mining social media: tracking content and predicting behavior
In this chapter, we address the following linking task: given a news article, find social media utterances that implicitly reference it. We follow a three-step approach: we derive multiple query models from a given source news article, which are then used to retrieve utterances from a target social media index, resulting in multiple ranked lists that we then merge using data fusion techniques. Query models are created by exploiting the structure of the source article and by using explicitly linked social media utterances that discuss the source article. To combat query drift resulting from the large volume of text, either in the source news article itself or in social media utterances explicitly linked to it, we introduce a graph-based method for selecting discriminative terms.

For our experimental evaluation, we use data from Twitter, Digg, Delicious, the New York Times Community, Wikipedia, and the blogosphere to generate query models. We show that different query models, based on different data sources, provide complementary information and manage to retrieve different social media utterances from our target index. As a consequence, data fusion methods manage to significantly boost retrieval performance over individual approaches. Our graph-based term selection method is shown to help improve both effectiveness and efficiency.

3.1 Introduction

A symbiotic relation has emerged between online news and social media such as blogs, microblogs, social bookmarking sites, news comments and Wikipedia. Much of what is discussed in social media is inspired by the news (e.g., 85% of Twitter statuses are news-related (Kwak et al., 2010)) and, vice versa, social media provide
us with a handle on the impact of news events (Becker et al., 2010; Leskovec et al., 2009; Mathioudakis et al., 2010). A key ingredient is to discover and establish links between individual news articles and the social media that discuss them.

Social media utterances (such as blog posts, tweets, diggs, etc) may be linked explicitly or implicitly to a news article. In explicitly linked utterances there is a hyperlink pointing to the article; automatic discovery of such utterances is trivial. In implicitly linked utterances, however, there is no hyperlink to the source article—the utterance is not merely about the same topic as the source news article but it directly discusses the article’s content. Consider an utterance discussing the FIFA World Cup 2010 final, expressing the utterance writer’s opinion on the match. This is not considered an implicitly linked utterance; would this utterance criticize the match report given in a news article, however, then it would be an implicitly linked utterance for this news article.

The task on which we focus in this chapter is discovering implicitly linked social media utterances: for a given news article we discover social media utterances that discuss the article. Both the notion of relevance (detailed above) and the fact that, to address the task, one needs to cross from edited content to the unedited and strongly subjective language usage of user generated content, make the task challenging. To quantify the potential “vocabulary gap” (Bell, 1991; Chen, 1994; Furnas et al., 1987) we conduct an exploratory experiment. We consider a set of news articles plus a range of social media platforms; for each news article we compute the (symmetric) Kullback-Leibler (KL) divergence between the article and the social media utterances explicitly linked to it (grouped by platform) as a way of approximating the difference in vocabularies; see Fig. 3.1 for a visualization. We observe varying levels of difference in vocabulary between news and social media. The vocabularies of blog posts, Digg and Twitter seem relatively close to that of the news articles—anecdotal evidence suggests that this is due to these sources copying parts of the original news article. Moreover, the social media platforms show varying degrees of difference between their vocabularies.

When attempting to link social media utterances to a given news article, the main question is: how do we represent the article as a query? Typically, the article itself has a fielded structure (title, lead, body, headers, etc) that can be exploited (Allan, 2002; Bell, 1991; LDC, 2008). Which of these is helpful in identifying implicitly linked social media utterances? Alternatively, one can try to identify a selection of “representative” terms from the article (Allan, 2002; Becker et al., 2010; Ikeda et al., 2006). Given the noisy or unedited character of many social media utterances, the selection procedure needs to be very robust. There is a third alternative, based on the observation that there may be many social media utterances that explicitly reference a given news article. For a sample of news articles (described in Section 3.4.3), Table 3.6 displays the number of articles that are explicitly referenced by the six social media platforms considered above.
3.1. Introduction

What if we used representations of a news article generated from social media utterances that explicitly link to it?

Given these options, we approach the task of discovering implicitly linked social media utterances for a news article as a data fusion problem. We generate multiple query models for an article, based on three strategies: (i) its internal document structure, (ii) explicitly linked social media utterances, and (iii) term selection strategies. This yields ranked lists per strategy and these ranked lists are then merged using data-fusion methods. The main research question we aim to answer is:

**RQ 1.** What is the retrieval effectiveness of modeling source articles using different strategies for retrieving implicitly linked social media utterances?

We break down this research question into five sub-questions:
RQ 1/1. Does the internal document structure of a news article help to retrieve implicitly linked social media utterances?

RQ 1/2. Do query models derived from social media models outperform models based on internal document structure?

RQ 1/3. Is implicit link discovery effectiveness affected by using reduced query models that only use a limited selection of words?

RQ 1/4. How can ranked lists from individual strategies be fused to improve performance?

RQ 1/5. What is the effect on retrieval effectiveness when we use a fusion strategy dependent on the news article for which we are seeking implicitly linked utterances versus a fusion strategy that is independent of the news article?

When talking about effectiveness of a method, we consider the performance of the method in terms of recall or precision-oriented metrics. Efficiency on the other hand deals with a method’s performance in terms of speed.

Our main contributions in this chapter are the following: (a) we introduce the task of discovering social media utterances implicitly linked to a news article; (b) we offer a comparison of query models derived from (i) the document itself and (ii) auxiliary social media platforms in terms of the effectiveness of finding implicitly linked utterances; (c) we propose a robust graph-based term selection method, apply it to document and social media models, and compare the effectiveness and efficiency of these reduced models to the original models; and (d) we compare three types of late data fusion methods for combining ranked lists: (i) without training, (ii) query independent training, and (iii) query dependent training.

The rest of the chapter is organized as follows: We present our approach in Section 3.2, our models are presented in Section 3.3, our experimental setup is described in Section 3.4, we report on results and discuss our findings in Section 3.5, and conclude in Section 3.6.
3.2 Approach

Starting from a source news article, we need to identify, in a target index, utterances that reference the source article. We view this task as a data fusion problem: starting from the source article, we derive and apply query models to generate multiple queries, which are then used to retrieve utterances from the target index, resulting in multiple ranked lists that we then merge into a single result list; see Fig. 3.2. Let us motivate these steps.

![Figure 3.2: Approach to finding linked social media utterances.](image)

Most of our attention in this chapter will be devoted to the query modeling step. Importantly, in the process of identifying social media utterances that reference a given source news article, we are crossing genres, from news to social media. When crossing genres, the vocabulary gap between source article (“the query”) and target utterances (“the documents”) is wider than within genres, especially when one of the genres involved is a social media genre (Weerkamp et al., 2009). To bridge the gap, we follow multiple alternative routes: starting from the source article, we consider multiple query modeling strategies, i.e., ways of arriving at a query to be fired against the target index. First, we consider different representations of the source news article itself. It is a semi-structured document that features elements such as title, lead and body. Derived representations such as the named entities mentioned in the article and quotations from interviews can also be used to represent the article and generate queries. Second, to help generate queries that represent the source article we also use auxiliary social media. Intuitively, to
bridge between the language usage of the source article and that of the utterances in the target index, we can exploit social media where the two types of language usage are, to some degree, mixed. E.g., a Digg story usually consists of the news article title and summary (edited content) and the user comments (unedited content), tweets mix the article title (edited) with the twitterer’s opinion/comment (unedited).

The textual representations from which queries are derived may be quite long as compared to, for example, article titles. E.g., when focusing on the source news article, the entire title and body of the article can be used as a query (Metzler et al., 2005); such long queries, however, are costly to process and may introduce noise and cause topic drift. For this reason, we identify and extract terms that are discriminative and characteristic of language usage pertinent to the source article (or auxiliary social media) and use these to derive a query.

In the retrieval step, we submit queries representing the source news article to an index of social media utterances, and retrieve ranked lists for each of these queries.

In the fusion step, we use late data fusion methods (Shaw and Fox, 1993; Snoek et al., 2005) to merge results lists produced by alternative query modeling methods. For the methods that support weighted fusion, we investigate two approaches for weight optimization: query independent and query dependent. In the former approach, the system learns weights for each query model from a training set so a given metric is maximized, and then these weights are used for fusing ranked lists in response to future articles. In the latter approach, weights are learned per source article (“query”) so the given metric is maximized for an article-specific training ground truth.

3.3 Methods

We describe the methods used in addressing the three steps identified in the approach outlined in Section 3.2: retrieving social media utterances, query modeling and data fusion.

3.3.1 Retrieval model

For the retrieval step, we use a language modeling approach. We compute the likelihood of generating a news article \(a\) from a language model estimated from an utterance \(u\):

\[
P_{lm}(a|u) = \prod_{w \in a} P(w|u)^{n(w,a)},
\]  

(3.1)
3.3. Methods

where \( w \) is a query term in \( a \), \( n(w, a) \) the term frequency of \( w \) in \( a \), and \( P(w|u) \) the probability of \( w \) estimated using Dirichlet smoothing:

\[
P(w|u) = \frac{n(w, u) + \mu P(w)}{|u| + \mu},
\]

(3.2)

where \( \mu \) is the smoothing parameter, \( |u| \) is the utterance length in words, and \( P(w) \) is the probability of \( w \) in the collection.

We impose two constraints on our content-based model expressed in (3.1). The first is on the publication date of utterances potentially discussing the source news article. The second is on the “quality” of utterances being retrieved. Both are modeled in a probabilistic fashion so they can be incorporated in our content-based model.

As to the first constraint, we want to favor utterances published close to the source news article, mainly due to the volatility of the news; most social media utterances are generated around the news article publication date (Leskovec et al., 2009). Given a date range \( t \) of length \( |t| \) in days, an utterance can or cannot appear in \( t \), therefore:

\[
P_{\text{date}}(u|t) = \begin{cases} 
1 & \text{if } u \text{ occurs in } t \\
0 & \text{otherwise},
\end{cases}
\]

(3.3)

where \( r \) is a time unit in \( t \), \( n(u, \cdot) \) is the number of utterances occurring in \( r \) or in \( t \). We want to avoid discarding potentially relevant utterances that occur outside \( t \), while still favoring those published in \( t \). Therefore, we follow the language modeling paradigm and derive an estimate for \( P_{\text{date}}(u|t) \) based on Dirichlet smoothing:

\[
\hat{P}_{\text{date}}(u|t) = \frac{1 + \mu P(u)}{n(u, t) + \mu},
\]

(3.4)

where \( \mu \) is a smoothing parameter as in (3.2), and \( P(u) = 1/n(u) \) is the a priori probability of an utterance to occur anytime and \( n(u) \) is the total number of utterances in the collection.

Our second refinement of the retrieval model aims to account for adversarial social media utterances and for utterances that do not provide informative context for the article. We incorporate the credibility factors introduced in (Weerkamp and de Rijke, 2012) as quality indicators. Specifically, we implement the following topic independent factors on the level of utterances: comments, emoticons, post length, pronouns, shouting, and spelling; Table 3.1 shows the model for each factor. All factors are given equal importance and are put together for the estimation of a global credibility prior probability \( P_{\text{cred}}(u) \) for an utterance \( u \):

\[
P_{\text{cred}}(u) = \frac{1}{|F|} \sum_{f \in F} p_f(u),
\]

(3.5)
Table 3.1: Models for individual credibility factors. \( |u| \) is the utterance length in words, \( n(X, u) \) is the number of \( X \) for utterance \( u \), where \( X = \{ r, e, o, z, l \} \), and \( r \) is comments, \( o \) is pronouns, \( e \) is emoticons, \( z \) is capitalized words, and \( l \) is misspelled (or unknown) words.

\[
\begin{align*}
 p_{comments}(u) & = \log(n(r, u)) \\
 p_{emoticons}(u) & = 1 - n(e, u) \cdot |u|^{-1} \\
 p_{post\_length}(u) & = \log(|u|) \\
 p_{pronouns}(u) & = 1 - n(o, u) \cdot |u|^{-1} \\
 p_{shouting}(u) & = 1 - n(a, u) \cdot |u|^{-1} \\
 p_{spelling}(u) & = 1 - n(m, u) \cdot |u|^{-1}
\end{align*}
\]

where \( F = \{ \text{comments, emoticons, post\_length, pronouns, shouting, spelling} \} \).

Finally, the content-based, recency and credibility models are combined through their geometric mean in one score for ranking an utterance \( u \) given a source news article \( a \) and a date range \( t \):

\[
\text{Score}(u, a, t) = \sqrt[3]{P_{tm}(a|u) \cdot P_{date}(a|t) \cdot P_{cred}(u)}
\] (3.6)

### 3.3.2 Query modeling

Most of our algorithmic contributions concern query modeling: building a representation of news article \( a \) to be used for retrieval (see (3.1)). We explore three families of query models, for which we consider (i) the source news article itself as a “generator” of query models, (ii) social media as such a generator, and (iii) “reducing” the sources from which we generate query models to single out target terms.

**Exploiting the source article** Obviously, the source news article itself is an important source of information for creating query models that represent it. News articles typically feature a title, lead and body as structural elements. The title is indicative of the article’s main topic and summarizes the article. The lead consists of a few sentences, gives insight on what will follow and includes the main actors of the article. The body is the main content. Following the probabilistic model in (Metzler et al., 2005), the contents of these structural elements are mapped to queries in a straightforward manner: we use the entire contents of a selected element. For article title, we tested the effectiveness of using exact phrases for modeling, however, plain title content outperformed exact phrases and, hence, we use the plain title content to model title.

In addition to structural elements, we use two extra features as a source for query modeling: named entities and quotations. A great majority of articles
Table 3.2: Query models grouped by source; in addition, THRank, our query reduction method, is applied to the following models: full, digg, and nyc; see the end of this section for a detailed description of THRank.

<table>
<thead>
<tr>
<th>Query Model</th>
<th>Source</th>
<th>Elements</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Exploiting the source article</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>Article</td>
<td>Title</td>
</tr>
<tr>
<td>lead</td>
<td>Article</td>
<td>Lead</td>
</tr>
<tr>
<td>body</td>
<td>Article</td>
<td>Body</td>
</tr>
<tr>
<td>metadata</td>
<td>Article</td>
<td>Author (byline), news agent</td>
</tr>
<tr>
<td>ne</td>
<td>Article</td>
<td>Named entities</td>
</tr>
<tr>
<td>quote</td>
<td>Article</td>
<td>Quotations</td>
</tr>
<tr>
<td>full</td>
<td>Article</td>
<td>Title and body</td>
</tr>
<tr>
<td><strong>Exploiting social media</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>digg</td>
<td>Digg</td>
<td>Title, description and comments</td>
</tr>
<tr>
<td>delicious</td>
<td>Delicious</td>
<td>Title, tags and their frequency</td>
</tr>
<tr>
<td>twitter</td>
<td>Topsy</td>
<td>Tweet</td>
</tr>
<tr>
<td>nyc</td>
<td>NYTC</td>
<td>Comment title and body</td>
</tr>
<tr>
<td>wikipedia</td>
<td>Wikipedia</td>
<td>Full article</td>
</tr>
<tr>
<td>blogposts</td>
<td>Blogs</td>
<td>Feed item in RSS</td>
</tr>
</tbody>
</table>

Refer to and discuss people, organizations, and locations. Given a news article $a$, we identify named entities in $a$ by extracting sequences of capitalized words. Quotations are text passages from interviews with people inside the article and as such are likely to remain intact throughout information spread (Leskovec et al., 2009). This characteristic renders them viable surrogates for an article. Starting from the two extra features, we arrive at query models by constructing exact phrases from the named entities and the quotations.

As a final step, we model article metadata, consisting of the byline that represents authorship, and the news agent. The byline consists of the first and last name of the author. For the news agent, we create a basic list of potential synonyms by examining how social media refer to the news agent. For example, New York Times is mapped with three synonyms: “New York Times,” NYTimes, NYT. Content from the byline is combined with list of synonyms to produce the final query.

Table 3.2 (top) lists query models derived from the source article.

**Exploiting social media** We consider a second family of query models, obtained from social media platforms that explicitly link to the source article. Examples include Digg stories (that have a URL), tweets that include a URL, etc. Consequently, it is possible to track a source news article to social media utterances via its URL.
The idea is to create query models by aggregating content from a range of social media sources, for two reasons:

1. not all sources cover all news articles with the same intensity;
2. different social media may exhibit different language usage around the same source article.

By sampling content from multiple social media sources we increase the possibility of capturing the creativity in the language usage. We use a small number of social media platforms with frequent explicit links to news articles: Digg, Delicious, Twitter and NYT Community (NYTC); see Section 3.4 for details. We also use content from blog posts that hyperlink to a source article and Wikipedia articles relevant to the article (Weerkamp et al., 2009).

Data harvested from social media platforms that explicitly links to a source news article is used as follows for the purposes of query modeling. Similarly to how we modeled internal structure elements, we use the entire contents from all elements in a source to model the news article. E.g., for a Digg story that links to a news article, we take all text from the story title, from the story description and from all comments, if any, attached to the story. For blog posts that include a hyperlink to the article, we consider the text of the post in the blog’s feed. For Wikipedia, we use the source article’s title to retrieve the ten most relevant Wikipedia articles from a Wikipedia index and use their content to model the news article.

Using social media for query modeling purposes raises issues. First, accumulating content from multiple blog posts and Wikipedia articles can lead to noisy queries. We reduce the model size by applying a graph-based term selection method (see below). Second, looking at other social media platforms, some news articles are “comment magnets,” accumulating thousands of comments. Third, with platforms that allow for the creation of hierarchical discussion threads, the relevancy of a comment to the source news article is dependent on its level in the thread. To limit potential topical noise, we perform comment selection (dependent on the platform) based on comment metadata. Next, we look at two methods for ranking comments for Digg and NYTC.

For a Digg comment $dc$, we consider the number of positive ($up$) and negative ($down$) votes, the number of replies ($replies$) to the comment and the depth ($level$) of the comment in the thread:

$$\text{Rank}(dc) = \frac{(replies + 1) \cdot (up - down)}{level} \quad (3.7)$$

The formula rewards comments with a high number of positive votes that triggered further discussion ($replies$) and that are more likely to be about the article than about other comments ($level$).
3.3. Methods

For a NYT comment \( nc \), we consider the number of recommendations \( rec \), and whether \( nc \) was selected from the editors \( se \):

\[
\text{Rank}(nc) = 2 \cdot (se + 1) \cdot rec
\]  

(3.8)

where \( se \) is a binary variable of value 1 when the comment is selected by the editors and 0 otherwise. The formula biases comment selection to highly recommended comments that are boosted further when selected from the NYT editors.

Table 3.2 (bottom) lists query models derived using social media.

**Reduced query models**  So far, we have used any and all the data identified for a data source above as “the query model generated from the source.” As a consequence, these query models (when viewed as lists of words) may be lengthy, which may have a negative impact on retrieval efficiency and potentially also on effectiveness; see Table 3.6 (top half) for the average query length per source. Next, we aim to identify and extract terms that are discriminative, either for the source news article at hand or for the discussion surrounding it. To this end we introduce THRank (“TextHitsRank”), a variation of TextRank (Mihalcea and Tarau, 2004). TextRank and other graph-based ranking methods are based on the idea of “recommendation,” where the importance of a vertex within a word-graph is computed using global information recursively drawn from the entire graph. Our modifications to TextRank are three-fold: how the graph is constructed, the scoring algorithm, and the cutoff threshold for the returned terms. We discuss these in turn.

To construct a directed (word) graph for a document, the text is tokenized and stemmed and multi-word named entities are collapsed into a single word. Unlike TextRank (where only nouns are considered for constructing the graph), we use all terms due to the low recognition accuracy of nouns in noisy text (Dey and Haque, 2009). For each token a vertex is created and an edge is added between tokens that co-occur within a window of two words. Intuitively, the edges are weighted according to the number of occurrences of a pair of tokens in the text. Words at sentence boundaries are not connected to avoid accidental recommendations.

We are not only interested in the most discriminative words, but also in their context. For this purpose, instead of the PageRank algorithm used by TextRank, we use the HITS algorithm, which makes a distinction between “authorities” and “hubs” (Kleinberg, 1999), for scoring. In our setting, the authority score determines how important a word is for the article (proceeded by how many words) and the hub score reflects the word’s contribution to the article’s context (how many words follow it).

We use a document-dependent threshold for which terms to select: from each set (authorities or hubs), we only return terms whose score is of the same magnitude as the highest scored term.
3. Linking Online News and Social Media

Table 3.3: Data fusion methods used in the chapter.

<table>
<thead>
<tr>
<th>Method</th>
<th>Gloss</th>
</tr>
</thead>
<tbody>
<tr>
<td>combMAX</td>
<td>Maximum of individual scores</td>
</tr>
<tr>
<td>combMIN</td>
<td>Minimum of individual scores</td>
</tr>
<tr>
<td>combSUM</td>
<td>Sum of individual scores</td>
</tr>
<tr>
<td>combMNZ</td>
<td>combSUM × number of nonzero scores</td>
</tr>
<tr>
<td>combANZ</td>
<td>combSUM ÷ number of nonzero scores</td>
</tr>
<tr>
<td>WcombSUM</td>
<td>weighted sum of individual scores</td>
</tr>
<tr>
<td>WcombMNZ</td>
<td>WcombSUM × number of nonzero scores</td>
</tr>
<tr>
<td>WcombWW</td>
<td>WcombSUM × sum of individual weights</td>
</tr>
<tr>
<td>RR</td>
<td>Round-robin</td>
</tr>
<tr>
<td>RR-W</td>
<td>Round-robin weighted</td>
</tr>
</tbody>
</table>

In Section 3.5, we apply THRank to the following models: full, digg, nytc, wikipedia, and blogposts (Table 3.2).

### 3.3.3 Late fusion

Different query models potentially give rise to different ranked result lists. To arrive at a single merged result list, we use late data fusion methods. In particular, we consider the methods listed in Table 3.3; see (Shaw and Fox, 1993) for a survey of these and other methods.

Let $N$ be the set of all ranked lists $n_i$ resulting from different query models. Let $s_{n_i}(a, u)$ be the score of an utterance $u$ (from the target index) given a source news article $a$, $w_{n_i}$ a weight assigned to $n_i$ and $N_{ret}$ a subset of $N$ consisting of ranked lists that returned $u$. Then, combMAX considers the highest score from $N$, combMIN considers the lowest score from $N$, WcombSUM sums up all scores factored by their weight (He and Wu, 2008):

$$score_{WcombSUM}(a, u) = \sum_{i=1}^{\lfloor N \rfloor} w_{n_i} \cdot s_{n_i}(a, u)$$

if $w_{n_i} = 1$ (for all $n_i$), it becomes combSUM. WcombWW is similar to WcombSUM except that final scores are multiplied by the sum of weights of the runs that returned the utterance:

$$score_{WcombWW}(a, u) = \sum_{m \in N_{ret}} w_m \times \sum_{i=1}^{\lfloor N \rfloor} w_{n_i} \cdot s_{n_i}(a, u)$$

for the special case where $w_m = 1$ (for all $m$), we get WcombMNZ. If we further assume $w_{n_i} = 1$ (for all $n_i$), we arrive at combMNZ. combANZ is similar to
3.4 Experimental setup

combMNZ but final scores are averaged over the number of runs that return the utterance $|N_{ret}|$:

$$score_{combANZ}(a, u) = \frac{1}{|N_{ret}|} \cdot \sum_{i=1}^{N} s_{n_i}(a, u)$$

Round-robin (RR) chooses one utterance from each ranked list, deleting any utterance if it has occurred before. Weighted round-robin (RR-W) is similar except that not all ranked lists are available at each round. Each ranked list is assigned a sampling frequency, defining every how many rounds it will be sampled.

Normalization of scores between ranked lists is required before producing the final rankings (Montague and Aslam, 2001). A standard practice is to first normalize the document scores per run and then merge them:

$$s_{normed, n_i}(a, u) = \frac{s_{n_i}(a, u) - \min(s_{n_i}(a))}{\max(s_{n_i}(a)) - \min(s_{n_i}(a))}.$$

We also consider a second normalization method, based on z-scoring, inspired from work in topic detection and tracking (Allan, 2002):

$$s_{z-score, n_i}(a, u) = \frac{s_{n_i}(a, u) - \mu}{\sigma},$$

where $\mu$ is the mean of the document score distribution for source news article $a$ in ranked list $n_i$, and $\sigma$ is the standard deviation.

3.4 Experimental setup

We present our research questions, experiments, dataset and evaluation method. For the purpose of finding social media utterances that reference individual news articles, we choose to focus on a single target collection in our experimental evaluation, namely the blogosphere. Nothing depends on this particular choice, though. Our choice is based on the observation that blogs, unlike many other social media, are not limited to a single dominant platform like Digg or Twitter. Content found on individual social media platforms can be biased according to the platform’s user demographics.

3.4.1 Experiments

To answer our research questions in the beginning of this chapter (see p. 31), we conduct two sets of experiments, aimed at (i) query modeling and (ii) late fusion.
3. Linking Online News and Social Media

Performance of three families of query models In this set of experiments we answer RQs 1/1–1/3. For each of the three families (document structure, social media, and reduced models) we construct queries, and submit them to an index of blog posts. We measure the performance of each model individually, and compare the results. Analysis of the results reveals differences in performance between the individual models, and the families of models.

Performance of three late fusion types The second set of experiments is aimed at answering RQs 1/4 and 1/5. Here, late fusion techniques are applied to the ranked lists produced by the individual models. We experiment with 10 fusion methods from three types: (i) no training required, (ii) query independent training, and (iii) query dependent training. Finally, we test the utility of two different score normalization methods.

3.4.2 Data set and data gathering

The data set that we use as our target social media collection is the Blogs08 collection provided by TREC; the collection consists of a crawl of feeds, permalinks, and homepages of 1.3M blogs during early 2008–early 2009. This crawl results in a total of 28.4M blog posts (or permalinks). We only used feed data, the textual content of blog posts distributed by feeds and ignored the permalinks. Two main reasons underly this decision: (i) our task is precision-oriented and benefits from a clean collection; and (ii) using feed data requires almost no preprocessing of the data. Extracting posts from the feed data gave us a coverage of 97.7% (27.8M posts extracted). As a second preprocessing step we perform language detection and remove all non-English blog posts from the corpus, leaving us with 16.9M blogs posts. Our index is constructed based on the full content of blog posts.

Our news article dataset is based on the headline collection from the top stories task in TREC 2009. This is a collection of 102,812 news headlines from the New York Times and include the article title, byline, publication date, and URL. For our experiments we extended the dataset by crawling the full body of the articles.

As auxiliary collections used in our query modeling experiments, we use data gathered from the following five platforms:

Digg: A collaborative news platform where people submit URLs that they find interesting.\(^1\) We collected 19,608 Digg stories corresponding to the same number of articles. On average each story is associated with 26 comments.

Delicious: A social bookmarking site, where people can store the addresses of web sites they want to keep.\(^2\) We collected 7,275 tagged articles with an

average of 3 unique tags per article, summing up to 3,906 unique tags.

**Twitter:** We use Topsy, a real-time search engine that indexes content from Twitter, a microblogging platform where people can submit short snippets of text 140 characters long.\(^3\) We collected tweets that mention 21,550 news articles, with each article being mentioned in 3 tweets on average.\(^4\)

**NYT Community:** A web service from New York Times for retrieving comments registered on their site.\(^5\) We collected comments for 2,037 articles with an average of 150 comments per article.

**Wikipedia:** The collaborative online encyclopedia. We use the Wikipedia dump that is included in the Clueweb09 collection,\(^6\) containing almost 6 million pages.

### 3.4.3 Evaluation

The ideal ground truth for our task would consist of tuples consisting of news articles and social media utterances. As a proxy, we follow (Geva and Trotman, 2010; Mihalcea and Csomai, 2007; Milne and Witten, 2008) and use items that are explicitly linked to a given news source. We then remove the explicit links and test our link generation methods by examining to which extent they succeed at identifying those explicit links. The reason for choosing this evaluation scheme is twofold: (i) the generation of such ground truth is cheaper than having human assessors judge whether a blog post is about a news article, and (ii) in this chapter we are interested in examining the relative effectiveness of the suggested approaches, not in absolute numbers.

Our ground truth is assembled in two phases. First, for each news article we find blog posts that include the article’s URL. Second, for each discovered blog post we look for other blog posts that include its URL. The process continues recursively until no more blog posts are discovered. For our experiments we sample headlines with more than ten explicit links and where social media possibly plays a role. For each news article, we take the temporally first five explicitly linked blog posts for using them in modeling. The remaining blog posts form the article’s ground truth. This selection procedure results in 411 news articles with an average of 14 explicitly linked (“relevant”) blog posts per article.\(^7\)

---

\(^4\)Topsy limits access to the ten most recent tweets for a URL. Consequently, the reported average might not reflect reality.
In our experiments we use the Indri framework (Metzler and Croft, 2004). Each experimental condition returns the top 1,000 results. We report on standard IR measures: recall, mean reciprocal rank (MRR), mean average precision (MAP), and r-precision. Statistical significance is tested using a two-tailed paired t-test and is marked as ▲ (or ▼) for significant differences for α = .01, or △ (and ▽) for α = .05.

3.4.4 Weight optimization for late fusion

For late fusion methods that allow for weighted fusion, we estimate a weight $w_{n_i}$ for each ranked list $n_i$ using query independent and query dependent approaches.

**Query independent weight optimization** Given a set of news articles and a set of ground truth assessments, we seek weights that maximize MAP over a set of source articles. For this, we conduct two fold cross-validation and split our ground truth in two sets of equal size: training (205 articles) and testing (206 articles). First, we learn weights that maximize MAP on the training set and then use these for evaluation on the test set. For estimating $w_{n_i}$, we follow He and Wu (2008). First, for each ranked list $n_i$ in the training set, the MAP score $\text{map}_{n_i}$ is computed. Then, $\text{map}_{n_i}$ is used as weight for $n_i$ in the test set: $w_{n_i} = \text{map}_{n_i}$. He and Wu suggest that the weight for the best individual run should be factored several times its MAP score. Fig. 3.3 shows that, in our setting, increasing the weight of the best individual run hurts performance.

**Query dependent weight optimization** Given a news article and a ground truth, we seek weights $w_{n_i}$ that maximize average precision (AP). Since the weights are dependent on the query, the ground truth for training and testing should be different. For building the training ground truth, we look for good surrogates of implicitly linked blog posts to use as proxy. For this purpose, for an article’s training ground truth, we consider the temporally first five explicitly linked blog posts. The testing ground truth is kept the same as in query independent optimization for the results to remain comparable. In the training step, the system learns weights such that the blog posts in the training ground truth rank at the top. Then, in the testing step, we report on MAP for the testing ground truth. For estimating $w_{n_i}$, we use maximum AP training and line search (Gao et al., 2005), where $w_{n_1}, \ldots, w_{n_n}$ is considered a set of directions in the range $[0, 1]$. We move along the first direction in steps of 0.2 so that AP is maximized; then move from there along the second direction to its maximum, and so on. We cycle through the whole set of directions as many times as necessary, until AP stops increasing.

For query dependent and query independent fusion, we combine all available ranked lists except from the blogpost model. The later is excluded because it
exploits the same explicitly linked blog posts to model the news article with those used as training ground truth in query dependent fusion. Also, for models that have reduced counterparts, we select the one performing the best. This selection leads to 11 models: title, lead, ne, quote, metadata, full, digg-comm, delicious, nyt-comm, twitter, and wikipedia-graph.

3.5 Results and analysis

We report on our results from the experiments in Section 3.4 for query modeling and late fusion and conduct an analysis on our findings.

3.5.1 Query modeling

We turn to the results of our query modeling approach; each paragraph discusses one of the research questions at the start the chapter (see p. 31). Next, we perform an analysis of the results to gain more insight.

RQ 1/1. Internal document structure vs. article title

Our baseline is set to the query model derived from an article's title only. This choice is supported by two considerations: first, the article's title is the most compact representation of the entire article and second, the article's title was chosen in prior research for ranking news headlines according to their mentions in the blogosphere (Macdonald et al., 2010).

Table 3.4 (top) reports on results from models derived from the article's internal document structure. The best performing model is the one that uses the full article, namely, content from the article's title and body. The high performance of full is possibly due to blog posts picking up different aspects of the article that are not available in more compact representations such as title and lead. Both ne and quotes show a precision-enhancing effect over the baseline, at the cost of a drop in recall. Depending on the application, these representations could be an efficient alternative to full.

RQ 1/2. Comparison of social media models over internal document structure models

Turning into models derived from social media, Table 3.4 (middle) shows that digg-comm, the model from Digg using only five comments (see (3.7)), is performing the best among all social media models and significantly improves over title on all metrics. delicious shows a high recall possibly due to the nature of tags which are more likely to capture the article's theme rather than precisely identify it.
Table 3.4: System performance for retrieving blog posts relevant to a source article using credibility priors and models derived from internal document structure and social media, and their reduced counterparts using THRank. Significance tested against baseline (title).

<table>
<thead>
<tr>
<th>runID</th>
<th>Recall</th>
<th>MRR</th>
<th>Rprec</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>0.4033</td>
<td>0.3812</td>
<td>0.1488</td>
<td>0.1069</td>
</tr>
<tr>
<td>(A) title</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Model based on:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal document structure</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(B) lead</td>
<td>0.2937</td>
<td>0.3339</td>
<td>0.1726</td>
<td>0.0886</td>
</tr>
<tr>
<td>(C) metadata</td>
<td>0.2206</td>
<td>0.1449</td>
<td>0.0466</td>
<td>0.0275</td>
</tr>
<tr>
<td>(D) ne</td>
<td>0.3739</td>
<td>0.4967</td>
<td>0.1787</td>
<td>0.1290</td>
</tr>
<tr>
<td>(E) quote-#1</td>
<td>0.2732</td>
<td>0.5101</td>
<td>0.1741</td>
<td>0.1259</td>
</tr>
<tr>
<td>(F) full</td>
<td>0.5919</td>
<td>0.6058</td>
<td>0.3190</td>
<td>0.2509</td>
</tr>
<tr>
<td>Social media</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(G) delicious</td>
<td>0.4122</td>
<td>0.2883</td>
<td>0.0875</td>
<td>0.0677</td>
</tr>
<tr>
<td>(H) digg</td>
<td>0.1108</td>
<td>0.1250</td>
<td>0.0433</td>
<td>0.0315</td>
</tr>
<tr>
<td>(I) digg-comm</td>
<td>0.5797</td>
<td>0.5490</td>
<td>0.2508</td>
<td>0.2010</td>
</tr>
<tr>
<td>(J) nytc</td>
<td>0.0072</td>
<td>0.0020</td>
<td>0.0008</td>
<td>0.0006</td>
</tr>
<tr>
<td>(K) nyt-c-comm</td>
<td>0.0949</td>
<td>0.0644</td>
<td>0.0160</td>
<td>0.0125</td>
</tr>
<tr>
<td>(L) twitter</td>
<td>0.1543</td>
<td>0.1150</td>
<td>0.0545</td>
<td>0.0445</td>
</tr>
<tr>
<td>(M) blogposts</td>
<td>0.1233</td>
<td>0.1289</td>
<td>0.0424</td>
<td>0.0298</td>
</tr>
<tr>
<td>Reduced using THRank</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(N) full-graph</td>
<td>0.4524</td>
<td>0.5254</td>
<td>0.2177</td>
<td>0.1681</td>
</tr>
<tr>
<td>(O) digg-graph</td>
<td>0.2799</td>
<td>0.2552</td>
<td>0.0890</td>
<td>0.0681</td>
</tr>
<tr>
<td>(P) nytc-graph</td>
<td>0.0691</td>
<td>0.0300</td>
<td>0.0122</td>
<td>0.0077</td>
</tr>
<tr>
<td>(Q) wikipedia-graph</td>
<td>0.0412</td>
<td>0.0142</td>
<td>0.0030</td>
<td>0.0020</td>
</tr>
<tr>
<td>(R) blogposts-graph</td>
<td>0.4170</td>
<td>0.4448</td>
<td>0.1727</td>
<td>0.1362</td>
</tr>
</tbody>
</table>

In general, social media models using all available content from the underlying source perform worse than models based on article internal structure. This is possibly due to noise found in user generated content, a claim supported by the improved performance of digg-comm and nytc-comm (which exploit only a subset of available content using comment selection methods) over their respective baselines (using all comments).

RQ 1/3. Reduced query models using THRank For most query models, THRank leads to improved performance. Among all reduced models, full-graph and blogposts-graph perform the best; both show significant improvements on precision-oriented metrics, without hurting recall. For full-graph, when compared to full,
3.5. Results and analysis

Table 3.5: System performance for articles present in either twitter or nytc-comm and the baseline.

<table>
<thead>
<tr>
<th>runID</th>
<th>Recall</th>
<th>MRR</th>
<th>Rprec</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>110 common topics between baseline and Twitter</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>0.4165</td>
<td>0.3876</td>
<td>0.1667</td>
<td>0.1192</td>
</tr>
<tr>
<td>twitter</td>
<td>0.5741</td>
<td></td>
<td>0.4206</td>
<td>0.2024</td>
</tr>
<tr>
<td><strong>197 common topics between baseline and NYTC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>0.4091</td>
<td>0.3576</td>
<td>0.1293</td>
<td>0.0951</td>
</tr>
<tr>
<td>nytc-comm</td>
<td>0.1979</td>
<td></td>
<td>0.1345</td>
<td>0.0334</td>
</tr>
</tbody>
</table>

performance drops by 33% due to a significant reduction (97%) in query size. Given the low noise levels in edited text, THRank sees to discard more words than required. For blogposts-graph, performance increases by an order of magnitude following a 80% reduction in query size. For blog posts, THRank manages to remove noise and select terms helpful to retrieval. In both cases, THRank offers a good balance between efficiency (shorter models are less computationally expensive) and effectiveness.

Next, we take a close look at the query modeling results and we perform an analysis in four directions: (i) uniqueness, (ii) silent models, (iii) NYT comments, (iv) THRank, and (v) opinionatedness of articles.

**Uniqueness:** Besides looking at the results in terms of precision and recall, we also explore the uniqueness of runs: how many linked utterances are identified by one model in the top $X$, and not by any of the other models? We do so for the top 10 results and top 1,000 results. First, we observe that all models have unique results; Second, quotes-#1 is able to capture unique results in the top 10, whereas delicious does so in the top 1,000. Finally, title, full, and digg-comm capture most unique results.

**Silent models:** Table 3.6 shows that certain models, like Twitter and NYT, are silent for a large number of queries, and it is therefore difficult to assess their utility when looking at overall performance. Table 3.5 reports on the performance for articles present in the baseline and the social media model (twitter or nytc-comm); results show that the Twitter model significantly outperforms the baseline on recall metrics.

**NYT comments:** An interesting observation from the results is the low performance of nytc and nytc-comm, despite their strong connection to the source news article (see Tables 3.4 and 3.5). This strong connection could be the reason for their failure: news comments are usually displayed on the same page as the news article and come after it. Consequently, when people comment, there is no need to explain what news event they are referring to, give context to their
opinion, or write full names of entities. This leads to a lack of discriminative and descriptive terms for that news article in the comments, potentially explaining the poor performance of news comments-based query models.

**THRank**: Why does THRank help performance for blogposts but not for other social media? Comment threads are prone to topic drift as the discussion goes on, while explicitly linked blog posts are more likely to be focusing on one topic, that of the article. Topical focus is likely to enable THRank in one case to reduce noise and improve performance and in the other to capture the “general theme” of the discussion which can be far away from what triggered the discussion initially.

The same can hold for models using comment selection methods which are found to outperform their THRank counterparts. Highly recommended comments are more likely to reflect what is also published in the blogosphere. On the other hand, when THRank is ran on all available data from a source it proves unable to capture accurately discriminative terms for the news article, although it returns more terms for digg and nyt (lower reduction ratio, see Table 3.6).

**Opinionatedness**: We measure how opinionatedness of news articles affects the performance of individual models. In order to do so, we split our ground truth of 411 articles into 131 opinionated and 280 non-opinionated articles depending on whether the article title contains the term “OPED” (e.g., columns, editorials).

We perform a two-tailed independent t-test between the opinionated and non-opinionated scores for each model. For most models, performance is stable across the two articles types with full and digg-comm performing the best. In terms of recall, six of the models drop significantly, when crossing from non-opinionated to opinionated articles. title is amongst them, possibly due to static titles assigned to opinionated articles, which usually consist of the column or editorial name with only few additional terms. We also notice that digg-comm sees the highest recall on non-opinionated articles over all models, whereas this is full for opinionated articles. An interesting case is the metadata model for opinionated articles: When compared to non-opinionated articles, the recall shows a large significant increase, which is due to blog posts referring to the article author’s name (and agency).

### 3.5.2 Late fusion

We start by addressing the remaining research questions not answered yet. After that we take a closer look at the results.

**RQ 1/4. Query independent late fusion** We experiment with 10 fusion methods and two document score normalization methods for the combination of 11 individual models; see Section 3.4.4. Table 3.7 shows that the best performing method in terms of MAP is WcombMNZ, which yields statistically significant...
results and analysis

Table 3.6: Number of queries per news article model, and their average length for query terms, phrases, and both.

<table>
<thead>
<tr>
<th>runID</th>
<th># queries</th>
<th>Average query length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Terms</td>
</tr>
<tr>
<td><strong>Query based on: Internal document structure</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>title</td>
<td>411</td>
<td>8</td>
</tr>
<tr>
<td>lead</td>
<td>411</td>
<td>23</td>
</tr>
<tr>
<td>metadata</td>
<td>411</td>
<td>8</td>
</tr>
<tr>
<td>ne</td>
<td>410</td>
<td>0</td>
</tr>
<tr>
<td>quote-#1</td>
<td>398</td>
<td>0</td>
</tr>
<tr>
<td>full</td>
<td>411</td>
<td>912</td>
</tr>
<tr>
<td><strong>Query based on: Social media</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>delicious</td>
<td>411</td>
<td>47</td>
</tr>
<tr>
<td>digg</td>
<td>411</td>
<td>1,476</td>
</tr>
<tr>
<td>digg-comm</td>
<td>411</td>
<td>225</td>
</tr>
<tr>
<td>nyt</td>
<td>197</td>
<td>15,048</td>
</tr>
<tr>
<td>nyt-comm</td>
<td>197</td>
<td>288</td>
</tr>
<tr>
<td>twitter</td>
<td>111</td>
<td>48</td>
</tr>
<tr>
<td>wikipedia</td>
<td>409</td>
<td>6,912</td>
</tr>
<tr>
<td>blogposts</td>
<td>408</td>
<td>617</td>
</tr>
<tr>
<td><strong>Query based on: Reduced using THRank</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>full-graph</td>
<td>411</td>
<td>27</td>
</tr>
<tr>
<td>digg-graph</td>
<td>395</td>
<td>37</td>
</tr>
<tr>
<td>nytc-graph</td>
<td>197</td>
<td>131</td>
</tr>
<tr>
<td>wikipedia-graph</td>
<td>409</td>
<td>117</td>
</tr>
<tr>
<td>blogposts-graph</td>
<td>408</td>
<td>23</td>
</tr>
</tbody>
</table>

improvements over the full model. WcombSUM, WcombWW, combSUM and comb-MNZ perform similar to, but slightly less than WcombMNZ, and RR-W outperforms RR.

We investigated the effect on MAP for a range of scale factors of the best individual run (full), when using z-scoring and linear normalization of document scores. Fig. 3.3 illustrates that, in our setting, WcombMNZ with z-scoring and the scale factor set to 2 achieves the best MAP among all fusion methods.

RQ 1/5. Query dependent late fusion For this experiment we use the best performing late fusion method from RQ 1/4: WcombMNZ with z-scoring. The goal here is to learn weights that maximize average precision for a training ground truth.

From Table 3.7 we can see that query dependent fusion significantly outper-
3. Linking Online News and Social Media

Figure 3.3: MAP scores for combination of 11 individual runs when increasing the weight of the best individual run for WcombMNZ and WcombSUM methods and using linear and z-score normalization of document scores.

forms full, but performs slightly worse than query independent fusion. One reason for this can be that the nature of relevant blog posts is evolving as we move farther in time from the source article publication date.

Next, we proceed with an analysis of: (i) query dependent vs. independent fusion, (ii) an oracle run, and (iii) early fusion.

**Query dependent vs. independent fusion:** In theory, query dependent fusion was expected to outperform other methods because of how weights were optimized. For each individual article, weights were estimated to maximize average precision. However, query independent fusion showed to perform better. The two methods differ in the training set. For query independent fusion each article in the training set was on average associated with 14 blog posts. For query dependent fusion, weights were estimated for a ground truth of 5 blog posts per article. It is, therefore, interesting to explore the utility of a larger sample of explicitly linked blog posts as training ground truth or to seek time dependent evolution patterns in the weights assigned to each ranked list.

**Oracle run:** For each source article we take the ranked list produced from the best performing model according to average precision, and combine these into a final “oracle” run. Since we only use one model per source news article and no mixture of models, this run does not achieve the maximum performance possible. Still, the oracle run gives an indication of what scores are achievable. Comparing the performance of the oracle run (Table 3.7) to WcombMNZ, the best performing query independent fusion method, we observe that the latter arrives remarkably close to the oracle run.

**Early fusion:** Belkin et al. (1995) conducted thorough experiments comparing
Table 3.7: System performance for query independent fusion using 10 late fusion techniques on the test set using z-score normalization and combining 11 individual runs for the best scale factor (2) of full. Query dependent fusion results are reported for the best fusion method. Significance tested against full. Results from an oracle run and early fusion are also reported.

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall</th>
<th>MRR</th>
<th>Rprec</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>0.5860</td>
<td>0.6196</td>
<td>0.3323</td>
<td>0.2522</td>
</tr>
<tr>
<td><strong>Query independent</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>combMAX</td>
<td>0.7214</td>
<td>0.5871</td>
<td>0.2820</td>
<td>0.2283</td>
</tr>
<tr>
<td>combMIN</td>
<td>0.3308</td>
<td>0.0766</td>
<td>0.0195</td>
<td>0.0131</td>
</tr>
<tr>
<td>combSUM</td>
<td>0.7194</td>
<td>0.6083</td>
<td>0.3202</td>
<td>0.2665</td>
</tr>
<tr>
<td>combMNZ</td>
<td>0.7265</td>
<td>0.6130</td>
<td>0.3252</td>
<td>0.2722</td>
</tr>
<tr>
<td>combANZ</td>
<td>0.6821</td>
<td>0.4547</td>
<td>0.1574</td>
<td>0.1256</td>
</tr>
<tr>
<td>WcombSUM</td>
<td>0.7190</td>
<td>0.6141</td>
<td>0.3317</td>
<td>0.2772</td>
</tr>
<tr>
<td>WcombMNZ</td>
<td>0.7248</td>
<td>0.6123</td>
<td>0.3422</td>
<td>0.2788</td>
</tr>
<tr>
<td>WcombWW</td>
<td>0.7169</td>
<td>0.6129</td>
<td>0.3315</td>
<td>0.2723</td>
</tr>
<tr>
<td>RR</td>
<td>0.7328</td>
<td>0.3990</td>
<td>0.2095</td>
<td>0.1664</td>
</tr>
<tr>
<td>RR-W</td>
<td>0.7298</td>
<td>0.3999</td>
<td>0.2358</td>
<td>0.1882</td>
</tr>
<tr>
<td><strong>Query dependent</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WcombMNZ</td>
<td>0.7011</td>
<td>0.6148</td>
<td>0.3277</td>
<td>0.2646</td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Oracle</td>
<td>0.6388</td>
<td>0.7727</td>
<td>0.3645</td>
<td>0.3141</td>
</tr>
<tr>
<td>Early fusion</td>
<td>0.5331</td>
<td>0.5356</td>
<td>0.5220</td>
<td>0.1956</td>
</tr>
</tbody>
</table>

performance of early and late fusion techniques. They found that by combining individual models at query time performance increased compared to individual models. As a check we use the best performing model from each family of query models and combine them in a query. For the reduced models of Wikipedia and blog posts, each term is weighted according to its hub or authority score. The performance of this run (again, Table 3.7) is less than the best individual run (i.e., full) and the query independent and dependent fusion methods (i.e., WcombMNZ). The lower performance is likely due to noise brought in after combing all models and suggests that term selection and term weighting methods on the combined query hold potential for improving retrieval effectiveness.

3.6 Conclusions and outlook

Most of the algorithmic contributions of this chapter lie in modeling news articles for the task of discovering implicitly linked social media utterances. We studied the
retrieval effectiveness of multiple query models that exploit content from individual elements in article's internal structure and from explicitly linked utterances from six social media platforms. Our experiments provide the following answers to the questions raised in p. 31 at the beginning of this chapter:

**RQ 1/1.** Does the internal document structure of a news article help to retrieve implicitly linked social media utterances?

Experimental evidence shows that query models based on the entire article perform the best. However, query models from social media bring in previously unseen utterances.

**RQ 1/2.** Do query models derived from social media models outperform models based on internal document structure?

Query models trained on anchor text from explicitly linked blog posts are interesting to explore, however our current experimental setup constrains us from further investigating their utility.

**RQ 1/3.** Is implicit link discovery effectiveness affected by using reduced query models that only use a limited selection of words?

To reduce the potential topical shift in our query models from using the entire contents of a news articles or the content from explicitly linked sources, we introduced THRank, an unsupervised graph-based method for selecting the most discriminative terms from each query model. We found that content selection helps to improve both effectiveness and efficiency.

**RQ 1/4.** How can ranked lists from individual strategies be fused to improve performance?

We studied the effect of combining ranked lists from individual query models and we experimented with ten late data
3.6. Conclusions and outlook

fusion methods and two document score normalization methods. We found that fusion methods significantly improve retrieval when document scores are normalized using z-scoring.

RQ 1/5. Does it help effectiveness when making the fusion strategy dependent on the news article for which we are seeking implicitly linked utterances?

Query independent weight optimization helped \( W_{combMNZ} \) to outperform all individual and fusion runs and to achieve performance remarkably close to an oracle run.

Summing up, as to RQ 1 as a whole—What is the retrieval effectiveness of modeling source articles using different strategies for retrieving implicitly linked social media utterances?—, we found that full articles combined with term selection and normalized result fusion achieved very high levels of effectiveness.

In future work we plan stricter recency conditions to our retrieval model, study the potential of query dependent fusion in more detail, compare our models to typical IR approaches such as BM25F, and experiment with additional target indexes such as Twitter. Results from this work and its future extensions lay the ground work for discovering social media utterances related to a topic of a group of news stories.

In this chapter we looked at a key aspect of content tracking, that of identifying links between news articles and social media utterances that discuss them. Our approach to the problem was based on IR where a news article for which we want to discover implicitly linked utterances is the query, and the utterances are documents to be retrieved from a target index. The IR method we used builds on the query likelihood model which expects the query length to be a fraction of the length of the documents in the index—which is the typical case for web queries. However, during our experiments we found that many blog posts are of similar length to the source news articles, which violates the assumption in the query likelihood model. In the next chapter, we revisit this assumption and suggest a fix for retrieval scenarios where queries and documents are of similar length.