Learning-to-Rank for Real-Time High-Precision Hashtag Recommendation for Streaming News

Bichen Shi, Georgiana Ifrim, Neil Hurley
Insight Centre for Data Analytics
University College Dublin, Ireland
{bichen.shi, georgiana.ifrim, neil.hurley}@insight-centre.org

ABSTRACT
We address the problem of real-time recommendation of streaming Twitter hashtags to an incoming stream of news articles. The technical challenge can be framed as large scale topic classification where the set of topics (i.e., hashtags) is huge and highly dynamic. Our main applications come from digital journalism, e.g., for promoting original content to Twitter communities and for social indexing of news to enable better retrieval, story tracking and summarisation. In contrast to state-of-the art methods that focus on modelling each individual hashtag as a topic, we propose a learning-to-rank approach for modelling hashtag relevance, and present methods to extract time-aware features from highly dynamic content. We present the data collection and processing pipeline, as well as our methodology for achieving low latency, high precision recommendations. Our empirical results show that our method outperforms the state-of-the-art, delivering more than 80% precision. Our techniques are implemented in a real-time system\(^1\), and are currently under user trial with a big news organisation.

Categories and Subject Descriptors
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

Keywords
learning-to-rank, dynamic topics, social indexing, news, hashtag recommendation

1. INTRODUCTION
A recent report by the BBC imagining the Future of News states that “In a democracy, news is the essential public service” [14]. Recently, social media platforms such as Twitter have taken a central role in the consumption, production and dissemination of news [35]. Twitter currently has about 240 million active users and receives more than 500 million tweets a day, a quarter of which are tagged with hashtags [31]. Hashtags are keyword-based tags, describing the content of a tweet, for example #taiwan, #transasia, #ge235 were used for tweets describing a recent plane crash in Taiwan. They tend to appear spontaneously around breaking news or developing news stories, and are a way for news followers to connect to a particular story and community, to get updates in real-time. However, an approach to real-time high-precision hashtag recommendation for incoming news is currently missing. Most existing solutions do not achieve high enough precision for practical use (precision of 38\% [6]), and are not able to deliver these recommendations in real-time. In this paper, we focus on the problem of recommending relevant hashtags to a stream of news articles as they are published online.

A key requirement of our approach is that the recommended hashtags should be highly specific to the tracked news and actively used by Twitter users. This comes in contrast to existing approaches that recommend hashtags based on the topics covered by the target content, regardless of whether users actively engage with that hashtag, e.g., topic modelling methods [20, 15]. In addition, many hashtags do not contain topical keywords related to the target content, but are rather emerging statements related to certain aspects of the news story. For example, for a recent attack at a coffee shop in Sydney, Australia, one emerging tag was #illridewithyou. Getting to this tag quickly from news related to those events, allowed journalists or the general public to connect to that Twitter community. It is also not enough to simply track the most frequently used hashtags, since general hashtags such as #news and #world, although used frequently, are not specific to particular news stories.

Our problem is related to topical classification, with the additional challenge that topics (i.e., hashtags) are in the scale of tens of thousands a day, and highly dynamic. In this paper, we show how it can be modelled as an Information Retrieval (IR) learning-to-rank problem. Nevertheless, classic IR approaches assume a static document collection and dynamic user-queries, while in our case both documents and tags come as streams, and the set of relevant tags for a given document continuously changes. For example, the news story headline “BREAKING: Plane crashes in southern France” was automatically tagged by our approach with the hashtags “#germanwings #airbusa320crash #france” within minutes of being retrieved from the RSS feed. Subsequently, the tags for this article got updated

---

\(^1\)Insight4News real-time system: http://insight4news.ucd.ie/insight4news/article_list/
every 5 mins with more specific recommendations such as “#4u9525 #a320” driven by the convergence of the crowdsourced content and our ranking algorithm.

In our framework, a news article plays the role of the query in classic IR, and the hashtags (represented by tweets using those tags), play the role of documents. The goal is to retrieve and rank hashtags for each article, whereby most hashtags are not relevant to a given news article. We propose a two-step approach: (1) a pre-ranking step based on automatic query formulation to connect each article to the hashtag stream and retrieve candidate hashtags and (2) a learning-to-rank approach to learn a model of relevance for article-hashtag pairs. Since both articles and hashtags are streaming, we need to address challenges regarding low-latency feature computation to predict hashtag relevance and continuously update the recommendations for each article in the stream. To this end, we investigate a set of time-aware features for describing the examples used to train and test the learning-to-rank model. We work with real-world data collected from existing RSS news feeds which we connect to relevant Twitter streams. To give a feel for the data scale, over a time period of 12 months, with about 1,000 news articles processed each day, we averaged more than 1 million tweets and 26,000 hashtags per day (used in at least 10 tweets). Many of these hashtags may be only relevant for 24h or 48h, e.g., #illridewithyou, #icantbreathe, #worldcancerday, with longer stories running over weeks or months having a more stable set of hashtags, e.g., #ebola, #ebolavaccine, #greexit.

Contributions. We summarise our main contributions as follows:

1. We formulate real-time hashtag recommendation for news as a learning-to-rank problem.
2. We investigate time-aware features for high-precision hashtag relevance ranking.
3. We conduct a real-life study of the impact of our recommendations on news engagement and discuss applications to social indexing of news.

Our paper is structured as follows. In Section 2 we discuss related work. We describe our methodology and the proposed features in Section 3. In Section 4 we present the evaluation setting and experimental results analysing our approach and the state-of-the-art. In Section 5 we discuss our real-world experiment evaluating the impact of hashtag recommendation on news engagement, and present an exciting new application, social indexing, a form of real-time crowdsourced tagging of news. We conclude in Section 6 and present directions for future work.

2. RELATED WORK

Hashtag Recommendation for Tweets. Prior work focusing on hashtag recommendation for tweets relies on topic modelling on static datasets. The work of [7, 22] builds Naive Bayes, KNN or SVM classifiers for hashtags, where a hashtag is seen as a category and the tweets tagged with that hashtag as labeled data for that category. Hashtag recommendation for tweets can be adapted to recommendation for news, by treating the news headline as a rich tweet. As we show in our experiments, this approach is overwhelmed by the data scale, sparsity and noise characteristics of tweets.

Most other approaches focus on topic modelling with PLSA and LDA [20, 15, 12, 6, 16]. For example [6] fits an LDA model to a set of tweets in order to recommend hashtags. They combine the LDA model with a translation model, to address the vocabulary gap between tweets and hashtags. LDA-type approaches face drastic challenges regarding both scalability and accuracy of recommendation, where either hashtags that are too general are recommended, e.g., #news, #life, or ones that are not used at all by the Twitter users, since the focus is on recommending hashtags solely driven by the topic of tweets [6].

Hashtag Recommendation for News. There is little prior work focusing specifically on hashtag recommendation for news. The approach in [33] relies on a manual user query to retrieve related articles, which are then clustered to create a topic profile. Similarly a hashtag profile is created from tweets collected from a set of manually selected accounts. This approach then recommends hashtags with a similar profile to a topic/cluster profile, without regard to user engagement with the hashtag, since the experiments are done on a static collection. The work in [27] proposes an approach that updates hashtag recommendations once daily, while the emphasis in our work is on real-time recommendation.

Real-time Tag Recommendation. Related to our work are also recent approaches to real-time tag recommendation for streaming scientific documents and webpages [29, 28]. In that work the set of tags is assumed to be static, and fairly small, which facilitates a lot of pre-processing steps. In our scenario, both articles and tags are continuously streaming into the system, and the set of hashtags is very large and dynamic (i.e., have a variable relevance lifecycle), which makes the problem more challenging.

Learning to Rank. In classic IR learning-to-rank approaches, a ranked list of documents is returned for a user query. The set of documents is typically assumed to be static, which allows for clever indexing. The set of queries is dynamic and a pre-processing step is used to produce an initial document ranking, followed by a re-ranking step using machine learning [17, 26]. Depending on the input representation and loss function, learning to rank algorithms can be categorised [19] as pointwise [18, 23], pairwise [1, 30, 10] and listwise [34, 25, 2]. Although listwise/pairwise approaches are commonly used, they are not suitable for our problem setting, due to the nature of our data and efficiency constraints. On average, there are about 1-5 relevant hashtags for each article (as identified by our labelling study involving journalists), so the key concern is to quickly identify the relevant few, in every time slot. From an efficiency point-of-view, the computational complexity of listwise and pairwise approaches is usually high [2, 11], making them less suitable for a real-time setting. Pointwise approaches were shown to be efficient and well suited for binary relevance labels [18, 26], therefore we take this approach in our work. We show how to model our problem in a learning-to-rank framework for dynamic settings. We compare our method to existing topic modelling approaches: Naive Bayes [7], Support Vector Machines [36] and Latent Dirichlet Allocation [6].

3. LEARNING-TO-RANK FOR REAL-TIME HASHTAG RECOMMENDATION

In this section we discuss the proposed learning-to-rank framework and the methodology for computing time-aware features for the relevance model.
3.1 Learning-to-Rank Approach

In an IR setting, a system maintains a collection of documents $D$. Given a query $q$, the system retrieves a subset of documents $d \in D_q$ from the collection, ranks the documents by using a ranking model $f(q, d)$, and returns the top ranked documents. In a learning-to-rank framework, the $f(q, d)$ model is constructed automatically using supervised machine learning techniques [13]. There are two ways of creating the labeled training data: via human judgements or via deriving relevance-labels based on automatically collected data, such as search-logs. We focus on the first approach and work with binary relevance labels (e.g., relevant/irrelevant).

In our hashtag recommendation setting, the query $q$ is extracted from an individual article $a \in A$, where $A$ is a stream of news. The document collection is a stream of hashtags $H$, extracted from a stream of tweets $T$. For the reasons stated in Section 2, we take a pointwise learning-to-rank approach by transforming the ranking problem into a classification problem [13, 18]: First, a subset of hashtags $H_a$ is retrieved for article $a$ through a hashtag-sharding method explained in Section 3.2. Then, for each article-hashtag pair $(a, h)$, $h \in H_a$, we create a feature vector $x$, with label $y \in \{0, 1\}$ ($y = 1$ if the hashtag is relevant for the article). Given $m$ training examples $S = \{x_i, y_i\}, i = 1, 2, ..., m$, we construct a classifier $f(x) = y$ to predict $y$ for any feature vector $x$ of an arbitrary article-hashtag pair.

To address the dynamic aspect of our problem (i.e., hashtags and articles come as streams, and the relevance of hashtags to articles is time-dependent since the hashtag representation changes due to the arrival of new tweets), we extract time-aware features $x_\gamma$ as shown in Section 3.3. We write $f(x_\gamma) = y_\gamma$ to denote that the feature vector is dependent on time, while the classification function $f$ is not. We employ two sliding time windows to transform the dynamic environment to a static one, at current time point $t_n$. The global time window $\gamma = [t_n - 24h, t_n]$ corresponds to the past 24h from the current time $t_n$, while the local time window $\lambda = [t_n - 4h, t_n]$, with $t_n$ the publishing time of article $a$, is an article-dependent time window, where $t_n \leq t_n$. The local time window restricts the computation of features to a local (in time) tweet subset. The choice of window parameters is justified empirically and by the application domain, e.g., in the news life-cycle most news either get updated (and become new articles) or are ignored after 24h [9]. We explain how we use these time windows in Section 3.2 and 3.3.

3.2 Sharding the Tag Stream

Similar to query sharding in classic IR, we identify a set of tweets associated with each article, which we call the article’s tweet-bag $T_a$. All hashtags contained in $T_a$ are the article’s tag shard $H_a$. Starting at some initial time $t_0$, at time-step $t_n$, the system carries out the following actions, where the interval between time-steps is 5mins:

1. Read RSS feeds, download articles, and extract keyphrases from each article (query formulation).
2. Pool the keyphrases for all articles published within the global time window $\gamma$.
3. If a retrieved tweet from $T$ contains at least one keyphrase of an article $a$, append it to $T_a$.
4. From each tweet-bag $T_a$, extract the hashtags and assign them to this article-shard $H_a$.
5. Compute the feature vector $x$ of the article-hashtag pair $(a, h)$. Feed $x$ to the relevance classifier, get hashtag recommendation $f(x)$.

Sharding the tag stream enables the retrieval of tags likely to be relevant to the article, as well as quick computation of feature vectors for the article-hashtag pairs. We investigate several methods for keyphrase extraction and show the impact of 3 such methods in our experiments (Section 4.2). The goal is to extract article-keyphrases to maximize the retrieved number of tweets (a form of tweet Recall) and the content similarity of the retrieved tweets to that article (a form of tweet Precision). The procedure for extracting keyphrases is as follows. Since news are written in an inverted pyramid form (i.e., the article focus is presented in the beginning) we focus on the pseudo-article formed of headline, subheadline and first sentence of each article, and tokenize and POS-tag that text. In the best performing method, only nouns are selected, giving priority to proper nouns over common nouns, as a light form of entity detection. Single keywords are then paired and long proper nouns are broken down into term-pairs. Finally, these pairs are ranked based on the average tf-idf of the individual terms, with term-frequency computed from the article body and inverse-document-frequency computed from the article collection within $\gamma$. The top-5 pairs are used as the keyphrases of the article. Table 1 and 2 show an example article text and the procedure for extracting keyphrases.

Table 1: Example article text used for extracting keyphrases.

| Headline | Vladimir Putin in good health, insists Kremlin |
| Subheadline | Spokesman says Russian president’s handshake is strong enough to ‘break hands’ |
| First Sentence | Kremlin spokesman Dmitry Peskov said on Thursday that President Vladimir Putin is in good health, but could not say when he would next appear in public. |

Table 2: Example process for extracting article-keyphrases.

<table>
<thead>
<tr>
<th>Original keywords</th>
<th>Paired keywords</th>
<th>Ranked keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>vladimir putin</td>
<td>dmitry peskov</td>
<td>dmitry peskov</td>
</tr>
<tr>
<td>kremlin spokesman</td>
<td>russia</td>
<td>russian</td>
</tr>
<tr>
<td>health</td>
<td>russia</td>
<td>russian</td>
</tr>
<tr>
<td>health</td>
<td>kremlin</td>
<td>kremlin</td>
</tr>
<tr>
<td>health</td>
<td>kremlin</td>
<td>kremlin</td>
</tr>
<tr>
<td>kremlin</td>
<td>kremlin</td>
<td>kremlin</td>
</tr>
<tr>
<td>health</td>
<td>kremlin</td>
<td>kremlin</td>
</tr>
<tr>
<td>kremlin</td>
<td>kremlin</td>
<td>kremlin</td>
</tr>
<tr>
<td>kremlin</td>
<td>kremlin</td>
<td>kremlin</td>
</tr>
</tbody>
</table>

3.3 Time-Aware Features

Given an article $a$ and corresponding article-shard $H_a$, we form article-hashtag pairs $(a, h), h \in H_a$, and for each
pair create a feature vector \( x_{a,h} \). Since the relevance of a hashtag to an article is time dependent, we extract time-aware features to describe the article-hashtag pair, and write the classification function as \( f(x_{a,h,t}) = \eta_t \).

We build on prior feature engineering work on Twitter and news data \([4, 24]\) to investigate useful features and adapt them to our local and global time-windows \( \lambda \) and \( \gamma \). One important aspect in feature engineering for learning-to-rank, is that features need to be comparable across queries, because we aim to learn a single ranking function for all queries using the same set of features. Additionally, all features have to be normalized at query-level for dealing with the issue of different candidate set sizes, and the variance between queries.

We identify five classes of features: Local, Global, Trending, Headline and User. Four of them reflect properties of the hashtag, while the fifth reflects social network characteristics of users. Considering the real-time and high precision requirement of our approach, we only use low-computation-cost features.

**Bag-of-words Representation** A tf-idf bag-of-words representation is formed from the text in each pseudo-article (headline, subheadline, first sentence) as a vector \( a \) whose components are:

\[
tf(t, a) \times idf(t, A)
\]

where \( tf(t, a) \) is the term frequency of the term \( t \) within the whole article defined as in \([21]\):

\[
tf(t, a) = 0.4 + \frac{(1 - 0.4) \cdot freq(t, a)}{\max\{freq(w, a) : w \in a\}}, \tag{1}
\]

The inverse-document-frequency is computed from the article collection \( A \), gathered in the time window \( \gamma \):

\[
idf(t, A) = \log \frac{|A|}{|\{a \in A : t \in a\}|} \tag{2}
\]

Similarly, given any tweet-bag, \( S \subseteq T \), we form a bag-of-words representation as a vector \( h(S) \), whose components are the term frequencies of all terms occurring in the tweets in \( S \): \( tf(t, S) \).

**Local similarity** \( LS_{a,h,\lambda} \): Compares the article text to a local hashtag description via the cosine similarity as shown in Equation 3. Let \( T_{a,h,\lambda} \) be the subset of tweets in \( T_a \) that mention \( h \) within time window \( \lambda \). \(||||\) denotes the L2 norm.

\[
LS_{a,h,\lambda} = \frac{a \cdot h(T_{a,h,\lambda})}{|a|||h(T_{a,h,\lambda})||} \tag{3}
\]

The local similarity is an important content feature that indicates how relevant a hashtag is to an article.

**Local hashtag frequency** \( LF_{a,h,\lambda} \): Captures local popularity of usage for a given hashtag in the article tweet-bag \( T_a \) within time window \( \lambda \).

\[
LF_{a,h,\lambda} = \frac{|T_{a,h,\lambda}| - \min\{|T_{a,v,\lambda}|\}}{\max\{|T_{a,v,\lambda}|\} - \min\{|T_{a,v,\lambda}|\}} \tag{4}
\]

\[
LF'_{a,h,\lambda} = \frac{\log|T_{a,h,\lambda}| - \min\{\log|T_{a,v,\lambda}|\}}{\max\{\log|T_{a,v,\lambda}|\} - \min\{\log|T_{a,v,\lambda}|\}} \tag{5}
\]

where \( v \in H_a \). We choose to include both the absolute size of the tweet-bag and the log of its size as separate features, which are normalised using min/max feature scaling, as shown in Equations 4 and 5. The local frequency feature compares all hashtags from the same set \( H_a \), and indicates whether a hashtag is dominating the topic.

**Global similarity** \( GS_{a,h,\gamma} \): Distinguishes between general and topic specific hashtags. It builds on similar equations as for local similarity, but now the article bag-of-words representation is compared with the whole hashtag tweet-bag \( T_h \) within global window \( \gamma \):

\[
GS_{a,h,\gamma} = \frac{a \cdot h(T_{h,\gamma})}{|a|||h(T_{h,\gamma})||} \tag{6}
\]

General hashtags like \#news, may seem relevant to an article when looking at only the article tweet-bag \( T_a \), but if we consider all tweets in \( T_h \), \#news is irrelevant since it is used with all news stories. A topic specific hashtag should maintain a high global similarity score to the article. However, global similarity is very expensive to obtain, as the system processes about 1 million tweets daily. To address this, in every computation round we take a random sample of \( T_{h,\gamma} \) of size 5k to estimate the term-frequency score, and reuse this for the global hashtag feature for all articles. The size of the sample is constrained by the low-latency requirement of the system.

**Global hashtag frequency** \( GF_{a,h,\gamma} \): Captures global popularity of usage for a given hashtag. Let \( |T_{h,\gamma}| \) denote the number of tweets in \( T_h \) within global time window \( \gamma \). \( GF_{a,h,\gamma} \) is computed as in Equations 4-5, after replacing \( |T_{a,h,\lambda}| \) by \( |T_{h,\gamma}| \). A globally popular hashtag usually indicates a breaking news, with more news articles published on that topic, which increases the probability of such hashtag being relevant to an article.

**Trending hashtag** \( TR_{a,h,t_a} \): Captures a significant increase in local hashtag frequency and aims to identify article-wise trending hashtags. In order to separate emerging topic specific hashtags (e.g., \#charliehebdo, \#jesuischarlie for a recent terrorist attack in France) from hashtags with a high general usage rate (e.g. \#news, \#breaking), being able to identify trending hashtags early on is very important. Therefore, it is not enough to only capture the current hashtag frequency, but we also need to check how quickly this is increasing.

Given time window \( W_n = t_n - t_{n-1} \), the number of tweets mentioning \( h \) in tweet stream \( T_a \) in time window \( W_n \) is \( |T_{a,h,W_n}| \), then:

\[
TR_{a,h,t_n} = \frac{|T_{a,h,W_n}| - |T_{a,h,W_{n-1}}|}{|T_{a,h,W_{n-1}}|} \tag{7}
\]

**Expected gain** \( EG_{a,\gamma,W_n} \): Captures the potential of \( h \) in the near future (a few minutes later), and is expected to boost trending hashtags while punishing fading ones.

Based on trending feature \( TR_{a,h,t_n} \), we also have the expected number of tweets in \( T_a \) mentioning \( h \) for the next time window \( W_{n+1} \), denoted by \( E(|T_{a,h,W_{n+1}}|) \):

\[
EG_{a,\gamma,W_n} = E(|T_{a,h,W_{n+1}}|) = (1 + TR_{a,h,t_n}) \cdot |T_{a,h,W_n}| \tag{8}
\]

We create two features, the absolute expected gain and the log of this value with min/max scaling as in Equations 4-5.

**Hashtag in headline** \( HE_{a,h} \): After observing user behaviour and trending hashtags over time, one can notice that many hashtags literally reflect their topic. Many hashtags are a variation of the name of the people/place/event being
discussed. It could be an acronym (e.g. #cwc2015 for cricket world cup 2015), or concatenated names (e.g. #sydneysiege for the Sydney hostage attack). Although this is not always the case (e.g. #carrythemhome for England Rugby, #icantbreathe for Eric Garner’s death), being able to use such information may help the classifier. We define $HE_{a,h}$ as a binary feature equal to 1 if the hashtag is in the pseudo-article (headline, sub-headline, first sentence) after removing space between terms.

**Unique user ratio** $UR_{a,h,\lambda}$: The ratio of unique Twitter users using $h$ in $T_a$ within time window $\lambda$, to the number of tweets. Function $User(T)$ returns the set of users in tweet stream $T$.

$$UR_{a,h,\lambda} = \frac{|User(T_{a,h,\lambda})|}{|T_{a,h,\lambda}|} \quad (9)$$

Noise filtering is extremely important for Twitter hashtag recommendation, because there are many spam users and twitter-bots posting spam tweets with self-created hashtags. The unique user ratio can help the classifier separate the genuinely popular hashtags from spammy hashtags, something that local/global frequency cannot achieve.

**User credibility** $UC_{a,h,\lambda}$: The quality of a hashtag depends on the users using it. A commonly used Twitter user credibility indicator is the number of followers. Users with more followers are usually celebrities, domain experts and experienced users who work hard to attract followers. Therefore, we define user credibility as the maximum, the average and the median of the followers of users tagging $h$ in article tweet bag $T_a$ in $\lambda$.

$$MaxF_{a,h,\lambda} = \max(\text{Follower}(u)), u \in User(T_{a,h,\lambda}) \quad (10)$$

$UC_{\text{max}}$ is the min/max scaled $MaxF_{a,h,\lambda}$; we also compute $UC_{\text{avg}}$ and $UC_{\text{median}}$ using a similar approach.

**Cost of features.** As shown by the previous equations, most of our features are based on the local article tweet-bag $T_a$, which is fairly small. Hence they are cheap to obtain. The experiment in Section 4.5 shows that the execution time for feature computation, which is the most time consuming step in our approach, grows linearly with the number of tweets for the entire article collection.

4. EVALUATION

In this section we discuss our methodology for gathering labeled data and show extensive experiments analysing our techniques in comparison to the state-of-the-art.

4.1 Gathering labeled data

As discussed in the previous sections, we model hashtag recommendation as a learning-to-rank problem via a relevance classification approach. Here we describe the process of gathering labeled data for the classifier. We define three classes of relevance for each article-hashtag pair:

- A hashtag is specifically on the topic of the news article. For example, for articles describing the recent German Wings plane crash, “#germanwings, #4u9525, #a320” fall in this category.
- A hashtag is generally on the topic of the news article. For example, for the same story, “#barcelona, #france” fall in this category.
- Irrelevant hashtags, including off topic and spammy hashtags. For example “#news, #bbc, #breaking” fall in this category.

In order to gather relevance labels, we have implemented our methods in a system that can be accessed via a Web interface\(^2\). We continuously track the RSS news feeds of 7 news organizations: Reuters, BBC, Irish Times, Irish Independent, Irish Examiner, RTE, and The Journal, publishing around 900-1,000 articles each day. Using the methods described above, we extract article-keyphrases and retrieve tweets using the Twitter Streaming API, updating the tweet-bag of each article every 5 mins over a 24h period. The users can see the hashtags retrieved via simple baselines for each article and can provide feedback for each hashtag. The baselines use simple frequency of usage for a hashtag or the local cosine similarity between the article and hashtag profiles for $h \in H_a$ over local time window $\lambda$. The interface as shown in Figure 1 enables users to quickly provide feedback while browsing the news presented.

One interesting aspect of labeling in this dynamic context is that for the same article-hashtag pair the label may change depending on the time of labeling, in particular more specific hashtags emerge as users engage with news stories on Twitter. For example, for the German Wings crash, #plane crash is initially very relevant, but as Twitter users focus more on the story details, #airbus320, #andreaslubitz become more specific and therefore more relevant. To simplify the labeling procedure, users were instructed to decide only if a hashtag is relevant (specifically or generally on the topic) or irrelevant to the article, at the time they are labeling it. We exposed this Web interface with the above instructions to a group of researchers and journalists over 1 month, allowing us to gather around 1,200 labeled examples\(^3\). We use this data as ground truth for evaluating various features and approaches. Details on the labeled data distribution are given in Table 3. Note that articles are only paired with subsets of hashtags, rather than all hashtags (e.g., the labeled pairs are a subset of the full cross-product of articles and hashtags).

\(^2\)The Insight4News system for gathering labeled data: insight4news.ucd.ie/insight4news/article_list

\(^3\)Labeled data available from https://sites.google.com/site/bichenshi/
### 4.2 Experiment 1: Keyphrase Extraction

In this section we evaluate three different approaches for extracting keyphrases with the goal of maximizing a form of precision and recall on the retrieved tweet set for the article. The methods compared are:

1. **Tf-idf unigrams**: Select all unigrams (single words) in the pseudo article (headline, sub-headline, first sentence). Compute tf-idf of single words using full article. Pair words to form 2-gram phrases\(^4\). Rank pairs by the average of tf-idf scores of individual terms, take the top 5 pairs as the article’s keyphrases.

2. **POS-tag**: Apply part-of-speech-tagging to the pseudo article. Take the first 5 nouns/phrases by giving priority to entities (noun-phrases, proper nouns), frequent nouns, all other nouns. Pair the single nouns to 2-gram phrases, break long noun phrases into 2-grams. Take the first 5 pairs as the article’s keyphrases (alphabetical order).

3. **POS-tag + Tf-idf** (our approach): Same process as POS-tag but for selecting final subset, rank pairs by average tf-idf score of individual terms and select the top 5 pairs as the article’s keyphrases.

**Experiment Setup.** We collect 300 news articles and extract keyphrases using the above three approaches. We track these article-keyphrases for 24h, via the Twitter Streaming API, and for each article we gather 3 tweet-bags corresponding to the three sets of article keyphrases. Table 4 shows example keyphrases for a given article, under different selection strategies.

For each of the 3 approaches, we have 300 articles, and each article has one tweet-bag. To estimate the precision of each approach, we compute the cosine similarity between the article and its tweet-bag tf-idf profile, then average over the 300 articles. This gives us an indicator of the focus of the tweet-bag. To estimate the recall, we average the sizes of the tweet-bags (number of tweets per article) over all articles.

**Evaluation.** As shown in Table 5, combining POS-tagging (for light entity detection) and tf-idf ranking (to focus on the right terms), retrieves twice as many tweets as compared to the other two approaches, and the tweets are also more similar to the article content. Note that this article-keyphrase extraction step is focused on retrieving relevant content from a very noisy and fast-paced social media stream such as Twitter, rather than being a generic article query formulation method. Although still noisy, this step is followed by a precision oriented ranking based on a learning approach.

### 4.3 Experiment 2: Feature Evaluation

**Experiment Setup.** We present a thorough analysis of the influence of different features on learning a hashtag relevance classifier. The evaluation is done via 10-fold cross-validation on the full labeled set (1.2k examples). Since this step is important for avoiding retrieval of noisy tweets.

<table>
<thead>
<tr>
<th>Features</th>
<th>Local Frequency</th>
<th>Global Frequency</th>
<th>POS-tag</th>
<th>POS-tag + Tf-idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic</td>
<td>0.1374</td>
<td>1.00</td>
<td>0.1321</td>
<td>0.1712</td>
</tr>
<tr>
<td>Norm</td>
<td>0.00</td>
<td>0.1321</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>(Norm1&amp;2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All((Norm1&amp;2))</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 2:** Boxplot distributions of 9 features in positive (blue) and negative (red) labelled data.

We test different combinations of the five types of features (Local, Global, Trending, Headline, and User, 14 features in total), and compare the classification performance. Figure 2 shows the distribution in the labelled data of 9 main features. We use three standard machine learning metrics to evaluate classification quality: Precision, Recall and AUC [32]. The first two measure classification quality after a threshold is imposed on the classification score. We show Precision and Recall on the positive class (coined PP and PR) as well as the weighted average Precision and weighted average Recall over both classes (coined WP and WR). The latter averages the Precision/Recall for the 2 classes weighting performance on each class by class size [32]. AUC measures ranking quality and is not dependent on a classification threshold. In practice we are more concerned with Precision and AUC quality, since for our application domain it is more important to have high Precision (recommend a few specific hashtags in top ranks), then high Recall (retrieve all relevant hashtags).

**Evaluation.** Table 6 shows the results of using different combinations of features. Basic refers to using Local and Global article-hashtag content and popularity features. Norm1 and Norm2 refer to min/max scaling of the original versus the log of feature values (as described in Equation 4 and 5). All((Norm1&2)) is the approach that includes all 14 features used in this work. Most related work in this area has focused on the type of features included in approach Basic. We observe that the three other categories of features (Trending, Headline and User), and the two normalization approaches, increase the Precision/Recall by 5% and the AUC by 9%.
4.4 Experiment3: Size of Training Data and Time Effect

In this experiment we analyse the influence of the number of training examples, as well as the time effect on the classification quality. We carry out two experiments. The first, studies the effect of recency and size of training data, on the quality of recommendation (variable training set, fixed test). The second, checks whether the quality of recommendation remains stable over time (5 months) given that we do not re-train the classifier (fixed training set, variable test).

4.4.1 Training Size versus Recency

Experiment Setup. We order the 1.2k labeled examples by time from the oldest to the most recent. We use the most recent 400 examples as hold-out test set, and gradually add in examples to the training set by batches of size 50, and train a Random Forest classifier. We compare two strategies for selecting training data: backward and random. The Backward approach selects examples starting with the most recent 50 examples and adds 50 by going back in time to older examples, until it reaches 800 training examples. The Random strategy selects a random sample of given size, from the set of 800 training examples.

Evaluation. Figure 3 shows the Precision of the classifier tested on the hold-out test set when increasing the number of training examples, with the two sampling strategies. The plots for Recall and AUC behave similarly and are not shown here. We note that both strategies behave similarly, with Precision increasing quickly with the number of labeled examples. The Random strategy delivers less stable Precision at smaller sample sizes. This is due to the variation in the positive/negative ratio of examples in those labeled training sets. Nevertheless, both methods achieve similar Precision at about 700 labeled examples, suggesting that the sampling strategy is not important once enough training data is available.

4.5 Experiment4: Comparison to State-of-Art

In this experiment, we compare our approach (coined Hashtagger) to three state-of-art hashtag recommendation techniques: Naive Bayes, Liblinear, and LDA. We study the precision as well as scalability of these approaches. The main difference between our method versus existing methods is in how modelling is done and what features are used. Regarding modelling, the state-of-the-art approaches focus on modelling each hashtag as a topic, as in classic topic
classification, while we model the hashtag relevance with a learning-to-rank approach. Regarding features, most methods rely on text similarity (between article and hashtag representation) and the frequency of usage of the hashtag. We compare our approach to prior techniques, on the same input data and set of features (local text similarity and frequency) to assess the impact of the modelling approach. Additionally, we also show results for our method using the full set of features proposed, to assess the impact of modelling plus features.

1. Naive Bayes\(^5\) [7]: A hashtag is seen as a category and tweets mentioning that hashtag are used as labeled data to train a Naive Bayes classifier via multi-class classification.

2. LibShortText [36]: A library for short-text classification and analysis that builds upon the state-of-the-art library LibLinear [8], which support millions of instances and features. LibShortText implements multi-class Gaussian-kernel SVM. Similar to the Naive Bayes approach, each hashtag is considered as a category and tweets mentioning a hashtag are used as labeled data.

3. LDA\(^6\) [6]: Topic modelling with Latent Dirichlet Allocation representing each tweet as a mixture of topics. Trained on a collection of tweets, LDA returns a set of scored topics that each tweet belongs to, each topic is typically represented as a group of ranked words. We use the highest scored topic to recommend hashtags.

### 4.5.1 Precision

As an evaluation metric we use Precision@1 by recommending the maximum score prediction of each method.

**Experiment Setup.** The three state-of-the-art approaches are designed to work best in a static environment, where the set of tweets and hashtags are static and are analysed in an offline batch mode. To adapt them to a real-time environment, we retrain these methods in a sliding window style: offline batch mode. To adapt them to a real-time environment, we retrain these methods in a sliding window style: offline batch mode.

At the next time point \(t_1 = t_0 + 2h\), we discard the previous models, retrain all models with new tweets that are \(4h\) ahead of \(t_1\), and use these models for another \(2h\).

We randomly pick a starting time point \(t_0\) (0:00, April 14th, 2015, UTC), then run the experiment for 24h, involving 270 articles and 313k tweets that have at least one hashtag (about 26.1k tweets per 4h time window). Then each pseudo article (headline, sub-headline and first sentence) is considered as a rich tweet, and each method recommends one hashtag to each article. For LDA, the number of topics is set to 50 per time window and the number of iterations is 100, and we use the top ranked term in the top ranked topic as a recommended hashtag [20, 6, 16]. In order for all methods to work from the same data, we test Hashtagger on feature vectors computed over the tweets in tweet-bag \(T_a\) that are published up to 4h ahead of the article publishing time \(t_a\). Also, we test two versions of the Hashtagger: Hashtagger(2) uses only two local features \((LF_{a,h,\lambda} \text{ and } LS_{a,h,\lambda})\), while Hashtagger(All) uses all 14 features. As training data, both of them use the full 1.2k labeled set.

**Evaluation.** We asked a group of annotators to evaluate the \(5 * 270 = 1350\) article-hashtag pairs as relevant/irrelevant and average their results. As each method gives one recommendation per article, accompanied by a prediction score, the Precision@1 and the number of articles that get a recommendation (article coverage rate) are both functions of the threshold on the prediction score. A higher threshold value results in a better recommendation quality, but will naturally reduce the article coverage rate. Since the predicted scores of different methods are not directly comparable, we compare the Precision@1 for the five methods under different article coverage rates. For each method, we change the threshold to each unique predicted value in increasing order, and record the article coverage rate and the Precision@1 at that threshold, as shown in Figure 5.

When the article coverage rate is 100% (e.g. we record 1 recommendation for each article regardless how low the prediction score), the Precision@1 for Hashtagger(All), Hashtagger(2), Naive Bayes, LibShortText, and LDA is 0.618, 0.533, 0.374, 0.447 and 0.385. The results for the SOA methods are in agreement with published studies [20, 15, 12, 6, 16]. Regardless of the article coverage rate, Hashtagger(2), which uses only basic similarity and frequency features, constantly out-performs the other three methods, showing the positive impact of our modelling. Hashtagger(All) has the highest Precision@1 score, suggesting that both modelling and feature engineering are important. For a fixed threshold of 0.5, Hashtagger(All) has Precision@1 of 0.89.

Table 7 shows recommended hashtags and prediction scores of the five approaches. Hashtags in bold are labelled as relevant by all our annotators. We note that Hashtagger gives more reliable recommendations, including recommending specific hashtags (e.g. #wiveng for West India vs England), while the other three approaches provide more general, even irrelevant hashtags.

### 4.5.2 Scalability

**Experiment Setup.** To further examine the scalability of the four approaches, we compare their execution time by increasing the number of tweets for training/testing. For Naive Bayes, LibShortText and LDA, we take different size samples of tweets from 10k to 150k, as training data, and record their model fitting time. We repeatedly run Hashtagger over randomly selected article collections with total tweet-bags size ranging from 10k to 150k.

**Evaluation.** The execution time shown in Figure 6 for...
5. APPLICATIONS

In this section we study two applications of real-time hashtag recommendation for news. The first uses Twitter as a publishing platform of online news and measures the effect of attaching hashtags to headlines as a way of reaching a publishing platform of online news and measures the effect of attaching hashtags to headlines as a way of reaching a publishing platform of online news.

### 5.1 Online News Publishing

We study the impact of our hashtag recommendations by automatically tweeting news headlines as follows. As soon as a headline is retrieved from an RSS feed and it receives a hashtag recommendation from our system, it falls into one of three groups, decided by a random variable. The first group is tweeted as is (headline + URL), the second it is tweeted by appending #news to each headline (headline + #news + URL) and the remaining group is tweeted with the top hashtag recommended by Hashtagger. We then use impact metrics provided by Twitter Analytics \(^7\) to compare the 3 groups of headlines. The goal is to assess whether tweeting the news headlines with our recommended hashtags leads to higher engagement with those news, as compared to not using any hashtags, or using a generic hashtag such as #news. The hypothesis is that by attaching good hashtags to the news headlines, those news reach wider and possibly more engaged audiences.

We automatically tweet from a Twitter account named @insight4news which we use for researching the effect of publishing hashtagged news on Twitter. This account was created in April 2015 and at the time of writing has issued 99k tweets and has 438 followers. We run the process described above over 3 months, and draw a sample of 15k tweeted news headlines, split into the 3 groups (5k per group). We collect the total impressions, engagement and url clicks as provided by Twitter Analytics. The original data is available here \(^8\). Figure 7 shows these metrics for the 3 groups. In order to avoid spurious results, we remove the outliers for each group and metric (the top 5% quantile).

We observe that tweets with no hashtag and with #news attract similar total amount of impressions (85k), engagements (400/600) and url clicks (300), with the #news group only slightly better than the no-hashtag group, showing that a generic hashtag does not draw more audience to tweets. Tweets with our recommended hashtags generate more traffic, with 150k impressions, 1.3k engagements and 750 url clicks. In addition, the engagement rate (the number of engagements over impressions) is also increased compared to the no-hashtag group: 0.86% versus 0.47%, suggesting that our approach helps tweets reach a wider audience, and leads to increased user engagement with the tweet content (news articles).

### 5.2 Social Indexing of News

The classic approach to indexing documents is to use keywords extracted from those documents. For example, for a news headline "Greek crisis: Euro zone rules out talks until after referendum", the corresponding article would hypothetically be indexed by the keywords "greek, crisis, euro, referendum". When issuing a query such as "greece crisis euro" articles indexed by these keywords are retrieved from the article collection. Although the accuracy of keyword search

---

8 https://goo.gl/A6oMQi

---

Table 7: Examples of recommended hashtags by the four compared methods.

<table>
<thead>
<tr>
<th>Article Headline</th>
<th>Hashtagger(All)</th>
<th>Score</th>
<th>Hashtagger(2)</th>
<th>Score</th>
<th>Naive Bayes</th>
<th>Score</th>
<th>LibShortText</th>
<th>Score</th>
<th>LDA</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nokia in deal Talks with Alcatel-Lucent</td>
<td>#nokia</td>
<td>0.43</td>
<td>#nokia</td>
<td>0.59</td>
<td>#news</td>
<td>0.06</td>
<td>#news</td>
<td>0.06</td>
<td>#news</td>
<td>0.06</td>
</tr>
<tr>
<td>Ian Bell-ton gives England the upper hand in Antigua</td>
<td>#westindies</td>
<td>0.52</td>
<td>#westindies</td>
<td>0.65</td>
<td>#icc</td>
<td>0.72</td>
<td>#icc</td>
<td>0.72</td>
<td>#icc</td>
<td>0.72</td>
</tr>
<tr>
<td>Syria-bound son of British councillor deported from Turkey</td>
<td>#syria</td>
<td>0.97</td>
<td>#syria</td>
<td>0.88</td>
<td>#yemen</td>
<td>0.68</td>
<td>#yemen</td>
<td>0.68</td>
<td>#yemen</td>
<td>0.68</td>
</tr>
<tr>
<td>Seventeen killed in attack on Somali education ministry</td>
<td>#somalia</td>
<td>0.97</td>
<td>#somalia</td>
<td>0.91</td>
<td>#somalia</td>
<td>0.91</td>
<td>#somalia</td>
<td>0.91</td>
<td>#somalia</td>
<td>0.91</td>
</tr>
</tbody>
</table>
has continuously improved, the main weaknesses remain: (1) Missing articles that don’t have these exact keywords; (2) Returning too many irrelevant results.

An accurate method for associating articles and Twitter hashtags, allows us to index articles using keywords and hashtags. For example, the above article could be indexed by "greek, crisis, euro, referendum, #greece, #grexit, #gref-erendum, #tsipras, #eurogroup". This also means that now we can formulate queries that mix keywords and hashtags, such as greece #grexit. We coin this social indexing, the benefits of which are three-fold:

1. Takes advantage of crowdsourced content as a form of real-time, continuous tagging of news.
2. Hashtags are not necessarily topical, and they have the advantage of grouping together articles belonging to the same story (e.g., racial conflicts in US, #eric-garner, #blacklivesmatter, #icantbreathe).
3. Hashtags allow the query to focus on diverse aspects of a story (e.g., Greek economic crisis, #grexit, #gref-erendum, #tsipras, #merkel, #ecb, #inf, #finland).

We discuss the above points in the context of story tracking. Many news organisations offer story-pages on their website, i.e., curated collections of news articles that allow the reader to get an overview and updates on particular problems, e.g., referendums, elections, budgets. The Irish Times has dedicated story-pages for issues of relevance to the Irish society, e.g., the introduction of a tax on water\(^9\) (Twitter hashtag #irishwater), the inquiry into the banking collapse\(^10\) of 2008 (Twitter hashtag #bankinginquiry), the recent marriage equality referendum\(^11\) (Twitter hashtag #marref). Similarly, the BBC and The Guardian also publish story-pages, e.g., the BBC story-page on the "Greek debt crisis"\(^12\) and Guardian page on Liberia\(^13\). Preparing these story-pages currently relies on prior agreement among the journalists, to manually tag all articles relevant to a pre-agreed set of stories, with the same set of tags. Once a decision is taken to create a story-page, those articles are continuously retrieved from the news archive via the manual tag set. The problem with this approach is that it relies on foresight over which stories are worth covering and what is the right tag to use for those story-articles. By building on our hashtag recommendation approach, we let the Twitter crowd do the tagging in real-time (via Hashtagger), potentially capturing novel emerging concepts. The assumption is that most stories that are worth story-pages have a lot of quality discussions and focused hashtags on Twitter, hypothesis currently supported by our experiments. For example #migrant covers the unfolding migrant/refugee crisis, retrieving 90 articles\(^14\) with this recommended hashtag in the time period August 27 to October 15, 2015. The #refugee tag retrieves 149 articles over the same time period, indicating a potential change of discourse around this issue. We intend to further study the use of social indexing for story tracking and retrieval.

6. CONCLUSION

We present Hashtagger, an approach for real-time high-precision hashtag recommendation for streaming news. Our method relies on a learning-to-rank model tailored to a dynamic setting where news and tags are streaming and have variable life-cycles. We systematically study our approach in comparison to the state-of-the-art and show that our method delivers much higher Precision compared to existing methods. This is due to our choice of modelling approach (relevance ranking versus topic modelling) and the set of time-aware features we investigate. Hashtagger is designed to work in real-time real-world application settings. We employ our recommendations in a real-life study using Twitter as an online news publishing platform, and show that accurate hashtagging drives higher news engagement. We also discuss the implications of building on hashtag recommendation for social indexing of news. For the future we intend to analyse the impact of hashtag recommendation on automatic story detection and tracking.

7. ACKNOWLEDGMENTS

This work was funded by Science Foundation Ireland (SFI) under grant number 12/RC/2289.

8. REFERENCES

[2] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise

Figure 7: Twitter Analytics metrics to measure impact of hashtagging on news engagement.


