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

Bichen Shi
bichen.shi@insight-centre.org

Georgiana Ifrim
georgiana.ifrim@insight-centre.org

Neil Hurley
neil.hurley@insight-centre.org

Insight Centre for Data Analytics
University College Dublin
Ireland

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., promoting original content to Twitter communities and social indexing of news to enable better retrieval and story tracking. In contrast to the state-of-the-art that focuses on topic modelling approaches, we propose a *learning-to-rank* approach for modelling *hashtag relevance*. This enables us to deal with the dynamic nature of the problem, since a relevance model is stable over time, while a topic model needs to be continuously retrained. 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 that is currently under user trial with a big news organisation.

Keywords

learning-to-rank; dynamic topics; social indexing; news; hashtag recommendation

1. INTRODUCTION

Social media platforms such as Twitter have taken a central role in the consumption, production and dissemination of news [35]. Twitter 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 (e.g., #parisattacks). News organisations use hashtags to target Twitter communities in order to promote original content and engage readers. Journalists sometimes introduce new hashtags, but the Twitter crowd is the one that most often creates and selects a few of the possibly many competing hashtags, thus echoing the current social discourse (e.g., #migrant, #refugee, #refugeeswelcome).

However, an automatic approach to *real-time, high precision* hashtag recommendation for news is currently missing, and both journalists and news readers have to invest significant effort to manually search for relevant hashtags. Most existing approaches use topic modelling [20, 15], by considering hashtags as topics, and mapping news articles to topics using content similarity, regardless of whether users actively engage with those hashtags. As the relevant hashtags change quickly (some die-off and new ones emerge), and the news and Twitter environments are highly dynamic, such approaches need to continuously retrain to adapt to new content. In addition, since most existing approaches are trained on static collections of noisy tweets, they do not achieve high enough precision for practical use (e.g., precision of 38% [6]), and cannot deliver recommendations in real-time.

In this paper, we model the problem of recommending *highly specific, actively used* hashtags, to a stream of news articles, as an Information Retrieval (IR) learning-to-rank (L2R) problem. 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. For incoming articles, we split the recommendation process into two steps: (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 pre-trained L2R approach to score the relevance of the candidate article-hashtag pairs. From our analysis, the model of what makes a hashtag relevant to an article doesn't change over time. This means that we can use a small amount of human-annotated data to train the L2R model once, and at test time compute features to describe and score the relevance of new article-hashtag pairs. Thus, we can recommend new hashtags that were not previously seen in the training set, since the relevance model is not hashtag specific, but encodes general characteristics (learned from the human-labeled training data) of relevant versus irrelevant pairs. Classic IR approaches typically assume a static document collection and

WWW 2016, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4143-1/16/04.
http://dx.doi.org/10.1145/2872427.2882982 .

dynamic user-queries, while in our case both documents and tags come as streams, and the set of relevant tags for a given document changes over time. For example, tweets discussing the news story *"Plane crashes in southern France"* are first tagged with *"#planecrash #france"*. Over time, the hashtags for this story become more specific, such as *"#germanwings #4u9525 #a320"* driven by the usage of Twitter users. To address the dynamic relevance of article-hashtag pairs, we investigate a set of low-cost, time-aware features.

We work with real-world data collected from existing RSS news feeds which we connect to relevant Twitter streams, and build a live demo system to test our framework[1]. To give a feel for the data scale, over a time period of 12 months, for about 1,000 news articles processed each day, we averaged more than 1 millions 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, #icantbreath, #worldcancerday, with longer stories running over weeks or months having a more stable set of hashtags, e.g., #ebola, #ebolavaccine, #grexit.

Contributions. We summarise our main contributions as follows:

1. We formulate real-time hashtag recommendation for streaming news as a L2R problem, and show that the L2R framework is more effective than topic modelling for our dynamic problem setting.
2. We investigate time-aware features for high-precision, low-latency 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 (a form of real-time crowdsourced tagging of news).

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 Naïve 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]. In addition, these models need to be constantly retrained to adapt to the new emerging hashtags, which makes the process more time consuming.

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 (L2R) 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 L2R framework for dynamic settings. We compare our method to existing topic modelling approaches: Naïve Bayes [7], Support Vector Machines [36] and Latent Dirichlet Allocation [6] (see Section 4.5).

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

In this section we discuss the proposed L2R 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 a global ranking model $f(q, d)$, and returns the top ranked documents. The $f(q, d)$ model is constructed automatically using supervised machine learning techniques with labelled ranking data [13].

In our hashtag recommendation setting, the query q is extracted from an individual article $a \in A$, where A is a stream

[1]Insight4news: `http://insight4news.ucd.ie`

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.

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 L2R 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 $\mathbf{x}$, with label $y \in \{0, 1\}$ ($y = 1$ if the hashtag is relevant for the article). Given m training examples $M = \{\mathbf{x}_i, y_i\}$, $i = 1, 2, ..., m$, we construct a global classifier $f(\mathbf{x}) = y$ to predict y for any feature vector $\mathbf{x}$ 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 $\mathbf{x}_t$ as shown in Section 3.3. We write $f(\mathbf{x}_t) = y_t$ 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 $\lambda = [t_a - 4h, t_n]$, with t_a the publishing time of article a, is an article-dependent time window, where $t_a \leq t_n$. 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 [3]. 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 γ, and retrieve a corresponding stream of tweets T.
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 $\mathbf{x}_t$ of the article-hashtag pair (a, h). Feed $\mathbf{x}_t$ to the relevance classifier, get hashtag recommendation $f(\mathbf{x}_t)$.

Table 2: Example process for extracting article-keyphrases.

Original keywords	Paired keywords	Ranked keyphrases
dmitry peskov	dmitry peskov	**putin vladimir**
vladimir putin	putin vladimir	**dmitry peskov**
russian	russian spokesman	**kremlin russian**
spokesman	kremlin russian	**health kremlin**
kremlin	health russian	**russian spokesman**
health	kremlin spokesman	health russian
	health kremlin	kremlin spokesman
	health spokesman	health spokesman

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. This step is important for avoiding retrieval of noisy tweets. 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 γ. 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.

The tf-idf ranking extracts keyphrases that reflect the main article focus, e.g., if several city names are extracted: New York, London, Paris, but the article main focus is on "London business", then "London business" will be ranked before "Paris business". Limiting to top-5 pairs achieves a good trade-off between scalability and quality of the retrieved tweet set.

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 $\mathbf{x}_{a,h}$. 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 $f(\mathbf{x}_{a,h,t}) = y_t$.

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 λ and γ. One important aspect in feature engineering for L2R, 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-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 $\mathbf{a}$:

$$\mathbf{a} = \mathrm{tf}(w, a) \times \mathrm{idf}(w, A)$$

where $\mathrm{tf}(w, a)$ is the term frequency of the term w within the whole article defined as in [21]:

$$\mathrm{tf}(w, a) = 0.4 + \frac{(1 - 0.4) * \mathrm{freq}(w, a)}{\max\{\mathrm{freq}(w', a) : w' \in a\}}\,. \tag{1}$$

The inverse-document-frequency is computed from the article collection A, gathered in the time window γ:

$$\mathrm{idf}(w, A) = \log \frac{|A|}{\{a \in A : w \in a\}} \tag{2}$$

Similarly, given any tweet-bag, $T' \subseteq T$, we form a bag-of-words representation as a vector $\mathbf{h}(T')$, whose components are the term frequencies of all terms occurring in the tweets in T': $tf(w, T')$.

Local similarity $LS_{a,h,\lambda}$**:** Compares the article text to a local hashtag tweet bag 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 λ. $||.||$ denotes the L2 norm.

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

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 λ.

$$LF_{a,h,\lambda} = \frac{|T_{a,h,\lambda}| - \min\{|T_{a,h',\lambda}|\}}{\max\{|T_{a,h',\lambda}|\} - \min\{|T_{a,h',\lambda}|\}} \tag{4}$$

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

where $h' \in H_a$. 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 γ:

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

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.

Global hashtag frequency $GF_{h,\gamma}$**:** 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 γ. $GF_{h,\gamma}$ 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_n}**:** 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,h,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 $E(|T_{a,h,W_{n+1}}|)$:

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

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. They 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 λ, to the number of tweets. Function $\mathrm{User}(T)$ returns the set of users in tweet stream T.

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

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 that 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 λ.

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

UC_{max} is the min/max scaled $MaxF_{a,h,\lambda}$; we also compute UC_{avg} and UC_{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 (SOTA).

4.1 Gathering labeled data

As discussed in the previous sections, we model hashtag recommendation as a L2R 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 spamy hashtags. For example "#news, #bbc, #breaking" fall in this category.

One way to gather cheap labels is to use the tweets that contain both article URLs and hashtags, and consider those hashtags as relevant labels. However, in our initial experiments, we found such data too little and too noisy. Most tweets with hashtags, although relevant to the article, do not contain the article URL. Additionally, a quarter of the tweets with article URLs are tagged with #news #breaking, which are too general, while many others are tagged with spam hashtags or a mixture of relevant and irrelevant hashtags. It is therefore difficult to directly use this data for training, so instead, we decided to collect high quality labels by involving manual annotators in a real-time labelling exercise.

In order to gather relevance labels quickly, 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.

[2]The Insight4News system for gathering labeled data: `insight4news.ucd.ie/`

Table 3: Details on the labeled data pairs.

Total	Positive	Negative	Collection Period
1238	348(28.1%)	890(71.9%)	04/12/2014-08/01/2015

Articles Involved	Hashtags Involved
217	725

Figure 1: The system interface used to gather feedback on hashtag relevance with respect to an article.

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 λ. 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. 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).

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. 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]Labeled data available from `https://sites.google.com/site/bichenshi/`

Table 4: Example article and extracted keyphrases using three approaches.

Headline	Putin re-emerges in public after rumours over 10-day absence
Subheadline	Russian president jokes to media that life 'would be boring if there was no gossip'
First Sentence	Vladimir Putin has reappeared in public after a mysterious 10-day absence that sparked frenzied speculation about the whereabouts of the Russian president, his health and mental wellbeing, and even his grip on power.
Tf-idf	president whereabouts, mysterious whereabouts, wellbeing whereabouts, frenzied whereabouts, absence whereabouts
POS-tag	gossip president, life power, media president, absence president, health media
POS-tag + Tf-idf	president russian, absence russian, putin vladimir, power russian, grip russian

Table 5: Similarity versus number of tweets retrieved via keyphrase extraction using 3 methods.

	Tf-idf	POS-tag	POS-tag + Tf-idf
Avg Cosine	0.1374	0.1321	**0.1712**
Avg No. Tweets	753.60	1092.13	**1870.58**

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 the time effect is encoded in the feature vector, randomising samples in the cross-validation step is not an issue. We show further evidence for this statement in Section 4.4. For the *relevance classifier* we use an ensemble approach: *Random Forest*. Our choice is based on previous studies that showed Random Forests are robust to noise and very competitive regarding accuracy [9].

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.

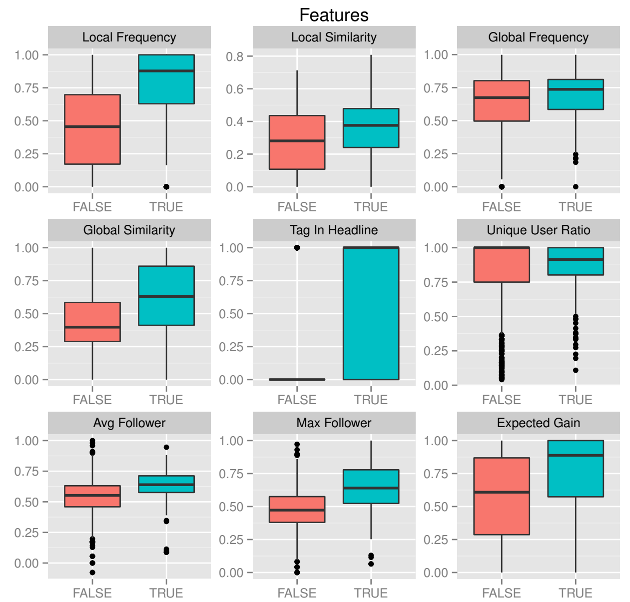

Figure 2: Boxplot distributions of 9 features in relevant (blue/TRUE) and irrelevant (red/FALSE) labelled data.

Figure 2 shows the distribution in the labelled data of 9 most important features. We use three standard machine learning metrics to evaluate classification quality: Precision, Recall and AUC [32]. The first two measure the 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), than 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 fea-

Table 6: **Evaluating features of the relevance classifier for hashtag recommendation.**

	PP	PR	WP	WR	AUC
Basic(Norm1)	81.5%	63.2%	85.3%	85.6%	86.7%
Basic(Norm2)	82.5%	63.8%	85.7%	86.0%	85.8%
Basic(Norm1&2)	83.8%	63.8%	86.1%	86.3%	87.0%
Basic+Trending	83.8%	66.7%	86.8%	87.0%	90.3%
Basic+Headline	82.2%	73.0%	87.7%	88.0%	92.5%
Basic+User	84.3%	64.7%	86.5%	86.7%	89.8%
All, no User	84.0%	74.1%	88.6%	88.8%	94.1%
All, no Headline	85.1%	64.1%	86.6%	86.8%	91.3%
All, no Trending	84.7%	75.0%	89.0%	89.2%	94.3%
All(Norm1)	87.2%	76.1%	90.0%	90.1%	94.9%
All(Norm2)	88.5%	75.0%	90.1%	90.2%	94.9%
All(Norm1&2)	**87.5%**	**76.4%**	**90.2%**	**90.3%**	**95.0%**

tures (Trending, Headline and User), and the two normalization approaches, increase the Precision/Recall by 5% and the AUC by 9%.

4.4 Experiment 3: 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 retrain 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.

4.4.2 *Recommendation Quality over Time*

Experiment Setup. We use the entire 1.2k labeled examples, which are collected in December 2014, to train a Random Forest classifier. For each month from March to July 2015, we randomly pick one day and use articles from that day as testing data. We ask a group of researchers to

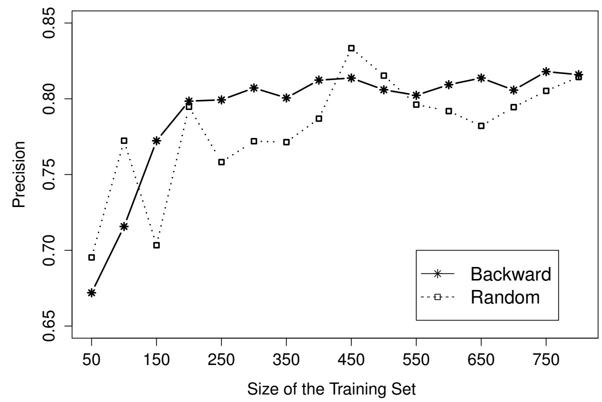

Figure 3: Precision for different size training data with two sampling strategies, tested on hold out set.

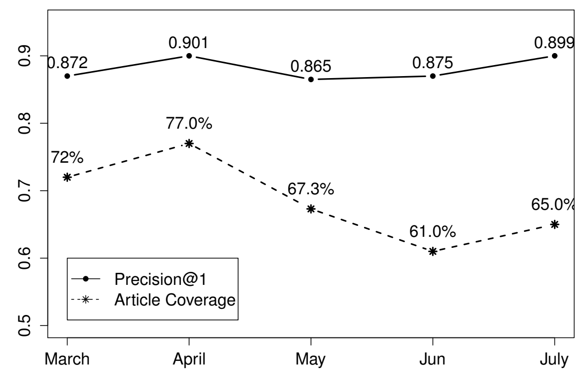

Figure 4: Precision@1 and article coverage for the 5 test days from March to July 2015 (relevance score threshold at 0.5).

evaluate the top one recommendation (ranked by the classification score), for each article in the 5 test days. The threshold of classification score is set to 0.5, which means an article gets a hashtag recommendation only if at least one hashtag has predicted relevance score above the 0.5 threshold.

Evaluation. We measure the average Precision@1 for each test day, based on the evaluation results of the annotators. In practice, we are also interested in the percentage of articles that get a recommendation (article coverage), which varies with the selected threshold, and is also influenced by the Twitter activity and topics of the news article on that day. Figure 4 shows the Precision@1 for all 5 days is around 0.87 and the percentage of articles covered is in the range of $60\% - 80\%$. The result suggests that even though the classifier is trained on December's data, the quality of recommendation remains stable when tested on data of half a year later, thus collecting new labeled data to retrain the classifier is not necessary.

4.5 Experiment 4: Comparison to SOTA

In this experiment, we compare our approach (named **Hashtagger**) to three SOTA hashtag recommendation techniques using topic modelling: Naïve 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 training data and features are used. Regarding modelling, the SOTA approaches focus on modelling each hashtag as a topic, as in classic topic classification, while we model the hashtag relevance with a L2R approach. For the training data, the SOTA approaches are trained on most recent tweets with hashtags, and need to be retrained constantly to adapt to the new content. As our ranking classifier only needs to be trained once, it uses the 1.2k label data for the entire run. 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, using the same tweets 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 asses the impact of modelling plus features.

1. Naïve Bayes[4] [7]: A hashtag is seen as a category and tweets mentioning that hashtag are used as labeled data to train a Naïve Bayes classifier via multi-class classification.
2. LibShortText [36]: A library for short-text classification and analysis that builds upon the SOTA library LibLinear [8], which support millions of instances and features. LibShortText implements multi-class Gaussian-kernel SVM. Similar to the Naïve Bayes approach, each hashtag is considered as a category and tweets mentioning a hashtag are used as labeled data.
3. LDA[5] [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 SOTA 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: Given a time t_0, all methods are trained on all tweets (with hashtags) falling in a 4h window ahead of t_0, then they recommend hashtags for articles that are posted up to 2h after t_0. 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. Because the Naïve Bayes' running time is very short (as shown in Section 4.5.2), we test Naïve Bayes under two settings: retrain every 2h and every 10min.

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($LF_{a,h,\lambda}$ and $LS_{a,h,\lambda}$), while Hashtagger(All) uses all 14 features.

Evaluation. We asked a group of annotators to evaluate the $6 * 270 = 1620$ 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), Naïve Bayes(2h), Naïve Bayes(10min), LibShortText, and LDA is 0.618, 0.533, 0.374, 0.396, 0.447 and 0.385. The results for the SOTA methods are in agreement with published studies [20, 15, 12, 6, 16]. Naïve Bayes(10min) has higher Precision@1 than Naïve Bayes(2h) showing that frequent retraining could reduce the content gap between training and test sets. 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. One advantage of Hashtagger is that, unlike other SOTA approaches, it can predict the relevance of unseen articles and hashtags that have not appeared in the training set. Due to our choice of modelling, it is also possible for us to gather a small amount of high quality manual labels for the classifier, which is much cleaner than the tweets with hashtags used by SOTA approaches. Gathering manual labels for other SOTA approaches is not feasible because they need to be retrained with new labels very often. Hence, Hashtagger achieves higher Precision@1 than SOTA approaches.

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 Naïve 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 Hash-

[4]`http://scikit-learn.org/...MultinomialNB`
[5]`https://pypi.python.org/pypi/lda`

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

Article Headline	Hashtagger(All)	Score	Hashtagger(2)	Score	Naïve Bayes	Score	LibShortText	Score	LDA	Score
Nokia in deal talks with Alcatel-Lucent	**#nokia**	0.83	#news	0.93	#news	0.90	#follow	0.09	#home	0.1
Ian Bell ton gives England the upper hand in Antigua	**#wiveng**	0.52	**#wiveng**	0.65	#lfc	0.72	#iran	0.13	#news	0.11
Syria-bound son of British councillor deported from Turkey	**#syria**	0.97	**#syria**	0.88	**#yemen**	0.68	#news	0.77	#wallstreet	0.04
Seventeen killed in attack on Somalia education ministry	**#somalia**	0.99	**#somalia**	0.94	**#somalia**	0.97	**#somalia**	0.23	**#somalia**	0.1

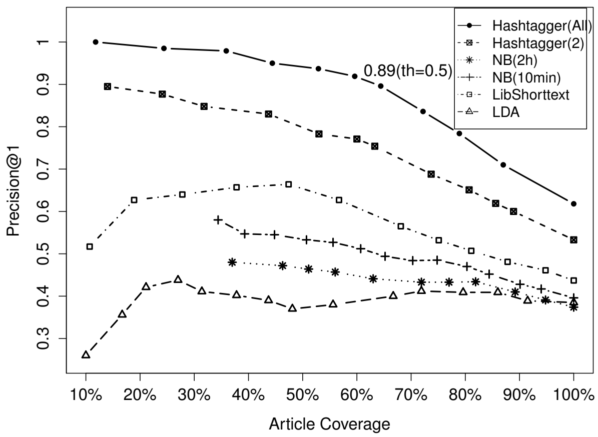

Figure 5: Precision@1 and article coverage of the six methods compared.

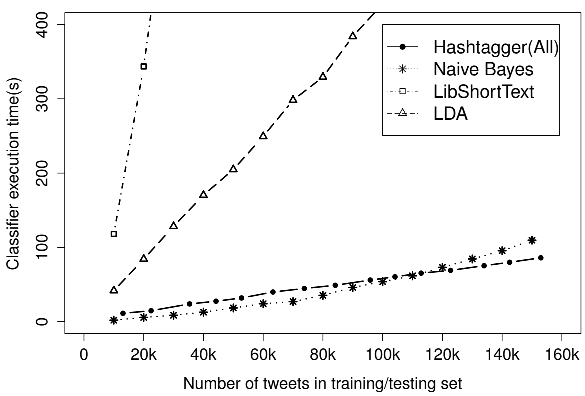

Figure 6: Running time of the four methods using different size of tweet set for training/testing.

tagger over randomly selected article collections with total tweet-bags size ranging from 10k to 150k.

Evaluation. The execution time shown in Figure 6 for each classifier matches known results: Naïve Bayes, known to be very efficient with linear training/testing complexity [21], takes around 100s to train on 150k tweets. The RBF SVM (LibShortText), with complexity $O(n^3)$ [5], takes 120s to train on only 10k tweets. The training speed of LDA is between Naïve Bayes and SVM taking 100s to train on 30k tweets (with 50 topics). Hashtagger has similar linear time complexity for testing as Naïve Bayes for training, processing 150k tweets in 100s. Nevertheless, Hashtagger delivers much higher recommendation precision.

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 wider Twitter communities, which in turn is hypothesised to lead to more engagement with those news (e.g., more URL clicks). The second application looks at the benefits of indexing news using recommended crowdsourced tags (which we call *social indexing*), for better news retrieval and story tracking.

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 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[6] 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 @insight4news3 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[7]. 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 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 hypothet-

[6]https://gnip.com/docs/Simply-Measured-Complete-Guide-to-Twitter-Analytics.pdf

[7]`https://drive.google.com/file/d/0B3N3pPOTCaegdFRtbzBGbkVXMnc`

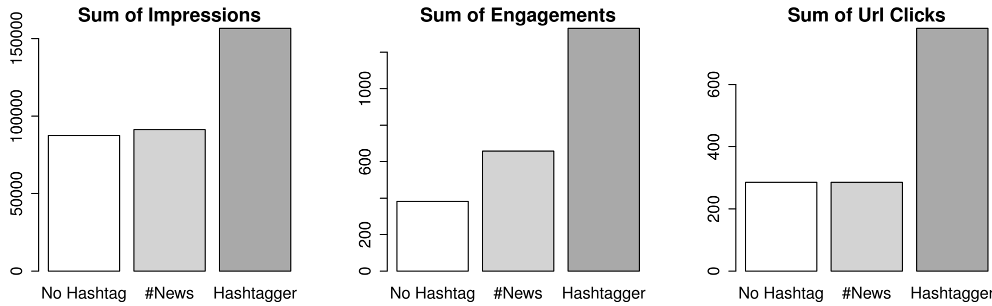

Figure 7: Twitter Analytics metrics to measure impact of hashtagging on news engagement.

ically 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 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, #greferendum, #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, #ericgarner, #blacklivesmatter, #icantbreathe).
3. Hashtags allow the query to focus on diverse aspects of a story (e.g., Greek economic crisis, #grexit, #greferendum, #tsipras, #merkel, #ecb, #imf, #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[8] (Twitter hashtag #irishwater), the inquiry into the banking collapse[9] of 2008 (Twitter hashtag #bankinginquiry), the recent marriage equality referendum[10] (Twitter hashtag #marref). Similarly, the BBC and The Guardian also publish story-pages, e.g., the BBC story-page on the "Greek debt crisis"[11] and Guardian page on Liberia[12]. Preparing these story-pages currently relies on prior agreement among 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[13] 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] `http://www.irishtimes.com/news/water-charges`
[9] `http://www.irishtimes.com/news/banking-inquiry`
[10] `http://www.irishtimes.com/news/politics/marriage-referendum`
[11] `http://www.bbc.com/news/world-europe-33225461`
[12] `http://www.theguardian.com/world/ebola`
[13] `http://insight4news.ucd.ie/insight4news/hashtag/\%23migrant`

8. REFERENCES

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96. ACM, 2005.

[2] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[3] C. Castillo, M. El-Haddad, J. Pfeffer, and M. Stempeck. Characterizing the life cycle of online news stories using social media reactions. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 211–223. ACM, 2014.

[4] J. Cheng, L. Adamic, P. A. Dow, J. M. Kleinberg, and J. Leskovec. Can cascades be predicted? In *Proceedings of the 23rd international conference on World wide web*, pages 925–936. ACM, 2014.

[5] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2002.

[6] Z. Ding, X. Qiu, Q. Zhang, and X. Huang. Learning topical translation model for microblog hashtag suggestion. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2078–2084. AAAI Press, 2013.

[7] R. Dovgopol and M. Nohelty. Twitter hash tag recommendation. *arXiv preprint arXiv:1502.00094*, 2015.

[8] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[9] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *J. Mach. Learn. Res.*, 15(1):3133–3181, Jan. 2014.

[10] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of machine learning research*, 4:933–969, 2003.

[11] J. Fürnkranz and E. Hüllermeier. Pairwise preference learning and ranking. In *Machine Learning: ECML 2003*, pages 145–156. Springer, 2003.

[12] F. Godin, V. Slavkovikj, W. De Neve, B. Schrauwen, and R. Van de Walle. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 593–596. International World Wide Web Conferences Steering Committee, 2013.

[13] L. Hang. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011.

[14] J. Harding. Future of news. *BBC*, 2015.

[15] T.-A. Hoang-Vu, A. Bessa, L. Barbosa, and J. Freire. Bridging vocabularies to link tweets and news.

[16] Z. D. Q. Z. X. Huang. Automatic hashtag recommendation for microblogs using topic-specific translation model. In *24th International Conference on Computational Linguistics*, page 265. Citeseer, 2012.

[17] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[18] P. Li, Q. Wu, and C. J. Burges. Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in neural information processing systems*, pages 897–904, 2007.

[19] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[20] Z. Ma, A. Sun, Q. Yuan, and G. Cong. Tagging your tweets: A probabilistic modeling of hashtag annotation in twitter. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 999–1008. ACM, 2014.

[21] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[22] A. Mazzia and J. Juett. Suggesting hashtags on twitter. *EECS 545m, Machine Learning, Computer Science and Engineering, University of Michigan*, 2009.

[23] R. Nallapati. Discriminative models for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71. ACM, 2004.

[24] N. Naveed, T. Gottron, J. Kunegis, and A. C. Alhadi. Bad news travel fast: A content-based analysis of interestingness on twitter. In *Proceedings of the 3rd International Web Science Conference*, page 8. ACM, 2011.

[25] C. Quoc and V. Le. Learning to rank with nonsmooth cost functions. *Proceedings of the Advances in Neural Information Processing Systems*, 19:193–200, 2007.

[26] D. Sculley. Combined regression and ranking. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 979–988. ACM, 2010.

[27] B. Shi, G. Ifrim, and N. Hurley. Be in the know: Connecting news articles to relevant twitter conversations. *arXiv preprint arXiv:1405.3117*, 2014.

[28] X. Si and M. Sun. Tag-lda for scalable real-time tag recommendation. *Journal of Computational Information Systems*, 6(1):23–31, 2009.

[29] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles. Real-time automatic tag recommendation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 515–522. ACM, 2008.

[30] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: a ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 383–390. ACM, 2007.

[31] Twitter. Twitter.

[32] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. Elsevier, 2011.

[33] F. Xiao, T. Noro, and T. Tokuda. News-topic oriented hashtag recommendation in twitter based on characteristic co-occurrence word detection. In *Web Engineering*, pages 16–30. Springer, 2012.

[34] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[35] S.-H. Yang, A. Kolcz, A. Schlaikjer, and P. Gupta. Large-scale high-precision topic modeling on twitter. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1907–1916. ACM, 2014.

[36] H. Yu, C. Ho, Y. Juan, and C. Lin. Libshorttext: A library for short-text classification and analysis. Technical report, Technical Report. http://www. csie. ntu. edu. tw/˜ cjlin/ papers/libshorttext. pdf, 2013.