Adding Semantics to Microblog Posts

Edgar Meij
edgar.meij@uva.nl

Wouter Weerkamp
w.weerkamp@uva.nl

Maarten de Rijke
derijke@uva.nl

ISLA, University of Amsterdam
Amsterdam

ABSTRACT

Microblogs have become an important source of information for the purpose of marketing, intelligence, and reputation management. Streams of microblogs are of great value because of their direct and real-time nature. Determining what an individual microblog post is about, however, can be non-trivial because of creative language usage, the highly contextualized and informal nature of microblog posts, and the limited length of this form of communication.

We propose a solution to the problem of determining what a microblog post is about through semantic linking: we add semantics to posts by automatically identifying concepts that are semantically related to it and generating links to the corresponding Wikipedia articles. The identified concepts can subsequently be used for, e.g., social media mining, thereby reducing the need for manual inspection and selection. Using a purpose-built test collection of tweets, we show that recently proposed approaches for semantic linking do not perform well, mainly due to the idiosyncratic nature of microblog posts. We propose a novel method based on machine learning with a set of innovative features and show that it is able to achieve significant improvements over all other methods, especially in terms of precision.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software

General Terms

Algorithms, Measurement, Performance, Experimentation

1. INTRODUCTION

In recent years Twitter has become one of the largest online microblogging platforms with over 65M unique visitors and around 200M tweets per day.[1] Microblogging streams have become invaluable sources for many kinds of analyses, including online reputation management, news and trend detection, and targeted marketing and customer services [4, 18, 32, 35]. Searching and mining microblog streams offers interesting technical challenges, because of the sheer volume of the data, its dynamic nature, the creative language usage, and the length of individual posts [17, 22].

In many microblog search scenarios the goal is to find out what people are saying about concepts such as products, brands, persons, et cetera [31]. Here, it is important to be able to accurately retrieve tweets that are on topic, including all possible naming and other lexical variants. So, it is common to manually construct lengthy keyword queries that (hopefully) capture all possible variants [2]. We propose an alternative approach, namely to determine what a microblog post is about by automatically identifying *concepts* in them. We take a concept to be any item that has a unique and unambiguous entry in a well-known large-scale knowledge source, Wikipedia.

Little research exists on understanding and modeling the semantics of individual microblog posts. Linking free text to knowledge resources, on the other hand, has received an increasing amount of attention in recent years. Starting from the domain of named entity recognition (NER), current approaches establish links not just to entity types, but to the actual entities themselves [15, 20, 30]. Instead of merely identifying types, we also aim to disambiguate the found concepts and link them to Wikipedia articles. With over 3.5 million articles, Wikipedia has become a rich source of knowledge and a common target for linking; automatic linking approaches using Wikipedia have met with considerable success [14, 25, 27, 28].

Most, if not all, of the linking methods assume that the input text is relatively clean and grammatically correct and that it provides sufficient context for the purposes of identifying concepts. Microblog posts are short, noisy, and full of shorthand and other ungrammatical text and provide very limited context for the words they contain [17, 22]. Hence, it is not obvious that automatic concept detection methods that have been shown to work well on news articles or web pages, perform equally well on microblog posts.

We present a robust method for automatically mapping tweets to Wikipedia articles to facilitate social media mining on a semantic level. The first research question we address is: What is the performance of state-of-the-art approaches for linking text to Wikipedia in the context of microblog posts? Our proposed approach involves a two-step method for semantic linking. The first step is recall-oriented where the aim is to obtain a ranked list of candidate concepts. In the next step, we enhance precision and determine which of the candidate concepts to keep. Our second research question concerns a comparison of methods for the initial concept ranking step; we consider lexical matching, language modeling, and other state-of-the-art baselines and compare their effectiveness. Our third

[1] http://blog.twitter.com/2011/06/200-million-tweets-per-day.html

WSDM'12, February 8–12, 2012, Seattle, Washington, USA.
Copyright 2012 ACM 978-1-4503-0747-5/12/02 ...$10.00.

Tweet	Concepts
Is it me or does Google Instant encourage you to pay more attention to their Ads and Shopping links?	ADS, AND, ATTENTION, DOES, GOOGLE, GOOGLE INSTANT, IS, IT, LINKS, ME, MORE, *etc.*
Keep your eyes out for an actress called Judi Dench. She's a promising talent and I predict we'll be hearing more about her.	A, ABOUT, ACTRESS, AN, AND, AND I, BE, CALLED, DENCH, FOR, HEARING, HER, I, I PREDICT, JUDI DENCH, *etc.*

Table 1: Example tweets with concepts recognized using lexical matching on Wikipedia article titles.

research question concerns the second, precision-enhancing step. We approach this as a machine learning problem and consider a broad set of features, some of which have been proposed previously in the literature on semantic linking, some newly introduced. In addition to multiple features, we also consider multiple machine learning algorithms and examine which of these are most effective for our problem. Finally, we examine the relative effectiveness of the precision-enhancing step on top of different initial concept ranking methods. The paper focuses on the *effectiveness* of concept detection methods in the setting of microblog posts. In the conclusion to the paper we also discuss efficiency considerations.

Our main contributions are: (i) a robust, successful method for linking tweets to Wikipedia articles, based on a combination of high-recall concept ranking and high-precision machine learning, including state-of-the-art machine learning algorithms, (ii) insights into the influence of various features and machine learning algorithms on the task, and (iii) a reusable dataset, with which we aim to facilitate follow-up research. The remainder of this paper is organized as follows. In Section 2 we discuss related work, followed by a description of our method. In Section 4 we discuss the experimental setup and, in Section 5, the experiments with which we answer our research questions. We end with a concluding section.

2. RELATED WORK

In this section we review related work, pertaining to semantic linking and to Twitter in particular.

2.1 Linking Text

Links to a knowledge structure are often seen as a way of providing semantics to digital items. The idea has been used for different media types (such as text [27, 28] and multimedia [34]) and for different text genres (such as news pages [9], queries [23], archives [6], and radiology reports [14]). A simple and frequently taken approach for linking text to concepts is to perform lexical matching between (parts of the text) and the concept titles [10, 26], an approach related to keyword-based interfaces to databases [38]. However, merely matching an input text with concept titles suffers from many drawbacks, including ambiguity (where different concepts with the same label can be confused) and a possible lack of specificity (in which case less "meaningful" concepts are identified). Table 1 shows two example tweets with concepts identified using lexical matching between word n-grams in the tweets and Wikipedia titles. From these examples it is obvious that, while relevant concepts such as GOOGLE INSTANT and JUDI DENCH are identified, this approach also retrieves many false positives.

In general, such issues can be addressed on either the Wikipedia or on the textual side. That is, we can reduce the amount of "noisy" concepts, e.g., by applying some kind of filtering or weighing, or we can clean up the text, e.g., by identifying segments, key phrases, or by weighing terms based on various heuristics, including stopwords, document frequency, et cetera. Our approach incorporates both; it first determines a high-recall ranking of concepts for each part of the tweet and, next, applies machine learning on the concepts to improve precision. Since we adopt a machine learning framework, we can also include Twitter-specific features. The method proposed by Milne and Witten [28] yields accurate results on relatively clean input texts, such as those from Wikipedia entries and news articles. For comparative purposes, we include their method as one of the baselines and show that it does not perform well on tweets. Ferragina and Scaiella [11] propose an approach similar to [28], but with an explicit focus on short texts. They incorporate a voting scheme as well as pruning of n-grams unrelated to the input text. Since this method is geared towards short texts, we include it as a baseline below.

2.2 Twitter

Twitter provides its users facilities to share short text messages, comprising a maximum of 140 characters. Tweets are published publicly by default, making Twitter an enormous resource of casual information exchanges. As Twitter has grown, novel language use and standards such as mentions (to reference another user), hashtags (to refer to a topic), and retweets (similar to an e-mail forward) have emerged [37].

Various authors have attempted to "give meaning" to text in general [24] or to text contained in tweets. Liu et al. [20] focus on NER on tweets and use a semi-supervised learning framework to identify four types of entities. Benson et al. [3] try to match tweets to "records." These records are, for example, artist-venue pairs and can be obtained from sources like music guides. They train a model that extracts artists and venues from tweets and automatically match these to the extracted records. Our approach is more general and aims at enriching tweets rather than extracting information from them. Our approach could therefore simplify the task proposed in [3].

Abel et al. [1] aim at contextualizing tweets, a task very similar to ours. After adding context, the authors use the tweets to profile Twitter users. Their approach depends on matching tweets to news articles, followed by semantic enrichment based on the news article's content. Finally, the semantically enriched tweets are used for user modeling. For the second step, semantic enrichment, the authors use OpenCalais. Our approach differs from their approach in that we do not assume tweets to be related to news articles, making our approach more general.

Mendes et al. [26] propose Linked Open Social Signals, a framework that includes annotating tweets with information from Linked Data. Their approach is rather straightforward and involves either looking up hashtag definitions or lexically matching strings to recognize (DBpedia) entities in tweets. We include the former as a feature in our framework and evaluate the latter as a baseline.

Kwak et al. [18] show that hashtags are good indicators to detect events and trending topics and Laniado and Mika [19] explore the use of hashtags in Twitter and the relation to (Freebase) concepts. Using manual annotations, they find that about half of the hashtags can be mapped to Freebase concepts, most of them being named entities. In a few cases, more general hashtags are mapped to concepts. Assessors showed high agreement on the task of mapping tags to concepts. The authors make the assumption that hashtags are mainly used to "ground" tweets, an assumption we lift in our work, enabling us to add semantics to tweets without hashtags. Previous work has shown that hashtag usage is quite low and differs a lot per country and language [36]. Huang et al. [16] analyze the semantics of hashtags in more detail and reveal that hashtagging in Twitter is more commonly used to join public discussions than to organize content for future retrieval. In order to verify the relative contribution of hashtags, we include a feature that leverages the highest-voted hashtag definition.

3. LINKING POSTS TO CONCEPTS

As we have seen in the previous section, linking text to concepts is most commonly approached by lexically matching the input text with the concept labels, i.e., the titles of the Wikipedia articles. Due to the noisy nature of microblog posts, this approach does not work well on our task, as we will see below. Furthermore, a large number of concept titles will match with any part of the tweet, as illustrated by the example in Table 1.

We therefore need a mechanism to improve precision, not only by removing spurious target concepts but also by limiting the number of n-grams used to generate a mapping. We approach this in two steps. The goal of the first step is obtain high recall, so we generate a ranked list of candidate concepts for each n-gram in a tweet. Moreover, using n-grams in this step also means we can keep track of which n-gram links to which concept. In the second step we aim to improve precision by applying supervised machine learning. Here, each candidate concept c is classified as being relevant or not (in the context of the tweet and the user). In a way, this setup is a form of learning to rerank. Alternatively, we could forego the first step (in which we obtain a set of candidate concepts) and apply a form of learning to rank on *all* concepts for each n-gram. However, since semantic linking is akin to known-item finding and thus geared towards high precision—with only a relatively small number of relevant concepts per tweet—we reduce the set of candidate concepts prior to applying machine learning. Moreover, limiting the set of concepts to be used as input for the machine learning algorithm also reduces the number of feature vectors that need to be created, decreasing the runtime.

3.1 Concept Ranking

In order to be able to identify the part of the tweet that is the source for a semantic link, we first identify word n-grams in the tweet. That is, we extract all possible n-grams from a tweet Q (where $1 \leq n \leq |Q|$). For each n-gram q in this set of n-grams, $\{q_i\}_{i=1}^{|Q|!}$, we generate a ranked list of candidate concepts. Various methods exist for creating a ranked list of concepts for an n-gram and in our experiments below we compare three families of approaches that are further detailed in Section 4. They include lexical matching, language modeling, and other state-of-the-art methods.

3.2 Machine Learning

Once we have obtained a ranked list of candidate concepts for each n-gram, we turn to concept selection. In this stage we need to decide which of the concepts are most viable. We use supervised machine learning, that takes as input a set of labeled examples (tweet to concept mappings) and several features of these examples (detailed below).

For training, each n-gram q in each tweet Q is associated with a set of concepts C_q and a set of associated relevance assessments for the concepts. The latter is created by considering all concepts that an annotator identified for the tweet. If a concept was not selected by any annotator, we consider it to be non-relevant for Q. Section 4.3 further details the specifics of the manual annotations. As we are performing semantic linking at the tweet level (instead of doing named entity resolution) it is sufficient to have a mapping from tweets to concepts for our task. So, we consider each combination of n-gram and concept an instance, represented by a high-dimensional feature vector. Its label indicates the concept's binary relevance to the tweet and, hence, the n-gram. The goal of the machine learning algorithm is to learn a function that outputs a relevance status for any new n-gram and concept pair given a feature vector of this new instance. In the remainder of this section we detail the used features; in Section 4.6 we introduce the machine learning algorithms.

Since we consider each Wikipedia article to be a concept, we have an obvious textual representation to use for extracting our textual features. Below we discern different parts of this representation (which we call "fields") for feature extraction and concept ranking. The fields we include are: `title` (the title of the article associated with concept c), `sentence` (the first sentence of the article), `paragraph` (the first paragraph of the article), `content` (the full contents of the article), and `anchor` (the aggregated anchor texts of all incoming links in Wikipedia). We employ several types of features, each associated with either an n-gram, concept, or their combination. We also include a separate set of Twitter-specific features.

3.2.1 N-gram Features

This set of features is solely based on information from an n-gram and are listed in Table 2 (first group). Here, $IDF(q)$ indicates the relative number of concepts in which q occurs, which is defined as $IDF(q) = \log(|\mathcal{C}|/df(q))$, where $|\mathcal{C}|$ indicates the total number of concepts and $df(q)$ the number of concepts in which q occurs [21]. The subscript f denotes the field of the Wikipedia articles used, see above.

$WIG(q)$ indicates the weighted information gain [39], which can be considered a predictor of the retrieval performance of a query. It uses the set of all candidate concepts retrieved for this n-gram, C_q, and determines the relative probability of q occurring in these documents as compared to the collection. Formally:

$$WIG(q) = \frac{\frac{1}{|C_q|}\sum_{c \in C_q} \log(P(q|c)) - \log(P(q))}{\log P(q)}.$$

$SNIL(q)$ and $SNCL(q)$ are an indicator of the polysemy of an n-gram and determine the number of Wikipedia articles whose title matches (part of) q. Similarly, we leverage the Wikipedia anchors to determine the probability that q is used as an anchor in Wikipedia. Let $C_a(q)$ be the set of Wikipedia articles that contain the n-gram q as an anchor and $df(q)$ the total number of Wikipedia articles in which q appears. $KEYPHRASENESS(q)$, then, determines the probability $|C_a(q)|/df(q)$ [27]. Similarly, we determine this probability based on all occurrences, also including multiple occurrences in an article:

$$LINKPROB(q) = \frac{\sum_{c \in C_a(q)} n(q,c)}{\sum_{c' \in \mathcal{C}} n(q,c')},$$

where $n(q,c)$ is the count of q in the content of the textual representation of c.

3.2.2 Concept Features

Table 2 (second group) lists the concept-related features. This set relates to the knowledge we have of each candidate concept, such as the number of other Wikipedia articles linking to or from it, the number of associated categories, the number of redirect pages pointing to it, and the number of terms or characters in its title. The last two features in this set capture the relative popularity of an Wikipedia article and are based on Wikipedia access logs.[2] That is,

[2] See http://dammit.lt/wikistats/.

N-gram features	
$LEN(q) = \vert q \vert$	Number of terms in the n-gram q
$IDF_f(q)$	Inverse document frequency of q in representation f, where $f \in$ {title, anchor, content}
$WIG(q)$	Weighted information gain using top-5 retrieved concepts
$KEYPHRASENESS(q)$	Probability that q is used as an anchor text in Wikipedia
$LINKPROB(q)$	Probability that q is used as an anchor text in Wikipedia (all occurrences)
$SNIL(q)$	Number of articles whose title equals a sub-n-gram of q
$SNCL(q)$	Number of articles whose title match a sub-n-gram of q
Concept features	
$INLINKS(c)$	Number of Wikipedia articles linking to c
$OUTLINKS(c)$	Number of Wikipedia articles linking from c
$GEN(c)$	Function of depth of c in the Wikipedia category hierarchy
$CAT(c)$	Number of categories associated with c
$REDIRECT(c)$	Number of redirect pages linking to c
$WLEN(c)$	Number of terms in the title of c
$CLEN(c)$	Number of characters in the title of c
$WIKSTATS(c)$	Number of times c was visited in 2010
$WIKISTATSWK(c)$	Number of times c was visited in the seven days before the tweet was posted
N-gram + concept features	
$TF_f(c,q) = \frac{n_f(q,c)}{\vert f \vert}$	Relative phrase frequency of q in representation f of c, normalized by length of f, where $f \in$ {title, anchor, first sentence, first paragraph, content}
$POS_n(c,q) = pos_n(q)/\vert c \vert$	Position of nth occurrence of q in c, normalized by length of c
$SPR(c,q)$	Distance between the first and last occurrence of q in c
$TF \cdot IDF(c,q)$	Importance of q for c
$RIDF(c,q)$	Residual IDF (difference between expected and observed IDF)
$\chi^2(c,q)$	χ^2 test of independence between q in c and all concepts
$NCT(c,q)$	Does q contain the title of c?
$TCN(c,q)$	Does the title of c contain q?
$TEN(c,q)$	Does the title of c equal q?
$SCORE(c,q)$	Language modeling score of c with respect to q
$RANK(c,q)$	Retrieval rank of c with respect to q
$COMMONNESS(c,q)$	Probability of c being the target of a link with anchor text q
Tweet features	
$TWCT(c,Q)$	Does Q contain the title of c?
$TCTW(c,Q)$	Does the title of c contain Q?
$TETW(c,Q)$	Does the title of c equal Q?
$TAGDEF(q,Q)$	Number of times q appears in the hashtag definition of any hashtag in tweet Q
$URL(q,Q)$	Number of times q appears in a webpage linked to by Q

Table 2: Features used, grouped by type. More detailed descriptions in Section 3.2.

we determine for each Wikipedia article the normalized frequency with which it was visited in 2010 ($WIKSTATS(c)$) and the frequency with which it was visited in the seven days before the tweet was posted ($WIKSTATSWK(c)$).

3.2.3 N-gram + Concept Features

This set of features considers the combination of an n-gram and a concept (Table 2, third group). Here, we first consider the relative frequency of occurrence of q in separate concept fields, the position of the first occurrence of the n-gram, the distance between the first and last occurrence, and various IR-based measures [21]. Of these, $RIDF$ [8] is the difference between expected and observed IDF for a concept, which is defined as

$$RIDF(c,q) = \log\left(\frac{|\mathcal{C}|}{df(q)}\right) + \log\left(1 - \exp\left(\frac{-n(q,\mathcal{C})}{|\mathcal{C}|}\right)\right).$$

We also consider whether the title of the Wikipedia article matches q in any way.

For the features $SCORE(c,q)$ and $RANK(c,q)$ we adopt a language modeling framework, in which a query is viewed as having been generated from a multinomial language model and each document is scored according to the likelihood that the words in the query were generated by randomly sampling the document language model [21]. The word probabilities are estimated from the document itself (using maximum likelihood estimation) and combined with background collection statistics to overcome zero probability and data sparsity issues; a process known as smoothing. We calculate the score for a concept $c \in \mathcal{C}$ according to the probability that it was generated by the n-gram, $P(c|q)$, which can be rewritten using Bayes' rule as: $P(c|q) = P(q|c)P(c)P(q)^{-1}$. Here, for a fixed n-gram q, the term $P(q)$ is the same for all concepts and can be ignored for ranking purposes. The term $P(c)$ indicates the prior probability of selecting a concept, which we assume to be uniform. We consider each n-gram to be a phrase and determine the probability $P(q|c)$ using the following estimate:

$$P(q|c) = \frac{n(q,c) + \mu P(q)}{\mu + |c|}. \quad (1)$$

This is an estimate using Bayes smoothing with a Dirichlet prior, where $P(q)$ indicates the probability of observing q in a large background collection. Here, μ is a hyperparameter that controls the influence of the background corpus and $|c|$ is the length of the textual representation of the concept.

Finally, we consider the prior probability that c is the target of a link with anchor text q in Wikipedia:

$$COMMONNESS(c,q) = \frac{|L_{q,c}|}{\sum_{c'} |L_{q,c'}|}, \quad (2)$$

where $L_{q,c}$ denotes the set of all links with anchor text q and target c.

3.2.4 Tweet Features

Finally, we consider features related to the entire tweet (Table 2, last group), including three kinds of lexical matches. We also fetch the webpage(s) linked to by the tweet (if any) and determine the frequency of occurrence of q in them ($URL(q,Q)$). Finally, we use the TagDef API[3] to lookup the highest-voted hashtag definition of all hashtags in the tweet (if any) and determine the frequency of occurrence of q in them ($TAGDEF(q,Q)$).

4. EXPERIMENTAL SETUP

In order to answer the research questions introduced in Section 1, we have designed several experiments. In this section we detail our experimental setup, including how we sample tweets, the manual annotations, the Wikipedia version, and how we evaluate linking tweets to concepts.

4.1 Tweets

Out of a sample of 2000 tweets, Pear Analytics [32] classified 40% as containing "pointless babble," with another 37.55% as merely conversational. So, a significant portion of all tweets are non-informative and only a small fraction contains topics of general interest. In order to account for this possible bias, we randomly sample users from the "verified accounts" Twitter list and, for each of these users, we retrieve the last 20 tweets.[4] The Twitter users in this list can be considered influential and their tweets are more likely to be picked up by other Twitter users than from a random sample of Twitter users. As to preprocessing, we record all URLs, mentions, and hashtags in each tweet. URLs are removed from each tweet, whereas mentions and hashtags are kept without the leading '@' and '#' respectively.

4.2 Wikipedia

As our target for linking tweets, we use a dump of Wikipedia that is dated 11 Oct. 2010. In this particular snapshot, we have 3,483,213 articles proper, 4,526,685 redirects, 120,547 disambiguation pages, and 71,204,142 hyperlinks between articles. For the `anchor` field we include not only the anchor texts found in intra-Wikipedia hyperlinks, but also the titles of any redirect pages pointing to an article. As to retrieval, we use the entire Wikipedia document collection as background corpus and set μ (cf. Eq. 1) to the average length of a Wikipedia article.

4.3 Manual Annotations

In order to obtain manual annotations (both for training and evaluation), we have asked two volunteers to manually annotate 562 tweets, each containing 36.5 terms on average. They were presented with an annotation interface with which they could search through Wikipedia articles using any of the fields defined above. The annotation guidelines specified that the annotator should identify concepts contained in, meant by, or relevant to the tweet. They could also indicate that an entire tweet was either ambiguous (where multiple target concepts exist) or erroneous (when no relevant concept could be assigned). Out of the 562 tweets, 419 were labeled as not being in either of these two categories and kept for further analysis. For these, the annotators identified 2.17 concepts per tweet on average. In order to facilitate follow-up research, we make all annotations and derived data available.[5]

4.4 Evaluation

We approach the task of linking tweets to concepts as a ranking problem; given a tweet, the goal of a system implementing a solution to this problem is to return a ranked list of concepts meant by or contained in it, where a higher rank indicates a higher degree of relevance of the concept to the tweet. The best performing method puts the most relevant concepts towards the top of the ranking. Our method identifies relevant concepts for each tweet's constituent n-grams and, since the semantic linking task is defined at the tweet instead of the n-gram level, we need a way to aggregate potential duplicate concepts. In our experiments, we aggregate each duplicate concept (if any) by naively summing the confidence scores for each source n-gram. Future work should indicate whether more elaborate methods perform better. The manual annotations described above are then used to determine the relevance status of each of the concepts with respect to a tweet; a concept is considered relevant if it was linked by an annotator. If it was not selected by any annotator, we consider it to be non-relevant.

The evaluation measures we employ include precision at rank 1 (P1), r-precision (R-prec), recall, mean reciprocal rank (MRR), and mean average precision (MAP) [21]. Together, these measures provide a succinct summary of the quality of the retrieved concepts. We use the top-50 returned concepts for evaluation. To test for statistical significance, we use a paired two-sided t-test (with pairing at the run level). We indicate the best performing run using boldface and a significant improvement with △ and ▲ (for $p < 0.05$ and $p < 0.01$ respectively) and, similarly, a significant decrease in performance with ▽ and ▼ (again for $p < 0.05$ and $p < 0.01$ respectively). Unless indicated otherwise, we test for significance against the top-most row in a table.

4.5 Baselines

We employ three sets of baselines to which we compare our approach and to which we apply supervised machine learning. They include: (i) lexical matching of the n-grams with the concepts, (ii) a language modeling baseline, and (iii) a set of other methods, augmented with using solely the $COMMONNESS(c,q)$ feature.

4.5.1 Lexical Match

As our first baseline we consider a simple heuristic which is commonly used [10, 26] and select concepts whose title lexically matches any n-gram in the tweet, similar to the $NCT(c,q)$ and $TEN(c,q)$ features described in Section 3.2. We subsequently rank the concepts based on the language modeling score of their

[3] See `http://tagdef.com/`.

[4] See `http://twitter.com/help/verified/`.

[5] See `http://ilps.science.uva.nl/resources/wsdm2012-adding-semantics-to-microblog-posts/`.

associated Wikipedia article given the tweet. This method is the most naive, yet most commonly used approach; in sum, it returns concepts for which consecutive terms in the title of the Wikipedia article are contained in the n-gram.

An example is given in Table 1. As a variant, we apply the same approach to the `anchor` field. We also apply an heuristic to handle n-grams; this method starts with the largest n-grams in the tweet and checks to see if it lexically matches with a concept. If it does, it discards any smaller constituent n-grams. If the n-gram doesn't match it recurses with its constituent n-grams.

4.5.2 Retrieval

This baseline builds on the approach described by Eq. 1 and is similar to using a search engine and performing a search within Wikipedia. That is, we use language modeling to determine the similarity of a tweet with the concepts. In particular, we explore two different tweet representations and three different concept fields, resulting in six retrieval baseline runs. For the tweet representation we try either the full tweet or the constituent n-grams. In the case of ranking concepts for the entire tweets, we assume independence between the individual terms $t \in Q$:

$$P(c|Q) \quad \propto \quad P(c) \prod_{t \in Q} P(t|c)^{n(t,Q)}. \tag{3}$$

The probability $P(t|c)$ is again smoothed using Bayes smoothing with a Dirichlet prior, in this case formulated as:

$$P(t|c) = \frac{n(t,c) + \mu P(t)}{\mu + \sum_{t'} n(t',c)}. \tag{4}$$

As to the concept fields we use either the full Wikipedia article, its title, or its incoming anchor texts.

To combine the rankings produced by each constituent n-gram of a tweet, we use combMNZ [33]. CombMNZ is a result list merging method and a variant of CombSUM—which sums a document's scores from all lists where it was retrieved. CombMNZ multiplies the CombSUM score by the number of lists that contained the particular document. Formally, let $q_1, \ldots, q_K$ be the term n-grams in the tweet and $C_1, \ldots, C_K$ the concept rankings corresponding to each n-gram. Then,

$$\text{CombSUM}(c) \quad = \quad \sum_{k: c \in C_k} P(c|q_k), \tag{5}$$

$$\text{CombMNZ}(c) \quad = \quad \text{CombSUM}(c) \cdot |\{k : c \in C_k\}|. \tag{6}$$

4.5.3 Other Methods

The final set of baselines that we consider comprises three established methods, including the one proposed by Milne and Witten [28], denoted M&W, which represents the state-of-the-art in automatic linking approaches. We use the algorithm and best-performing settings as described in [28], trained on our version of Wikipedia. We also include a novel service provided by DBpedia, called DBpedia Spotlight.[6] The third baseline in this set (Tagme) is provided by Ferragina and Scaiella [11]. These first three baselines in this set do not perform optimally when linking individual n-grams. Therefore, we use the entire, cleaned tweet as input. Our last concept ranking method in this set (CMNS) corresponds to the $COMMONNESS(c, q)$ feature, detailed by Eq. 2. This method scores each concept based on the relative frequency with which the n-gram is used as an anchor text for that particular concept. We exclude OpenCalais from this set, since this webservice only recognizes entity types without performing any kind of disambiguation, similar to returning Wikipedia disambiguation pages. Moreover, the precise algorithmic details are not made public.

4.6 Machine Learning Methods

For all machine learning algorithms, we perform 5-fold cross-validation at the tweet level, in order to reduce the possibility of errors being caused by artifacts in the data and to verify the generalizability to unseen data. The reported scores are averaged over all testing folds. We experiment with multiple machine learning algorithms in order to confirm that our results are generally valid, i.e., not dependent on any specific algorithm. Following Milne and Witten [28], we include a Naive Bayes (NB), Support Vector Machines (SVM), and a C4.5 decision tree classifier.

Random forests (RFs) are a very efficient alternative to C4.5, since they are (i) relatively insensitive to parameter settings, (ii) resistant to overfitting, and (iii) easily parallelizable [5]. It is an ensemble-based decision tree classifier based on bagging, in which a learning algorithm is applied multiple times to a subset of the instances and the results averaged. In this case, for each iteration a bootstrap sample is taken and a full tree is constructed. For each node of the tree, m features are randomly selected to obtain the best split. This process reduces overfitting by averaging classifiers that are trained on different subsets of the data with the same underlying distribution. We set m to roughly 10% of the size of the feature set, i.e., $m = 4$. Besides m, RF has one additional parameter: the number of iterations, k. We set $k = 1000$, but also report on the effect of varying this setting on the linking effectiveness.

In recent years, gradient boosted regression trees (GBRTs) have been established as the de facto state-of-the-art learning paradigm for web search ranking [7, 12, 29]. It is a point-wise learning to rank algorithm that predicts the relevance score of a result to a query by minimizing a loss function (e.g., the squared loss) using stochastic gradient descent. It is similar to RF in that it is also based on tree averaging. In this case, however, many low-depth trees (instead of full ones) are sequentially created, each with a bias towards instances that are responsible for the current regression error. Let $\mathbf{q}_i$ be the feature vector associated with q_i, y_i the associated label, i.e., relevance status, and $\mathcal{T}(\mathbf{q}_i)$ the current prediction for q_i. Then assume we have a continuous, convex, and differentiable loss function $\mathcal{L}(\mathcal{T}(\mathbf{q}_1), \ldots, \mathcal{T}(\mathbf{q}_n))$ that reaches its minimum if $\mathcal{T}(\mathbf{q}_i) = y_i$ for all $\mathbf{q}_i$. GBRT performs gradient descent in the instance space, where the current prediction is updated with a gradient step,

$$\mathcal{T}(\mathbf{q}_i) \leftarrow \mathcal{T}(\mathbf{q}_i) - \alpha \frac{\mathcal{L}}{\mathcal{T}(\mathbf{q}_i)} \tag{7}$$

during each iteration. Here, $\alpha > 0$ denotes the learning rate. For our experiments we use the the square loss, which is defined as: $\mathcal{L} = \frac{1}{2} \sum_{i=1}^{n} (\mathcal{T}(\mathbf{q}_i) - y_i)^2$. So, GBRT depends on three parameters: the learning rate α, the depth of the tree, d, and the number of iterations, k. We set $\alpha = 0.02$ and $k = 1000$, and, following Hastie et al. [13], we set $d = 4$.

Finally, Mohan et al. [29] show that, since RF is resistant to overfitting and also often outperforms GBRT, the RF predictions can be used as starting point for GBRT. By doing so, GBRT starts at a point relatively close to the global minimum and is able to further improve the already good predictions. We also include this approach, labeled iGBRT.

5. RESULTS AND DISCUSSION

In this section we answer the research questions that we identified in Section 1 by presenting, comparing, and discussing the results of the baselines and our method.

[6] See `http://dbpedia.org/spotlight`.

	P1	R-prec	Recall	MRR	MAP
Lexical match					
`anchor`	0.0366	0.0422	0.1592	0.0796	0.0485
`title`	0.0262	0.0160	0.0616	0.0338	0.0205
Lexical match, longest n-gram					
`anchor`	**0.1990**	**0.1344**	0.2408	**0.2440**	**0.1405**
`title`	0.1885	0.1335	**0.2530**	0.2345	0.1372

Table 3: First set of baseline results, obtained using lexical matching on titles or anchor texts (first group) and applying the longest n-gram heuristic (second group).

	P1	R-prec	Recall	MRR	MAP
Tweet					
`content`	0.1492	0.1173	0.2464	0.2255	0.1396
`title`	0.1597	0.1185	0.1901$^{\blacktriangledown}$	0.2106	0.1261
`anchor`	**0.2932**$^{\blacktriangle}$	**0.2147**$^{\blacktriangle}$	0.3719$^{\blacktriangle}$	0.3793$^{\blacktriangle}$	0.2415$^{\blacktriangle}$
N-grams					
`content`	0.2539	0.1939	**0.6864**$^{\blacktriangle}$	0.3910	0.2664
`title`	0.2120$^{\blacktriangledown}$	0.1552$^{\blacktriangle}$	0.3985	0.2795$^{\blacktriangledown}$	0.1818$^{\blacktriangledown}$
`anchor`	0.2644	0.1994	0.6686$^{\blacktriangle}$	**0.4003**$^{\blacktriangle}$	**0.2810**$^{\blacktriangle}$

Table 4: Retrieval results using two tweet representations and three fields. Significance is tested against line 1 (first group) and line 3 (second group).

5.1 Establishing a Baseline

Since the machine learning step takes as input a ranked list of concepts for each n-gram in a tweet, the ideal baseline method should have a high recall with sufficient precision. Table 3 shows the results of using lexical matching to obtain a ranked list of concepts. Merely matching parts of the tweet with titles or anchors does not perform well. Note that this is a commonly taken approach, but these results indicate that it fails on tweets. When we apply the longest n-grams heuristic we observe a notable increase in concept ranking performance, in terms of recall and precision.

We now turn to the results for the retrieval baseline, using different fields and ways of handling the tweet as introduced earlier. Table 4 (first group) shows the results for ranking concepts based on the entire tweet, assuming independence between the terms in the tweet. This is a rather naive approach, but performs remarkably well. Especially when we only use the anchor texts of each Wikipedia article, we obtain a P1 value of 0.2932, i.e., the first returned concept is relevant for almost 30% of the tweets. Moreover, the average rank of of the first relevant concept lies around 3. Table 4 (second group) shows the results when we generate a ranking for each constituent n-gram and apply CombMNZ to merge these. We observe the following: First, recall improves significantly using n-grams and the `anchor` or `content` field. Second, both MAP and MRR increase significantly using the `anchor` field.

Table 5 shows the results of the last set of baselines. DBpedia Spotlight and the learning to link approach by Milne and Witten [28] achieve comparable results that slightly improve over the best language modeling baseline. The performance results for M&W are much lower than the results reported in [28], which can be attributed to the different nature of tweets as compared to Wikipedia and/or news articles. The Tagme system—designed especially for short texts—fares much better, achieving marked improvements.

	P1	R-prec	Recall	MRR	MAP
Spotlight	0.3717	0.2688	0.3068	0.4215	0.2658
M&W	0.3770	0.3033	0.3741	0.4255	0.3127
Tagme	0.5628	0.4643	0.5814	0.6339	0.4894
CMNS	**0.6021**	**0.5271**	**0.7775**	**0.7080**	**0.5853**

Table 5: Results for the third set of baselines.

	P1	R-prec	Recall	MRR	MAP
lex. match	0.1990	0.1344	0.2408	0.2440	0.1405
NB	0.3010$^{\blacktriangle}$	0.1996$^{\blacktriangle}$	0.2302	0.3440$^{\blacktriangle}$	0.2021$^{\blacktriangle}$
C4.5	0.4372$^{\blacktriangle}$	0.2553$^{\blacktriangle}$	0.2658	0.4503$^{\blacktriangle}$	0.2575$^{\blacktriangle}$
SVM	0.3194$^{\blacktriangle}$	0.2337$^{\blacktriangle}$	0.2943$^{\vartriangle}$	0.3878$^{\blacktriangle}$	0.2328$^{\blacktriangle}$
RF	**0.5681**$^{\blacktriangle}$	**0.4091**$^{\blacktriangle}$	**0.4872**$^{\blacktriangle}$	**0.6078**$^{\blacktriangle}$	**0.4222**$^{\blacktriangle}$
GBRT	0.5550$^{\blacktriangle}$	0.4034$^{\blacktriangle}$	0.4825$^{\blacktriangle}$	0.5964$^{\blacktriangle}$	0.4133$^{\blacktriangle}$
iGBRT	0.5654$^{\blacktriangle}$	0.4082$^{\blacktriangle}$	0.4819$^{\blacktriangle}$	0.6053$^{\blacktriangle}$	0.4209$^{\blacktriangle}$

Table 6: Results for the best lexical matching run with subsequent machine learning.

Simply applying the $COMMONNESS(c, q)$, i.e., CMNS, concept ranking method, however, achieves the highest scores so far, on all metrics. This relatively simple approach is able to retrieve over 75% of the relevant concepts and place the first relevant concept around rank 1.4 on average.

In conclusion, the results on the three baseline sets show that the task of linking tweets to concepts can be addressed relatively successfully; using a retrieval-based approach outperforms an approach based on mere lexical matching. Using the CMNS method obtains the highest performance, in terms of both precision and recall.

5.2 Applying Machine Learning

In this section we present the results of our approach, i.e., applying machine learning to further improve the results of the concept ranking step. That is, we take the best performing run per baseline set and apply the machine learning approaches introduced in Section 3.2. In particular, we select the "Lexical match, longest n-gram" for the first set and the "n-grams" retrieval for the second set, both using the `anchor` field. Finally, we select the CMNS run from the last set.

In Table 6 we show the results when we apply machine learning to the lexical matching baseline. All machine learning algorithms are able to significantly improve precision, MRR, and MAP. Only NB and C4.5 do not significantly improve in terms of recall. Here, RF obtains the highest scores overall, although the differences with GBRT and iGBRT are minimal and not significant.

Table 7 shows the results of the retrieval baseline, augmented by the machine learning step. Almost all machine learning algorithms are able to improve over the retrieval baseline, except for recall. Furthermore, iGBRT achieves the highest scores on most metrics, closely followed by RF.

Finally, in Table 8 we show the results of applying machine learning to the CMNS baseline. Of all the baselines, CMNS obtains the highest recall and precision levels; it does so by returning relatively few concepts per n-gram. We note that NB and C4.5 do not perform well in this case; their performance drops significantly for all metrics. Indeed, recall is significantly worse for NB, C4.5, and SVM, mainly due to the fact that a lot of concepts are identified as non-relevant by these algorithms. Interestingly, P1 improves significantly for SVM, while R-prec is significantly worse

	P1	R-prec	Recall	MRR	MAP
retrieval	0.2644	0.1994	**0.6686**	0.4003	0.2810
NB	0.4058$^{\blacktriangle}$	0.3212$^{\blacktriangle}$	0.4510$^{\blacktriangledown}$	0.4911$^{\blacktriangle}$	0.3450$^{\blacktriangle}$
C4.5	0.3822$^{\blacktriangle}$	0.2294	0.2530$^{\blacktriangledown}$	0.4106	0.2341$^{\blacktriangledown}$
SVM	0.2958	0.2419$^{\vartriangle}$	0.4947$^{\blacktriangledown}$	0.4138	0.2799
RF	0.5419$^{\blacktriangle}$	0.4197$^{\blacktriangle}$	0.6362$^{\blacktriangledown}$	0.6195$^{\blacktriangle}$	**0.4662**$^{\blacktriangle}$
GBRT	0.5079$^{\blacktriangle}$	0.3978$^{\blacktriangle}$	0.6362$^{\blacktriangledown}$	0.6012$^{\blacktriangle}$	0.4517$^{\blacktriangle}$
iGBRT	**0.5445**$^{\blacktriangle}$	**0.4209**$^{\blacktriangle}$	0.6349$^{\blacktriangledown}$	**0.6229**$^{\blacktriangle}$	0.4660$^{\blacktriangle}$

Table 7: Results of applying machine learning, applied to the best performing retrieval baseline.

	P1	R-prec	Recall	MRR	MAP
CMNS	0.6021	0.5271	0.7775	0.7080	0.5853
NB	0.4921$^{\blacktriangledown}$	0.4091$^{\blacktriangledown}$	0.5292$^{\blacktriangledown}$	0.5853$^{\blacktriangledown}$	0.4294$^{\blacktriangledown}$
C4.5	0.5209$^{\blacktriangledown}$	0.3542$^{\blacktriangledown}$	0.3774$^{\blacktriangledown}$	0.5585$^{\blacktriangledown}$	0.3528$^{\blacktriangledown}$
SVM	0.6708$^{\blacktriangle}$	0.4670$^{\blacktriangledown}$	0.4784$^{\blacktriangledown}$	0.6846	0.4711$^{\blacktriangledown}$
RF	**0.6780**$^{\blacktriangle}$	**0.5739**$^{\vartriangle}$	**0.8417**$^{\blacktriangle}$	**0.7676**$^{\blacktriangle}$	**0.6561**$^{\blacktriangle}$
GBRT	0.6649$^{\vartriangle}$	0.5624$^{\vartriangle}$	0.8361$^{\blacktriangle}$	0.7568$^{\blacktriangle}$	0.6421$^{\blacktriangle}$
iGBRT	0.6754$^{\vartriangle}$	0.5715$^{\vartriangle}$	0.8400$^{\blacktriangle}$	0.7644$^{\blacktriangle}$	0.6511$^{\blacktriangle}$

Table 8: Results of applying machine learning, with as input the best performing "other" baseline run, i.e. *COMMONNESS* (CMNS).

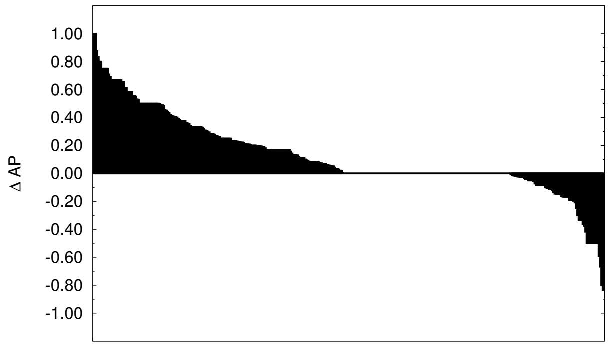

Figure 1: Per-tweet difference in AP between CMNS and CMNS-RF, where a positive value indicates an improvement over CMNS.

than CMNS. In many cases, SVM is able to push a single relevant concept to the top of the ranking, but fails to do so for all relevant concepts. RF, GBRT, and iGBRT obtain significant improvements over CMNS for all metrics; RF outperforms all other methods.

In sum, we find that the iterative machine learning methods obtain the best improvements overall and that they are able to significantly improve precision in all cases. In the remainder of this section we perform a more detailed analysis of the obtained results. We base these analyses mainly on the best performing method, i.e., machine learning using RF on the CMNS concept ranking, which we denote CMNS-RF.

5.3 Error Analysis

Figure 1 shows a per-tweet plot of the difference in terms of average precision (AP) between CMNS and CMNS-RF. As this figure indicates, most tweets are helped by applying machine learning, except for about a fifth of them. The tweets that are hurt most are mainly tweets with only a single relevant concept, that is retrieved by CMNS but classified as not relevant by CMNS-RF.

When we look at the errors being made in general, we observe that there are three typical cases of errors. First, there are concepts that are clearly relevant, but that were missed by the annotators. In some other cases, the concepts identified by the annotators do not always seem relevant. In a future study we intend to investigate this further, mainly by performing a post-hoc relevance analysis of annotated tweets. Second, we find that in some cases the machine learning algorithms seem to focus on a particular sense of an n-gram. E.g., in the case of "St. Patrick," most algorithms return various churches instead of the arguably more common concept SAINT PATRICK'S DAY. Finally, it also occurs that a non-content-bearing term in the tweet is being linked, such as the n-gram "wow" that gets linked to the concept WORLD OF WARCRAFT. In future work, we plan on tackling these two classes of errors, e.g., by including a post-ranking filter that first tries to match longer n-grams before moving on to shorter ones or by including more features that are related to the entire tweet; this way the remainder of the tweet might assist in disambiguating a particular n-gram.

5.4 Parameter Settings

In this section we consider the various parameters that are associated with the machine learning algorithms. For all of these, we have selected the top-ranked concepts returned by the concept ranking step, up to a maximum of 50 per n-gram. Varying this number does not significantly alter the obtained results.

Recall that RF, GBRT, and iGBRT have several parameters associated with them, including the number of iterations and the learning rate. In the case of gradient boosting there exists an inherent trade-off between the learning rate and the number of iterations. Ideally, the learning rate needs to be infinitesimally small and the number of iterations extremely large to obtain the true global minimum. In our case, varying the learning rate parameter α only influences the obtained end results very little.

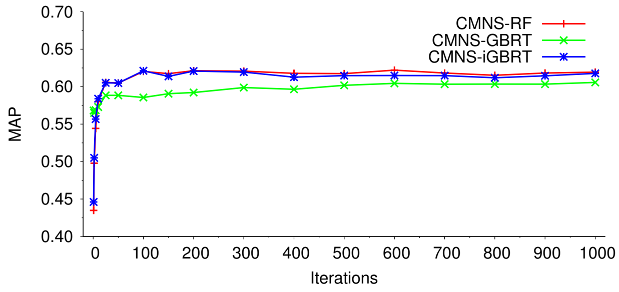

Figure 2: Effect of varying k on MAP.

Figure 2 shows the effect of varying the number of iterations using the RF, GBRT, and iGBRT machine learning algorithms on MAP. Both RF and iGBRT start at roughly the same level, since iGBRT uses the RF predictions as initialization weights. Both algorithms also reach their optimum after approximately 100 iterations, indicating that the iterative learning process can be stopped quite early. GBRT is not able to reach the same level of performance as iGBRT; it takes over 600 iterations to approach the others' scores.

5.5 Feature Analysis

Finally, we zoom in on the relative effectiveness of each of the features. The top-5 features with the highest discriminative power, measured using information gain, are, in decreasing order:

- $TEN(c, q)$,
- $TWCT(c, Q)$,
- $REDIRECT(c)$,
- $LINKPROB(q)$, and
- $IDF_{anchor}(q)$.

Interestingly, this is a mix between features that are based solely on the n-gram, the concept, and their combination. The binary feature that indicates whether the tweet contains the concept title is also a strong indicator for the relevance of a concept regarding a tweet.

In a follow-up experiment, we rank all features by their information gain and add each of them incrementally to the set of used features, starting with the features with the highest values. We keep the number of features RF selects at each iteration at $m = 4$ and thus start with selecting the 4 highest ranked features; Table 9 shows the results. CMNS-RF achieves MAP comparable to CMNS (on which this ranking is based) after adding the 12th feature, $TCN(c, q)$. When CMNS is subsequently added as a feature, we observe an improvement of 6% in retrieval performance. After $IDF_{content}(q)$ we truncate the table; MAP remains relatively constant around 0.65 after adding this feature. Almost all of the features shown here can be computed quite easily, either from an inverted index or from a cache that contains the link structure and anchor information of Wikipedia. The $URL(q, Q)$ and $WIG(q)$ features are the most costly features in this list. For the first an HTTP connection needs to be made and the web page content fetched, parsed, and matched with the n-gram. The second is a function over the n-gram and the top-ranked concepts for that n-gram. In a situation where the running time is of importance, we recommend removing these particular features.

Added feature	MAP	Δ
$TEN(c, q)$ / $TWCT(c, Q)$ / $REDIRECT(c)$ / $LINKPROB(q)$	0.4964	
+ $IDF_{anchor}(q)$	0.5452	9.83%
+ $KEYPHRASENESS(q)$	0.5523	1.30%
+ $SNIL(q)$	0.5502	-0.38%
+ $IDF_{title}(q)$	0.5547	0.82%
+ $SNCL(q)$	0.5593	0.83%
+ $TF_{paragraph}(c, q)$	0.5674	1.45%
+ $TF_{sentence}(c, q)$	0.5834	2.82%
+ $TCN(c, q)$	0.5866	0.55%
+ $COMMONNESS(c, q)$	0.6216	5.97%
+ $TF_{title}(c, q)$	0.6340	1.99%
+ $URL(q, Q)$	0.6405	1.03%
+ $POS_1(c, q)$	0.6435	0.47%
+ $GEN(c)$	0.6475	0.62%
+ $WIG(q)$	0.6491	0.25%
+ $IDF_{content}(q)$	0.6511	0.31%

Table 9: MAP of CMNS-RF after incrementally adding features proportional to their information gain (truncated to show only the top features).

6. CONCLUSION AND FUTURE WORK

Microblogging streams have become an invaluable resource for marketing, search, information dissemination, and online reputation management. Searching and mining microblog streams offers interesting challenges and in this paper we have presented a successful semantic linking method for microblog posts. The identified concepts, i.e., Wikipedia articles, can subsequently be used for, e.g., social media mining or advanced search result presentation. Our novel method uses machine learning and is based on a high-recall concept ranking and a high-precision concept selection step. Using a purpose-built test collection, we have shown that it significantly outperforms other methods, including various recently proposed approaches. Moreover, the concept selection step can be applied to any method that returns concepts for an input text. Our results show that this step, in particular using random forests or gradient boosted regression trees, can significantly improve a weak baseline, especially in terms of precision. It is even able to improve when the concept ranking performance is already strong on its own.

We have focused mainly on the effectiveness of semantic linking in the setting of microblog posts as opposed to the efficiency. Since both best performing machine learning algorithms are easily parallelizable, the bulk of the processing happens during feature extraction. From the results obtained during feature analysis, we note that not all features are equally important and that a minimal, easily computable set can already obtain good performance. Moreover, our analysis has shown that only a relatively small number of iterations is needed to achieve optimal performance. We finally note that, in the cases where a real-time analysis of a stream of microblog posts is required, merely using the low-cost CMNS feature already obtains very good performance.

Future work includes the following. First, although our method is not language-dependent in any way, the manual annotations are indeed language-specific. Wikipedia, on the other hand, already contains numerous, manually-curated inter-language links that we could use for this purpose. Second, we already mentioned a post-hoc evaluation of our semantic linking method for future work in Section 5.3. We also acknowledge that our sample of tweets, based on “authoritative users,” is comparatively small and might be biased. Therefore, we intend to apply the best-performing methods to a much larger, random sample of microblog posts to see how it performs there. Further, in this paper we have focused on a domain-independent way of obtaining high-recall concept candidate rankings. We believe, however, that including additional information such as from NER could further improve semantic linking performance. For future work we also intend to consider bootstrapping or co-training, in which the concepts with the highest confidence are fed back as new training material. Finally we note that Wikipedia contains a few thousand links to Twitter in the articles’ “External Links” sections and we intend to investigate to what extent we can use this information for semantic linking.

ACKNOWLEDGEMENTS

This research was supported by the European Union’s ICT Policy Support Programme as part of the Competitiveness and Innovation Framework Programme, CIP ICT-PSP under grant agreement nr 250430, the European Community’s Seventh Framework Programme (FP7/ 2007-2013) under grant agreements nr 258191 (PROMISE Network of Excellence) and 288024 (LiMoSINe project), the Netherlands Organisation for Scientific Research (NWO) under project nrs 612.061.814, 612.061.815, 640.004.802, 380-70-011, 727.011.005, the Center for Creation, Content and Technology (CCCT), the Hyperlocal Service Platform project funded by the Service Innovation & ICT program, the WAHSP project funded by the CLARIN-nl program, under COMMIT project Infiniti and by the ESF Research Network Program ELIAS.

REFERENCES

[1] F. Abel, Q. Gao, G.-J. Houben, and K. Tao. Semantic Enrichment of Twitter Posts for User Profile Construction on the Social Web. In *ESWC '11*, 2011.

[2] E. Amigó, J. Artiles, J. Gonzalo, D. Spina, B. Liu, and A. Corujo. WePS3 Evaluation Campaign: Overview of the On-line Reputation Management Task. In *2nd Web People Search Evaluation Workshop (WePS 2010), CLEF 2010 Conference*, 2010.

[3] E. Benson, A. Haghighi, and R. Barzilay. Event discovery in social media feeds. In *ACL '11*, 2011.

[4] D. Boyd, S. Golder, and G. Lotan. Tweet, tweet, retweet: Conversational aspects of retweeting on twitter. In *Hawaii Intern. Conf. on System Sciences*, 2010.

[5] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[6] M. Bron, B. Huurnink, and M. de Rijke. Linking archives using document enrichment and term selection. In *Research and Advanced Technology for Digital Libraries*. Springer Berlin / Heidelberg, 2011.

[7] C. J. C. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. *Journal of Machine Learning Research - Proceedings Track*, 14:25–35, 2011.

[8] K. W. Church and W. A. Gale. Inverse document frequency (IDF): A measure of deviations from poisson. In *Proc. Third Workshop on Very Large Corpora*, 1995.

[9] S. Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In *EMNLP '07*, 2007.

[10] S. Dill, N. Eiron, D. Gibson, D. Gruhl, R. Guha, A. Jhingran, T. Kanungo, S. Rajagopalan, A. Tomkins, J. Tomlin, and J. Zien. Semtag and seeker: Bootstrapping the semantic web via automated semantic annotation. In *WWW '03*, 2003.

[11] P. Ferragina and U. Scaiella. Tagme: on-the-fly annotation of short text fragments (by wikipedia entities). In *CIKM '10*, 2010.

[12] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[13] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2003.

[14] J. He, M. de Rijke, M. Sevenster, R. van Ommering, and Y. Qian. Generating links to background knowledge: A case study using narrative radiology reports. In *CIKM '11*, 2011.

[15] D. W. C. Huang, Y. Xu, A. Trotman, and S. Geva. Overview of INEX 2007 Link the Wiki Track. In *INEX '07*, 2007.

[16] J. Huang, K. M. Thornton, and E. N. Efthimiadis. Conversational tagging in twitter. In *HT '10*, 2010.

[17] G. Inches, M. J. Carman, and F. Crestani. Statistics of online user-generated short documents. In *ECIR '10*, 2010.

[18] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a social network or a news media? In *WWW '10*, 2010.

[19] D. Laniado and P. Mika. Making sense of twitter. In *ISWC '10*, 2010.

[20] X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. In *ACL: HLT '11*, 2011.

[21] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[22] K. Massoudi, E. Tsagkias, M. de Rijke, and W. Weerkamp. Incorporating query expansion and quality indicators in searching microblog posts. In *ECIR '11*, 2011.

[23] E. Meij, M. Bron, B. Huurnink, L. Hollink, and M. de Rijke. Learning semantic query suggestions. In *ISWC '09*, 2009.

[24] E. Meij, D. Trieschnigg, M. de Rijke, and W. Kraaij. Conceptual language models for domain-specific retrieval. *Inf. Process. Manage.*, 46(4):448–469, 2010.

[25] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. de Rijke. Mapping queries to the Linking Open Data cloud: A case study using DBpedia. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4):418 – 433, 2011.

[26] P. N. Mendes, A. Passant, P. Kapanipathi, and A. P. Sheth. Linked open social signals. In *WI-IAT '10*, 2010.

[27] R. Mihalcea and A. Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *CIKM '07*, 2007.

[28] D. Milne and I. H. Witten. Learning to link with Wikipedia. In *CIKM '08*, 2008.

[29] A. Mohan, Z. Chen, and K. Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. *Journal of Machine Learning Research - Proceedings Track*, 14:77–89, 2011.

[30] D. Nadeau and S. Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[31] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *ICWSM '10*, 2010.

[32] Pear Analytics. Twitter study – August 2009, 2009. `http://bit.ly/nYUJz7` [Online; accessed June 2011].

[33] J. A. Shaw and E. A. Fox. Combination of multiple searches. In *Text REtrieval Conference*, 1993.

[34] C. G. M. Snoek, B. Huurnink, L. Hollink, M. de Rijke, G. Schreiber, and M. Worring. Adding semantics to detectors for video retrieval. *IEEE Transactions on Multimedia*, 9(5):975–986, 2007.

[35] E. Tsagkias, M. de Rijke, and W. Weerkamp. Linking online news and social media. In *Fourth ACM Web Search and Data Mining (WSDM)*, Hong Kong, 2011.

[36] W. Weerkamp, S. Carter, and M. Tsagkias. How people use twitter in different languages. In *WebSci '11*, 2011.

[37] M. J. Welch, U. Schonfeld, D. He, and J. Cho. Topical semantics of twitter links. In *WSDM '11*, 2011.

[38] J. X. Yu, L. Qin, and L. Chang. Keyword search in relational databases: A survey. *IEEE Data Eng. Bull. Special Issue on Keyword Search*, 33(1):67–78, 2010.

[39] Y. Zhou and B. W. Croft. Query performance prediction in web search environments. In *SIGIR '07*, 2007.