Groundhog Day: Near-Duplicate Detection on Twitter

Ke Tao[1], Fabian Abel[1,2], Claudia Hauff[1], Geert-Jan Houben[1], Ujwal Gadiraju[1]
[1]TU Delft, Web Information Systems, PO Box 5031, 2600 GA Delft, the Netherlands
wis@st.ewi.tudelft.nl
[2]XING AG, Gänsemarkt 43, 20354 Hamburg, Germany
fabian.abel@xing.com

ABSTRACT

With more than 340 million messages that are posted on Twitter every day, the amount of duplicate content as well as the demand for appropriate duplicate detection mechanisms is increasing tremendously. Yet there exists little research that aims at detecting near-duplicate content on microblogging platforms. We investigate the problem of near-duplicate detection on Twitter and introduce a framework that analyzes the tweets by comparing (i) syntactical characteristics, (ii) semantic similarity, and (iii) contextual information. Our framework provides different duplicate detection strategies that, among others, make use of external Web resources which are referenced from microposts. Machine learning is exploited in order to learn patterns that help identifying duplicate content. We put our duplicate detection framework into practice by integrating it into Twinder, a search engine for Twitter streams. An in-depth analysis shows that it allows Twinder to diversify search results and improve the quality of Twitter search. We conduct extensive experiments in which we (1) evaluate the quality of different strategies for detecting duplicates, (2) analyze the impact of various features on duplicate detection, (3) investigate the quality of strategies that classify to what exact level two microposts can be considered as duplicates and (4) optimize the process of identifying duplicate content on Twitter. Our results prove that semantic features which are extracted by our framework can boost the performance of detecting duplicates.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Information filtering*; H.4.m [**Information Systems**]: Miscellaneous

Keywords

Duplicate Detection; Twitter; Diversification; Search

1. INTRODUCTION

On microblogging platforms such as Twitter or Sina Weibo, where the number of messages that are posted per second exceeds several thousands during big events[1], solving the problem of information overload and providing solutions that allow users to access new information efficiently are non-trivial research challenges. The majority of messages on microblogging platforms refer to news, e.g. on Twitter more than 85% of the tweets are news-related [9]. Many of the microposts convey the same information in slightly different forms which puts a burden on users of microblogging services when searching for new content. Teevan et al. [23] revealed that the search behaviour on Twitter differs considerably from the search behaviour that can be observed on regular Web search engines: Twitter users issue repetitively the same query and thus monitor whether there is new content that matches their query.

Traditional Web search engines apply techniques for detecting near-duplicate content [8, 12] and provide diversification mechanisms to maximize the chance of meeting the expectations of their users [19]. However, there exists little research that focuses on techniques for detecting near-duplicate content and diversifying search results on microblogging platforms. The conditions for inferring whether two microposts comprise highly similar information and can thus be considered near-duplicates differ from traditional Web settings. For example, the textual content is limited in length, people frequently use abbreviations or informal words instead of proper vocabulary and the amount of messages that are posted daily is at a different scale (more than 340 million tweets per day[2]).

In this paper, we bridge the gap and explore near-duplicate detection as well as search result diversification in the microblogging sphere. The main contributions of this paper can be summarized as follows[3].

- We conduct an analysis of duplicate content in Twitter search results and infer a model for categorizing different levels of duplicity.
- We develop a near-duplicate detection framework for microposts that provides functionality for analyzing (i) syntactical characteristics, (ii) semantic similarity and (iii) contextual information. The framework also exploits external Web content which is referenced by the microposts.
- Given our duplicate detection framework, we perform extensive evaluations and analyzes of different duplicate detection strategies on a large, standardized Twit-

[1] http://blog.twitter.com/2012/02/post-bowl-twitter-analysis.html

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

[2] http://blog.twitter.com/2012/03/twitter-turns-six.html
[3] We make our framework and datasets publicly available on our supporting website [22].

ter corpus to investigate the quality of (i) detecting duplicates and (ii) categorizing the duplicity level of two tweets.

- We integrate our duplicate detection framework into a Twitter search engine to enable search result diversification and analyze the impact of the diversification on the search quality.

2. RELATED WORK

Since Twitter was launched in 2006 it has attracted a lot of attention both from the general public and research communities. Researchers managed to find patterns in user behaviours on Twitter, including users' interests towards news articles [1], users' behaviour over time [10], and more general habits that users have on Twitter [18]. Previous research also studied the characteristics of emerging network structures [14] and showed that Twitter is rather a news media than a social platform [9]. Another area of interest is event detection in social Web streams [24], for example in the context of natural disasters [20].

The keyword-based search functionality is a generic tool for users to retrieve relevant information. Teevan et al. [23] analyzed the search behaviour on Twitter and found differences with respect to normal Web search. A first benchmark on Twitter data was introduced at TREC[4] 2011 with a track related to search in microblogs[5]. Among the solutions developed by participating researchers are many feature-driven approaches that exploit topic-insensitive features such as "does the tweet contain a URL?" to rank the tweets that match a given keyword query, e.g. [16]. More sophisticated search solutions also extract named entities from tweets in order to analyze the semantic meaning of tweets [21]. Bernstein et al. [5] investigated alternative topic-based browsing interfaces for Twitter while Abel et al. [2] investigated the utility of faceted search for retrieval on Twitter. However, none of the aforementioned research initiatives investigated strategies for near-duplicate detection and search result diversification in microblogs.

Traditional Web search engines benefit from duplicate detection algorithms and diversification strategies that make use of Broder et al.'s [6] shingling algorithm or Charikar's [7] random projection approach. Henzinger conducted a large-scale evaluation to compare these two methods [8] and Manku et al. [12] proposed to use the latter one for near-duplicate detection during Web crawling. To achieve diversification in search results, Agrawal et al. [3] studied the problem of search result diversification in the context of answering ambiguous Web queries and Rafiei et al. [19] suggested a solution to maximize expectations that users have towards the query results. However, to the best of our knowledge, the problem of identifying near-duplicate content has not been studied in the context of microblogs. In this paper, we thus aim to bridge the gap and research near-duplicate detection and search result diversification on Twitter.

3. DUPLICATE CONTENT ON TWITTER

In this section, we provide the outcomes of our study of duplicate content on the Twitter platform. We present a definition of near-duplicate tweets in 5 levels and show concrete examples. We then analyze near-duplicate content in a large Twitter corpus and investigate to what extent near-duplicate content appears in Twitter search results.

All our examples and experiments utilize the Twitter corpus which is provided by TREC [13].

3.1 Different levels of Near-Duplicate tweets

In this paper, we define duplicate tweets as tweets that convey the same information either syntactically or semantically. We distinguish near-duplicates in 5 levels.

Exact copy The duplicates at the level of *exact copy* are identical in terms of characters. An example tweet pair (t_1, t_2) in our Twitter corpus is:

t_1 *and* t_2*:* `Huge New Toyota Recall Includes 245,000 Lexus GS, IS Sedans - http://newzfor.me/?cuye`

Nearly exact copy The duplicates of *nearly exact copy* are identical in terms of characters except for *#hashtags*, *URLs*, or *@mentions*. Consider the following tweet:

t_3*:* `Huge New Toyota Recall Includes 245,000 Lexus GS, IS Sedans - http://bit.ly/ibUoJs`

Here, the tweet pair of (t_1, t_3) is a near-duplicate at a level of *nearly exact copy*.

Strong near-duplicate A pair of tweets is *strong near-duplicate* if both tweets contain the same core messages syntactically and semantically, but at least one of them contains more information in form of new statements or hard facts. For example, the tweet pair of (t_4, t_5) is strong near-duplicate:

t_4*:* `Toyota recalls 1.7 million vehicles for fuel leaks: Toyota's latest recalls are mostly in Japan, but they also... http://bit.ly/dHOPmw`

t_5*:* `Toyota Recalls 1.7 Million Vehicles For Fuel Leaks http://bit.ly/flWFWU`

Weak near-duplicate Two *weak near-duplicate* tweets either (i) contain the same core messages syntactically and semantically while personal opinions are also included in one or both of them, or (ii) convey semantically the same messages with differing information nuggets. For example, the tweet pair of (t_6, t_7) is a weak near-duplicate:

t_6*:* `The White Stripes broke up. Oh well.`

t_7*:* `The White Stripes broke up. That's a bummer for me.`

Low-overlapping The *low-overlapping* pairs of tweets semantically contain the same core message, but only have a couple of common words, e.g. the tweet pair of (t_8, t_9):

t_8*:* `Federal Judge rules Obamacare is unconsitutional...`

t_9*:* `Our man of the hour: Judge Vinson gave Obamacare its second unconstitutional ruling. http://fb.me/zQsChak9`

If a tweet pair does not match any of the above definitions, it is considered as *non-duplicate*.

3.2 Near-Duplicates in Twitter Search Results

In Section 3.1, the example tweets come from the Tweets 2011 corpus [13], which was used in the Microblog track of TREC 2011. The corpus is a representative sample from tweets posted during a period of 2 weeks (January 23rd to February 8th, 2011, inclusive). As the corpus is designed to be a reusable test collection for investigating Twitter search

[4] `http://trec.nist.gov`

[5] `http://sites.google.com/site/microblogtrack/`

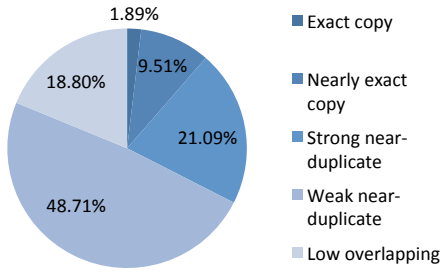

Figure 1: Ratios of near-duplicates in different levels

and ranking, it is used for the experiments of duplicate detection and search result diversification in the rest of the paper. The original corpus consists of 16 million tweets.

Besides the tweets, 49 topics (or queries) were provided for retrieval purposes. Moreover, TREC assessors judged the relevance between 40,855 topic-tweet pairs. A total of 2,825 topic-tweet pairs were judged as relevant. In other words, each topic on average has 57.65 relevant tweets. Employing Named Entity Recognition (NER) services on the content of these relevant tweets and the content of the 1,661 external resources referred by the links mentioned in them results in 6,995 and 56,801 entity extractions respectively when using DBpedia Spotlight [15], or 6,292 and 35,774 entity extractions respectively when using OpenCalais[6]. For each topic, we manually labelled all pairs of relevant tweets according to the levels of near-duplicates that we defined in in Section 3.1. In total, we labelled 55,362 tweet pairs. As a result, we found that 2,745 pairs of tweets are duplicate, 1.89% of them were labelled as exact copy and 48.71% of them were judged as weak near-duplicates (see Figure 1).

For each of the 49 topics, we ranked the tweets according to their relevance to the corresponding topic based on previous work [21] to investigate to what extent the ranked search results contain duplicate items. In the top 10, 20, 50 items and whole range of search results, we find that 19.4%, 22.2%, 22.5%, and 22.3% respectively are duplicates. Given one fifth of the items are duplicates, we consider duplicate detection an important step in the processing pipeline to diversify the search results.

4. DUPLICATE DETECTION FRAMEWORK

We consider the problem of duplicate detection as a classification task that can be performed in two steps: (i) deciding whether a pair of tweets are duplicates or not; and (ii) determining the duplicate level. For both steps, we rely on a collection of features that exploit syntactical elements, the semantics in both tweets and the content of referred Web pages, as well as context information about tweets and users. Finally, we employ logistic regression classifiers to ensemble the characteristics from pairs of tweets into the detection of duplicates and the determination of the levels.

4.1 Features of Tweet Pairs

We now provide an overview of the different features that we extract from tweet pairs for the task of duplicate detection. Given a pair of tweets (t_a, t_b), four sets of features are constructed. In the following sections, we elaborate on the definition of the features and the hypotheses that led us to include them in our strategies.

[6]`http://www.opencalais.com`

4.1.1 Syntactic Features

We construct syntactical features by matching the tweet pairs with respect to their overlap in letters, words, hashtags and URLs.

Levenshtein distance This feature indicates the number of characters required to change one tweet to the other. Each change can be a deletion, insertion, or substitution. Hence, Levenshtein distance evaluates the difference between a pair of tweets on the basis of differences in the usages of words, phrases, et cetera. As the furthest Levenshtein distance between a pair of tweets is $L_{max} = 140$ (the maximum length of a tweet), we normalize this feature by dividing the original value by L_{max}. Therefore, the final value of this feature is in the range of $[0, 1]$.
Hypothesis H1: The smaller the Levenshtein distance between a pair of tweets, the more likely they are duplicates and the higher the duplicate score.

Overlap in terms This feature compares tweet pairs by words. Although the tweets of near-duplicates use similar sets of words, the ordering of words may differ. Therefore we check the overlap in terms between tweet pairs. In our implementation, this feature is measured by using the Jaccard similarity coefficient as following:

$$overlap(w(t_a), w(t_b)) = \frac{|w(t_a) \cap w(t_b)|}{|w(t_a) \cup w(t_b)|} \quad (1)$$

Here, $w(t_a)$ and $w(t_b)$ are the sets of words that are used in t_a and t_b respectively. As we use the Jaccard similarity coefficient to measure the overlap, the value of this feature is in the range of $[0, 1]$. Similarly, the following features that describe overlap in different aspects are measured by the Jaccard similarity coefficient.
Hypothesis H2: The more overlap in terms we find between a pair of tweets, the higher the duplicate score.

Overlap in hashtags Hashtags are often used by users in tweets to get involved in the discussion about a topic, and also to make their voice easier to be found by others. This feature measures the overlap in hashtags between tweet pairs.
Hypothesis H3: The more common hashtags we find between a pair of tweets, the more likely they are duplicates and the higher the duplicate score.

Overlap in URLs Due to the length limitation of tweets, users often make use of URLs to give pointers to relevant detailed information. Hence we check the overlap of the links contained in the given pair of tweets. If a pair of tweets contain the same URL, they are probably about the same topic and are likely to be duplicates.
Hypothesis H4: The more overlap in URLs we find between a pair of tweets, the more likely they are duplicates and the higher the duplicate score.

Overlap in expanded URLs Various Twitter client applications and sharing functions used by news media sites shorten the URLs in order to give more space for real content [4]. As a result, we may miss some actual overlap in URLs if we only check original URLs. For this reason, we measure the overlap in expanded URLs between tweets. The expanded URLs can be obtained via the redirected locations given in the HTTP responses.
Hypothesis H5: The more common URLs we found between a pair of tweets after expanding the URLs, the more likely they are duplicates and the higher the duplicate score.

Length difference Besides matching letters, words, hashtags, and URLs, we also calculate the difference in length between two tweets and normalize it by L_{max}:

$$length\_difference = \frac{abs(|tweet_a| - |tweet_b|)}{140} \quad (2)$$

Hypothesis H6: The smaller the difference in length between two tweets, the higher the likelihood of them being duplicates and the higher their duplicate score.

4.1.2 Semantic Features

Apart from syntactical features of tweet pairs, semantic information may also be valuable for identifying duplicates, especially when the core messages or important entities in tweets are mentioned in different order. For this reason, we analyze the semantics in both tweets of a pair and construct features that may help with distinguishing duplicate tweets. We utilize NER services like DBpedia Spotlight, OpenCalais as well as the lexical database WordNet to extract the following features.

Overlap in entities Given extracted entities or concepts by employing NER services, we can check the overlap between the sets of entities in tweet pairs. The near-duplicate tweet pairs should contain the same core messages and therefore the same entities should be mentioned.
Hypothesis H7: The tweet pairs with more overlapping entities are more likely to have a high duplicate score.

Overlap in entity types For entities extracted from NER services, the types of the entities can also be retrieved. For example, if t_b contains the entities of type *person* and *location*, t_a should also contain the same type of entities to convey the core messages if they are a near-duplicate tweet pair. Otherwise, more types of entities may indicate it contains more information or less types may suggest only a partial coverage of the core message in t_a. Therefore, we construct features that measure the overlap in entity types between tweet pairs.
Hypothesis H8: The tweet pairs with more overlapping entity types are more likely to have a high duplicate score.

In fact, we found only a slight difference in performance between using DBpedia Spotlight and OpenCalais. In practice, we construct two features and derivative features (introduced later) by using *DBpedia Spotlight* because it yields slightly better results.

Overlap in topics Besides outputting entities with types, OpenCalais can classify the input textual snippets into 18 different categories a.k.a. topics. In this case, each tweet may be assigned more than one topic label or no topic at all. Therefore, it is possible to construct a feature by checking the overlap in topics.
Hypothesis H9: The tweet pairs that share more topics are more likely to have a high duplicate score.

Overlap in WordNet concepts We constructed this feature to compute the overlap based on lexical standards. To achieve this, we make use of the lexical database WordNet [17] to identify the nouns in pairs of tweets and calculate their overlap in these nouns. Practically, we use JWI (MIT Java Wordnet Interface)[7] to find the root concepts of the nouns in the tweets.
Hypothesis H10: The more overlap in WordNet noun concepts we find in a pair of tweets, the more likely they are to be duplicates and the higher their duplicate score.

[7] `http://projects.csail.mit.edu/jwi/`

Algorithm 1: WordNet similarity of a tweet pair

input : Tweet Pair (t_a, t_b)
output: WordNet similarity of Tweet Pair (t_a, t_b)

```
acc ← 0;
if |t_a| > |t_b| then
    swap(t_a, t_b);
foreach WordNet noun concept c_a in t_a do
    maximum ← 0;
    foreach WordNet noun concept c_b in t_b do
        if maximum < similarity_lin (c_a, c_b) then
            maximum ← similarity_lin (c_a, c_b);
    acc ←acc + maximum;
return acc / |W_a|;
```

Overlap in WordNet synset concepts Making use of merely WordNet noun concepts may not fully cover the overlap in information because different tweets may use different words or synonyms to convey the same information. In WordNet, synsets are interlinked by means of conceptual-semantic and lexical relations. We can make use of synsets to include all words with similar meaning for checking the overlap between tweet pairs.
Hypothesis H11: If the concepts in synsets are included for checking overlap between tweet pairs then the overlap feature may have a more positive correlation with the duplicate scores.

WordNet similarity There are several existing algorithms for calculating the semantic relatedness between WordNet concepts, e.g. the method proposed by Lin et al. [11] can measure the semantic relatedness between two concepts with a value between $[0, 1]$. The WordNet concepts are paired in order to get the highest relatedness. Practically, we follow Algorithm 1 to get this feature for a tweet pair (t_a, t_b). In the description of the algorithm, W_a stands for the set of WordNet noun concepts that appear in t_a.
Hypothesis H12: The higher the WordNet similarity of a tweet pair, the higher the likelihood of the tweets being duplicates and the higher their duplicate score.

4.1.3 Enriched Semantic Features

Due to the length limitation of tweets, 140 characters may not be enough to tell a complete story. Furthermore, some tweets, created by sharing buttons from other news sites for example, may even break the complete message. Thus, we make use of the external resources that are linked from the tweets. This step yields additional information and further enriches the tweets' semantics. Finally, we build a set of so-called *enriched* semantic features.

We construct six enriched semantic features, which are constructed in the same way as semantic features introduced in Section 4.1.2. The only difference is that the source of semantics contains not only the content of the tweets but also the content that we find by retrieving the content of the Web sites that are linked from the tweets.

4.1.4 Contextual Features

Besides analyzing syntactical and semantic aspects, which describe the characteristics of tweet pairs, we also evaluate the effects of the context in which the tweets were published on the duplicate detection. We investigate three types

of contextual features: temporal difference of the creation times, similarity of the tweets' authors, and the client application that the authors used.

Temporal difference For several popular events, e.g. UK Royal wedding, Japanese earthquake, and Super Bowl, users have posted thousands of tweets per second. During these events, breaking news are often retweeted not long after being posted. Therefore, it is reasonable to assume that the time difference between duplicate tweets is rather small. We normalize this feature by dividing the original value by the length of the temporal range of the dataset (two weeks in our setup).

Hypothesis H13: The smaller the difference in posting time between a pair of tweets, the higher the likelihood of it being a duplicate pair and the higher the duplicate score.

User similarity Similar users may publish similar content. We measure user similarity in a lightweight fashion by comparing the number of followers and the number of followees. Hence, we extract two features: the differences in *#followers* and *#followees* to measure the similarity of the authors of a post. As the absolute values of these two features vary in magnitude, we normalize this feature by applying log-scale and dividing by the largest difference in log-scale, which is 7 in our case.

Hypothesis H14: The higher the similarity of the authors of a pair of tweets, the more likely that the tweets are duplicates.

Same client This is a boolean feature to check whether the pair of tweets were posted via same client application. With authorization, third-party client applications can post tweets on behalf of users. Hence, different Twitter client applications as well as sharing buttons on various Web sites are being used. As the tweets that are posted from the same applications and Web sites may share similar content, provenance information and particularly information about the client application may be used as evidence for duplicate detection.

Hypothesis H15: The tweet pairs that are posted from the same client application tend to be near-duplicates.

4.2 Feature Analysis

As previously stated, we take the Twitter corpus released at TREC (*Tweets2011*) as our Twitter stream sample for the task of duplicate detection. Before we turn to (evaluating) duplicate detection strategies, we first perform an in-depth analysis of this sample with respect to the features that we presented in Section 4.1. We extracted these features for the 55,362 tweet pairs with duplicity judged (see Section 3.2). In Table 1, we list the average values and the standard deviations of the features and the percentages of *true* instances for the boolean feature respectively (*same client*). Moreover, Table 1 shows a comparison between features of duplicate (on all 5 levels) and non-duplicate tweet pairs.

Unsurprisingly, the Levenshtein distances of duplicate tweet pairs are on average 15% shorter than the ones of non-duplicate tweet pairs. Similarly, duplicate tweet pairs share more identical terms than non-duplicate ones: the duplicates have a Jaccard Similarity of 0.2148 in terms, whereas only 0.0571 for the non-duplicates. Hence, these two features which compare the tweets in letters and words may be potentially good indicators for duplicate detection. Although there is a difference in common hashtags between the duplicates and the non-duplicates, the overlap in hashtags does not seem to be a promising feature because of the low absolute value. This may be explained by the low usage of hashtags. The two features that check overlap in hyperlinks show similar characteristics but are slightly better. As expected, we discover more overlap in links by expanding the shortened URLs.

Tweet pairs may convey the same messages with syntactically different but semantically similar words. If this is the case then the syntactical features may fail to detect the duplicate tweets. Therefore, the features that are formulated as overlap in semantics are expected to be larger in absolute values than the syntactical overlap features. Overall, the statistics that are listed in Table 1 are in line with our expectations. We discover more overlap in the duplicates along 3 dimensions, including entities, entity types, and topics, by exploiting semantics with NER services. More distinguishable differences can be found in the features constructed from WordNet. The duplicate tweet pairs have more overlap in WordNet noun concepts or synsets (0.38) than the non-duplicate pairs (0.12). The feature of WordNet similarity is also potentially a good criterion for duplicate detection: the average similarity of duplicate pairs is 0.61 compared to 0.35 for non-duplicate pairs. The comparison of the enriched semantic features shows similar findings to those we observed for the semantic features. Again, the features that compare WordNet-based concepts are more likely to be good indicators for duplicate detection. However the WordNet similarity shows less difference if we consider external resources.

Finally, we attempted to detect the duplicates based on information about the context in which the tweets were posted. Hypothesis H13 (see Section 4.1.4) states that duplicates are more likely to be posted in a short temporal range. The result for the feature of temporal difference in Table 1 supports this hypothesis: the average value of this feature for the duplicate pairs is only 0.0256 (about 8 hours before normalization, see Section 4.1.4) in contrast to 0.2134 (about 3 days) for the non-duplicate ones. With respect to user similarity, we have not discovered an explicit difference between the two classes. Regarding the client applications from which duplicate tweets are posted, we observe the following: 21.1% of the duplicate pairs were posted from the same client applications whereas only 15.8% of the non-duplicate ones show the same characteristic.

4.3 Duplicate Detection Strategies

Having all the features constructed in Section 4.1 and preliminarily analyzed in Section 4.2, we now create different strategies for the task of duplicate detection. In practice, as requirements and limitations may vary in processing time, real-time demands, storage, network bandwidth et cetra, different strategies may be adopted. Given that our models for duplicate detection are derived from logistic regression, we define the following strategies by combining different sets of features, including one *Baseline strategy* and six *Twinder strategies*: *Sy* (only syntactical features), *SySe* (including tweet content-based features), *SyCo* (without semantics), *SySeCo* (without enriched semantics), *SySeEn* (without contextual features), and *SySeEnCo* (all features).

4.3.1 Baseline Strategy

As baseline strategy, Levenshtein distance, which compares tweet pairs in letters, is used to distinguish the duplicate pairs and further the duplicate levels.

Category	Feature	Duplicate	Std. deviation	Non-duplicate	Std. deviation
syntactical	Levenshtein Distance	0.5340	0.2151	0.6805	0.1255
	overlap in terms	0.2148	0.2403	0.0571	0.0606
	overlap in hashtags	0.0054	0.0672	0.0016	0.0337
	overlap in URLs	0.0315	0.1706	0.0002	0.0136
	overlap in expanded URLs	0.0768	0.2626	0.0017	0.0406
	length difference	0.1937	0.1656	0.2254	0.1794
semantics	overlap in entities	0.2291	0.3246	0.1093	0.1966
	overlap in entity types	0.5083	0.4122	0.3504	0.3624
	overlap in topics	0.1872	0.3354	0.0995	0.2309
	overlap in WordNet concepts	0.3808	0.2890	0.1257	0.1142
	overlap in WordNet Synset concepts	0.3876	0.2897	0.1218	0.1241
	WordNet similarity	0.6090	0.2977	0.3511	0.2111
enriched semantics	overlap in entities	0.1717	0.2864	0.0668	0.1230
	overlap in entity types	0.3181	0.3814	0.1727	0.2528
	overlap in topics	0.2768	0.3571	0.1785	0.2800
	overlap in WordNet concepts	0.2641	0.3249	0.0898	0.0987
	overlap in WordNet Synset concepts	0.2712	0.3258	0.0927	0.1046
	WordNet similarity	0.7550	0.2457	0.5963	0.2371
contextual	temporal difference	0.0256	0.0588	0.2134	0.2617
	difference in #followees	0.3975	0.1295	0.4037	0.1174
	difference in #followers	0.4350	0.1302	0.4427	0.1227
	same client	21.13%	40.83%	15.77%	36.45%

Table 1: The comparison of features between duplicate and non-duplicate tweets

4.3.2 Twinder Strategies

The Twinder strategies exploit the sets of features that have been introduced in Section 4.1). In our duplicate detection framework which is integrated in the Twinder search engine for Twitter streams (see Section 6), new strategies can easily be defined by grouping together different features.

Sy The *Sy* strategy is the most basic strategy in Twinder. It includes only *sy*ntactical features that compare tweets on a term level. These features can easily be extracted from the tweets and are expected to have a good performance on the duplicates for the levels of *Exact copy* or *Nearly exact copy*.

SySe This strategy makes use of the features that take the actual content of the tweets into account. Besides the *sy*ntactical features, this strategy makes use of NER services and WordNet to obtain the *se*mantic features.

SyCo The strategy of *SyCo* (without semantics) is formulated to prevent the retrieval of external resources as well as a large amount of semantics extractions that rely on either external Web services or extra computation time. Only *sy*ntactical features and *co*ntextual features are considered by this strategy.

SySeCo Duplicate detection can be configured as applying features without relying on external Web resources. We call the strategy that uses the *sy*tactical features, *se*mantics that are extracted from the content of tweets, and the *co*ntextual informaiton *SySeCo*.

SySeEn The contextual features, especially the ones related to users, may require extra storage and may be recomputed frequently. Therefore, the duplicate detection may work without contextual information by applying the so-called *SySeEn* (without contextual features).

SySeEnCo If enough hardware resources and network bandwidth are available then the strategy that integrates all the features can be applied so that the quality of the duplicate detection can be maximized.

5. EVALUATION OF DUPLICATE DETECTION STRATEGIES

To understand how different features and strategies influence the performance of duplicate detection, we formulated a number of research questions, which can be summarized as follows:

1. How accurately can the different *duplicate detection strategies* identify duplicates?
2. What kind of *features* are of particular *importance* for duplicate detection?
3. How does the accuracy vary for the *different levels* of duplicates?

5.1 Experimental Setup

We employ logistic regression for both steps of the task of duplicate detection: (i) to classify tweet pairs as duplicate or non-duplicate and (ii) to estimate the duplicate level. Due to the limited amount of duplicate pairs (of all 5 levels, 2,745 instances) in the manually labelled dataset (55,362 instances in total, see Section 3.2), we use 5-fold cross-validation to evaluate the learned classification models. At most, we used 22 features as predictor variables (see Table 1). Since the fraction of positive instances is considerably smaller than the negative one, we employed a cost-sensitive classification setup to prevent all the tweet pairs from being classified as non-duplicates. Moreover, because the precision and recall for non-duplicate are over 90%, we use the non-duplicate class as the reference class and focus on the performance of the class of duplicates. We use precision, recall, and F-measure to evaluate the results. Furthermore, since our final objective in this paper is to reduce duplicates in search results, we also point out the fraction of false positives as the indicator of the costs of losing information by applying our framework.

5.2 Influence of Strategies on Duplicate Detection

Table 2 shows the performance of predicting the duplicate tweet pairs by applying the strategies described in Section 4.3. The baseline strategy, which only uses Levensthein distance, leads to a precision and recall of 0.5068 and 0.1913 respectively. It means, for example, if 100 relevant tweets are returned for a certain search query and about 20 tweets (the example ratio of 20% according to the statistics given in Section 3.2) are duplicates that could be removed, the baseline strategy would identify 8 tweets as duplicates. However, only 4 of them are correctly classified while 16 other

Strategies	Precision	Recall	F-measure
Baseline	0.5068	0.1913	0.2777
Sy	0.5982	0.2918	0.3923
SyCo	0.5127	0.3370	0.4067
SySe	0.5333	0.3679	0.4354
SySeEn	0.5297	0.3767	0.4403
SySeCo	0.4816	0.4200	0.4487
SySeEnCo	0.4868	0.4299	0.4566

Table 2: Performance Results of duplicate detection for different sets of features

true duplicates are missed. In order to measure both precision and recall in once, the F-measure is used and for the *Baseline* strategy the value is 0.2777. In contrast, the *Sy* strategy, which is the most basic one for Twinder, leads to a much better performance in terms of all measures, e.g. an F-measure of 0.3923. By combing the contextual features, the *SyCo* strategy achieves a slightly better F-measure of 0.4067. It appears that the contextual features contribute relatively little to the performance.

Subsequently, we leave out the contextual features and check the importance of semantics in the content of the tweets and external resources. The *SySe* (including tweet content-based features) strategy considers not only the syntactical features but also the semantics extracted from the content of the tweets. We find that the semantic features can boost the classifier's effectiveness as the F-measure increased to 0.4354. The enriched semantics extracted from external resources brought little benefit to the result as the *SySeEn* strategy has a performance with F-measure of 0.4403. Overall, we conclude that semantics play an important role as they lead to a performance improvement with respect to F-measure from 0.3923 to 0.4403.

Thus the so-called *SySeCo* strategy excludes the features of enriched semantics but again includes the contextual features. Given this strategy, we observe an F-measure of 0.4487. However, if we adopt the strategy of *SySeEnCo* (all features), the highest F-measure can be achieved. At the same time, we nearly keep the same precision as with the *Baseline* strategy but boost the recall from 0.1913 to 0.4299. This means that more than an additional 20% of duplicates can be found while we keep the accuracy high. In this stage, we will further analyze the impact of the different features in detail as they are used in the strategy of *SySeEnCo*.

In the logistic regression approach, the importance of features can be investigated by considering the absolute value of the coefficients assigned to them. We have listed the details about the model derived for the *SySeEnCo* (all features) strategy in Table 3. The most important features are:

- *Levenshtein distance*: As it is a feature of negative coefficient in the classification model, we infer that a shorter Levenshtein distance indicates a higher probability of being duplicate pairs. Therefore, we confirm our Hypothesis H1 made in Section 4.1.1.
- *overlap in terms*: Another syntactical feature also plays an important role as the coefficient is ranked fourth most indicative in the model. This can be explained by the usage of common words in duplicate tweet pairs. This result supports Hypothesis H2.
- *overlap in WordNet concepts*: The coefficients of semantic and enriched semantic vary in the model. However, the most important feature is overlap in WordNet concepts. It has the largest positive weight which means that pairs of tweets with high overlap in WordNet concepts are more likely to be duplicates, confirming Hypothesis H10 (Section 4.1.2). However, we noticed a contradiction in the feature set of enriched semantics, in which the coefficient for overlap in WordNet concepts is negative (-2.0867) whereas the one the coefficient for the overlap in WordNet synset concept is positive (2.5496). It can be explained by the high correlation between these two features, especially for high coverage of possible words in external resources. For this reason, they counteract each other in the model.
- *temporal difference*: In line with the preliminary analysis, the shorter the temporal difference between a pair of tweets, the more likely that it is a duplicate pair. The highest value of the coefficient is partially due to low average absolute values of this feature. However, we can still conclude that Hypothesis H13 holds (see Section 4.1.4).

	Performance Measure	Score
	precision	0.4868
	recall	0.4299
	F-measure	0.4566
Category	**Feature**	**Coefficient**
syntactical	Levenshtein distance	<u>-2.9387</u>
	overlap in terms	<u>2.6769</u>
	overlap in hashtags	0.4450
	overlap in URLs	1.2648
	overlap in expanded URLs	0.8832
	length difference	1.2820
semantics	overlap in entities	-2.1404
	overlap in entity types	0.9624
	overlap in topics	1.4686
	overlap in WordNet concepts	<u>4.5225</u>
	overlap in WordNet Synset concepts	0.6279
	WordNet similarity	-0.8208
enriched semantics	overlap in entities	-0.8819
	overlap in entity types	0.9578
	overlap in topics	-0.1825
	overlap in WordNet concepts	-2.0867
	overlap in WordNet Synset concepts	<u>2.5496</u>
	WordNet similarity	0.7949
contextual	temporal difference	<u>-12.6370</u>
	difference in #followees	0.4504
	difference in #followers	-0.3757
	same client	-0.1150

Table 3: The coefficients of different features. The five features with the highest absolute coefficients are underlined.

Overall, we noticed that the hypotheses that we made for syntactical features can all be confirmed. Although the same conclusion could not be made for all the features constructed based on semantics, some of them can be explained. For example, the overlap of WordNet concepts in the set of enriched semantics is negative. The reason for this may be twofold: (i) more general terms (such as politics, sport, news, mobile) are overlapping if we consider external resources; (ii) the features in the set of enriched semantics may mislead when we extract the features for a pair of tweets from which no external resources can be found or only one tweet contains a URL. The situation for other features, e.g. WordNet similarity, can be explained by the dependencies between some of them. More specifically, the features that are based on WordNet similarity in the sets of semantics and enriched semantics may have positive correlation. Therefore, the coefficients complement each other in values. When we

consider only the contextual features, all other three features except the temporal difference do not belong to the most important features. More sophisticated techniques for measuring user similarity might be used to better exploit, for example, the provenance of tweets for the duplicate detection task.

5.3 Influence of Topic Characteristics on Duplicate Detection

In all reported experiments so far, we have considered the entire Twitter sample available to us. In this section, we investigate to what extent certain topic (or query) characteristics play a role for duplicate detection and to what extent those differences lead to a change in the logistic regression models.

Consider the following two topics: *Taco Bell filling lawsuit* (MB020[8]) and *Egyptian protesters attack museum* (MB010). While the former has a business theme and is likely to be mostly of interest to American users, the latter topic belongs into the category of politics and can be considered as being of global interest, as the entire world was watching the events in Egypt unfold. Due to these differences, we defined a number of topic splits. A manual annotator then decided for each split dimension into which category the topic should fall. We investigated four topic splits, three splits with two partitions each and one split with five partitions:

- Popular/unpopular: The topics were split into popular (interesting to many users) and unpopular (interesting to few users) topics. An example of a popular topic is *2022 FIFA soccer* (MB002) – in total we found 24. In contrast, topic *NIST computer security* (MB005) is classified as unpopular (as one of 25 topics).
- Global/local: In this split, we considered the interest for the topic across the globe. The already mentioned topic MB002 is of global interest, since soccer is a highly popular sport in many countries, whereas topic *Cuomo budget cuts* (MB019) is mostly of local interest to users living or working in New York where Andrew Cuomo is the governor. We found 18 topics of global and 31 topics of local interest.
- Persistent/occasional: This split is concerned with the interestingness of the topic over time. Some topics persist for a long time, such as MB002 (the FIFA world cup will be played in 2022), whereas other topics are only of short-term interest, e.g. *Keith Olbermann new job* (MB030). We assigned 28 topics to the persistent and 21 topics to the occasional topic partition.
- Topic themes: The topics were classified as belonging to one of five themes, either business, entertainment, sports, politics or technology. MB002 is, e.g., a sports topic while MB019 is considered to be a political topic.

Our discussion of the results focuses on two aspects: (i) the difference between the models derived for each of the two partitions, and (ii) the difference between these models (denoted $M_{splitName}$) and the model derived over all topics ($M_{allTopics}$) in Table 5. The results for the three binary topic splits are shown in Table 4.

Popularity: A comparison of the most important features of $M_{popular}$ and $M_{unpopular}$ shows few differences with the exception of a single feature: temporal difference. While temporal difference is the most important feature in $M_{popular}$, it is ranked fourth in $M_{unpopular}$. We hypothesize that the discussion on popular topics evolve quickly on Twitter, thus the duplicate tweet pairs should have less differences in posting time.

Global vs. local: The most important feature in M_{global}, the overlap in terms, and the second most important feature in M_{local}, Levenshtein distance, do not have a similar significance in each others' models. We consider it as an interesting finding and the possible explanation can lie in the sources of the information. In more detail, on the one hand the duplicate tweets about the local topics may share the same source thus are low in Levenshtein distances; on the other hand, different sources may report on the global topics in their own styles but with the same terms.

Temporal persistence: Comparing the $M_{persistent}$ and the $M_{occasional}$ models, yields to similar conclusions as in the previous two splits: (i) the persistent topics are continuously discussed so that the duplicate pairs are more likely to have short temporal differences, while the temporal differences between tweets on occasional topics are relatively insignificant; (ii) the occasionally discussed topics are often using the same set of words.

Topic Themes: The partial results of the topic split according to the theme of the topic are shown in Table 5 (full results can be found on the supporting website for this paper [22]). Three topics did not fit in one of the five categories. Since the topic set is split into five partitions, the size of some partitions is extremely small, making it difficult to reach conclusive results. Nevertheless, we can detect trends such as the fact that duplicate tweet pairs in sports topics are more likely to contain the same source links (positive coefficient of overlap in original URLs and the opposite of overlap in expanded URLs), while duplicate pairs in entertainment topics contain more shortened links (positive coefficient of overlap in expanded URLs). The overlap in terms has a large impact on all themes but politics. Another interesting observation is that a short temporal difference, is a prominent indicator for the duplicates in the topics of entertainment and politics but not in the other models.

The observation that certain topic splits lead to models that emphasize certain features also offers a natural way forward: if we are able to determine for each topic in advance to which theme or topic characteristic it belongs to, we can select the model that fits the topic best.

5.4 Analysis of Duplicate levels

Having estimated whether a tweet pair is duplicate or not, we now proceed to the second step of the duplicate detection task: determining the exact level of the duplicate tweet pairs. We compare the different strategies (see Section 4.3) in the same way as we have done in Section 5.2. To analyze the performance in general, we used weighted measures, including precision, recall, and F-measure across 5 levels. The results are summarized in Table 6; a similar pattern in performance improvement can be observed. However, it appears that the enriched semantics are more prominent than the contextual features as the so-called *SySeEn* strategy (without contextual features) performs better than *SySeCo* strategy (without enriched semantics).

In Figure 2, we plot the performance of the classification for 5 different levels and the weighted average of them with all the strategies that we have introduced in Section 4.3. The weight of each level depends on the ratio of duplicate instances. The curves for 5 different levels show the simi-

[8]The identifiers of the topics correspond to the ones used in the official TREC dataset.

Performance	Measure	popular	unpopular	global	local	persistent	occasional
	#topics	24	25	18	31	28	21
	#samples	32,635	22,727	19,862	35,500	33,474	21,888
	precision	0.4480	0.6756	0.6148	0.4617	0.4826	0.6129
	recall	0.4569	0.6436	0.5294	0.5059	0.5590	0.5041
	F-measure	0.4524	0.6592	0.5689	0.4828	0.5180	0.5532
Category	**Feature**	**popular**	**unpopular**	**global**	**local**	**persistent**	**occasional**
syntactical	Levenshtein distance	-3.4919	-0.6126	0.1342	-3.5136	-3.5916	-1.2338
	overlap in terms	2.8352	6.0905	6.7126	1.0498	1.3474	5.7705
	overlap in hashtags	0.6234	-2.5868	-1.8751	1.2671	2.2210	-2.7187
	overlap in URLs	-0.1865	5.8130	0.3275	1.6342	3.4323	0.4907
	overlap in expanded URLs	0.5180	2.7594	1.4933	0.6362	1.4936	1.0751
	length difference	1.2459	0.5974	0.5028	1.3236	1.5043	0.3793
semantics	overlap in entities	-2.3460	0.5430	-0.2263	-3.3525	-3.1071	0.5538
	overlap in entity types	1.2612	-1.1651	-0.3301	1.1571	0.8802	-0.8804
	overlap in topics	1.6607	0.7505	1.2147	1.2294	1.3911	0.7848
	overlap in WordNet concepts	5.5288	7.1115	5.8185	2.5319	4.0427	4.6365
	overlap in WordNet Synset concepts	-0.7763	-2.4393	-0.7335	3.0327	1.8752	0.5750
	WordNet similarity	-0.6254	1.8168	1.1355	-0.3909	-0.5141	1.0457
enriched semantics	overlap in entities	-1.3013	0.0583	0.0501	-0.5548	-1.9666	0.8555
	overlap in entity types	0.8997	1.2098	0.1179	0.8132	1.0279	0.3228
	overlap in topics	-0.5470	0.6581	0.0118	-0.1282	0.0292	-0.1884
	overlap in WordNet concepts	-1.8633	-1.2312	-2.3160	-2.4825	-2.5058	-2.1868
	overlap in WordNet Synset concepts	2.4274	1.0336	3.4218	2.6263	3.0461	2.5514
	WordNet similarity	0.7328	0.4342	-1.0359	0.9406	1.0470	-1.1867
contextual	temporal difference	-15.8249	-2.9890	-5.2712	-17.3894	-18.9433	-5.6780
	difference in #followees	0.7026	0.2322	-1.0797	0.5489	0.0659	-1.0473
	difference in #followers	-0.9826	1.1960	-0.3224	0.0048	-0.1281	-0.5178
	same client	-0.1800	0.3851	-0.0272	-0.0692	-0.2641	0.3058

Table 4: Influence comparison of different features among different topic partitions. There are three splits shown here: popular vs. unpopular topics, global vs. local topics and persistent vs. occasional topics. While the performance measures are based on 5-fold cross-validation, the derived feature weights for the logistic regression model were determined across all topics of a split. The total number of topics is 49. For each topic split, the three features with the highest absolute coefficient are underlined.

Performance	Measure	business	entertainment	sports	politics	technology
	#topics	6	12	5	21	2
	#samples	11,445	7,678	1,722	30,037	1,622
	precision	0.6865	0.6844	0.5000	0.4399	0.6383
	recall	0.6615	0.7153	0.6071	0.4713	0.7143
	F-measure	0.6737	0.6995	0.5484	0.4551	0.6742

Table 5: This table shows the comparison of performance when partitioning the topic set according to five broad topic themes. The full results can be found on the supporting website for this paper [22].

Strategies	Precision	Recall	F-measure
Baseline	0.5553	0.5208	0.5375
Sy	0.6599	0.5809	0.6179
SyCo	0.6747	0.5889	0.6289
SySe	0.6708	0.6151	0.6417
SySeEn	0.6694	0.6241	0.6460
SySeCo	0.6852	0.6198	0.6508
SySeEnCo	0.6739	0.6308	0.6516

Table 6: Performance Results of predicting duplicate levels for different sets of features

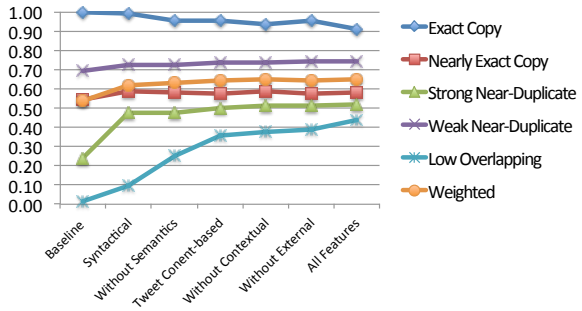

Figure 2: The F-measure of classification for different levels and weighted average by applying different strategies

lar trend with the weighted average of them that indicates the overall performance. We observe that the level of *Weak near duplicate* performs better than average and the reason can be attributed to the large ratio of learning instances (see Figure 1). The classification regarding the level of *Exact copy* is the best because of the decisive influence of the Levensthein distance. However, we see a declining trend in performance as we integrate more other tweets. Hence, further optimization is possible.

5.5 Optimization of Duplicate Detection

To optimize the duplicate detection procedure, we exploit the fact that duplicate pairs of level *Exact copy* can easily be detected by their Levenshtein distance of 0. After the removal of mentions, URLs, and hashtags, we can also apply the same rule for *Nearly exact copy*. Therefore, we can optimize the duplicate detection procedure with the following cascade:

1. If the Levensthein distance is zero between a pair of tweets or after removal of mentions, URLs, and hashtags from both of them, they can be classified as *Exact copy* or *Nearly exact copy*;

Strategies	Precision	Recall	F-measure
Baseline	0.9011	0.2856	0.4337
Sy	0.7065	0.4095	0.5185
SyCo	0.6220	0.4550	0.5256
SySe	0.6153	0.4849	0.5424
SySeEn	0.5612	0.5395	0.5501
SySeCo	0.6079	0.4914	0.5435
SySeEnCo	0.5656	0.5512	0.5583

Table 7: Performance Results of duplicate detection using different strategies after optimization

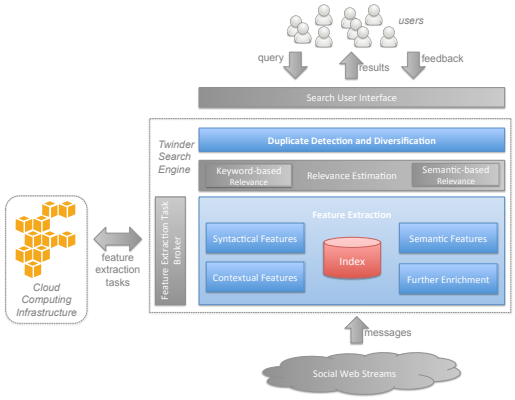

Figure 3: Architecture of the Twinder Search Engine: duplicate detection and diversification based on feature extraction modules.

2. Otherwise, we apply aforementioned strategies to detect duplicity.

After this optimization, we get a performance improvement from 0.45 to 0.55 with respect to the F-measure. The corresponding results are listed in Table 7 (the original results are given in Table 2).

6. SEARCH RESULT DIVERSIFICATION

A core application of near-duplicate detection strategies is the diversification of search results. Therefore, we integrated our duplicate detection framework into the so-called Twinder [21] (*Twitter Finder*) search engine which provides search and ranking functionality for Twitter streams. Figure 3 depicts the architecture of Twinder and highlights the core modules which we designed, developed and analyzed in the context of this paper.

The duplicate detection and diversification is performed after the relevance estimation of the tweets. Hence, given a search query, the engine first ranks the tweets according to their relevance and then iterates over the top k tweets of the search result to remove near-duplicate tweets and diversify the search results. Both, the duplicate detection and the relevance estimation module, benefit from the features that are extracted as part of the indexing step which is performed iteratively as soon as new tweets are monitored.

The lightweight diversification strategy applies the near-duplicate detection functionality as listed in Algorithm 2. It iterates from the top to the bottom of the top k search results. For each tweet i, it removes all tweets at rank j with $i < j$ (i.e. tweet i has a better rank than tweet j) that are near-duplicates of tweet i.

In Section 3.2, we analyzed the ratios of duplicates in the search results. After applying the lightweight diversification strategy proposed above, we again examine the ratios. The results are listed in Table 8 and reveal that the fraction

Algorithm 2: Diversification Strategy

```
input  : Ranking of tweets T, k
output: Diversified top k ranking T'@k
T'@k ← ∅;
i ← 0;
while i < k and i < T.length do
    j ← i+1 ;
    while j < T.length do
        if T[i] and T[j] are duplicates then
            remove T[j] from T;
        else
            j++;
    T'[i] = T[i]
return T'@k;
```

Range	Top 10	Top 20	Top 50	All
Before diversification	19.4%	22.2%	22.5%	22.3%
After diversification	9.1%	10.5%	12.0%	12.1%
Improvement	53.1%	52.0%	46.7%	45.7%

Table 8: Average ratios of near-duplicates in search results after diversification

of near-duplicate tweets within the top k search results is considerably smaller. For example, without diversification there exists, on average, for 22.2% of the tweets at least one near-duplicate tweet within the top 20 search results. In contrast, the diversification strategy improves the search result quality with respect to duplicate content by more than 50%. Thus there are, on average, less than 11% duplicates in the top 20 search results.

7. CONCLUSIONS

People are confronted with a high fraction of near-duplicate content when searching and exploring information on microblogging platforms such as Twitter. In this paper, we analyzed the problem of near-duplicate content on Twitter and developed a duplicate detection and search result diversification framework for Twitter. Our framework is able to identify near-duplicate tweets with a precision and recall of 48% and 43% respectively by combining (i) syntactical features, (ii) semantic features and (iii) contextual features and by considering information from external Web resources that are linked from the microposts. For certain types of topics such as occasional news events, we even observe performances of more than 61% and 50% with respect to precision and recall. Our experiments show that semantic features such as the overlap of WordNet concepts are of particular importance for detecting near-duplicates. By analyzing a large Twitter sample, we also identified five main levels of duplicity ranging from *exact copies* which can easily be identified by means of syntactic features such as string similarity to *low overlapping duplicates* for which an analysis of the semantics and context is specifically important. Our framework is able to classify the duplicity score on that level with an accuracy of more than 60%. Given our near-duplicate detection strategies, we additionally developed functionality for the diversification of search results. We integrated this functionality into the Twinder search engine and could show that our duplicate detection and diversification framework improves the quality of top k retrieval significantly since we decrease the fraction of duplicate content that is delivered to the users by more than 45%.

Acknowledgements. This work is co-funded by the EU FP7 project ImREAL (`http://imreal-project.eu`).

8. REFERENCES

[1] F. Abel, Q. Gao, G.-J. Houben, and K. Tao. Analyzing User Modeling on Twitter for Personalized News Recommendations. In *Proceedings of the 19th International Conference on User Modeling, Adaption and Personalization (UMAP)*, pages 1–12. Springer, July 2011.

[2] F. Abel, C. Hauff, G.-J. Houben, R. Stronkman, and K. Tao. Twitcident: Fighting Fire with Information from Social Web Stream. In *Proceedings of the 21st International Conference Companion on World Wide Web (WWW)*, pages 305–308. ACM, April 2012.

[3] R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying Search Results. In *Proceedings of the 2nd ACM International Conference on Web Search and Data Mining (WSDM)*, pages 5–14. ACM, February 2009.

[4] D. Antoniades, I. Polakis, G. Kontaxis, E. Athanasopoulos, S. Ioannidis, E. P. Markatos, and T. Karagiannis. we.b: The web of Short URLs. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 715–724. ACM, April 2011.

[5] M. S. Bernstein, B. Suh, L. Hong, J. Chen, S. Kairam, and E. H. Chi. Eddi: Interactive Topic-based Browsing of Social Status Streams. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 303–312. ACM, October 2010.

[6] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic Clustering of the Web. In *Selected Papers from the 6th International Conference on World Wide Web (WWW)*, pages 1157–1166. Elsevier Science Publishers Ltd., September 1997.

[7] M. S. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC)*, pages 380–388. ACM, May 2002.

[8] M. Henzinger. Finding Near-Duplicate Web Pages: A Large-Scale Evaluation of Algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 284–291. ACM, August 2006.

[9] H. Kwak, C. Lee, H. Park, and S. Moon. What is Twitter, a Social Network or a News Media? In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 591–600. ACM, April 2010.

[10] C. Lee, H. Kwak, H. Park, and S. Moon. Finding Influentials Based on the Temporal Order of Information Adoption in Twitter. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 1137–1138. ACM, April 2010.

[11] D. Lin. An Information-Theoretic Definition of Similarity. In *Proceedings of the 15th International Conference on Machine Learning (ICML)*, pages 296–304. Morgan Kaufmann, July 1998.

[12] G. S. Manku, A. Jain, and A. Das Sarma. Detecting Near-Duplicates for Web Crawling. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, pages 141–150. ACM, May 2007.

[13] R. McCreadie, I. Soboroff, J. Lin, C. Macdonald, I. Ounis, and D. McCullough. On Building a Reusable Twitter Corpus. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1113–1114. ACM, August 2012.

[14] B. Meeder, B. Karrer, A. Sayedi, R. Ravi, C. Borgs, and J. Chayes. We Know Who You Followed Last Summer: Inferring Social Link Creation Times In Twitter. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, pages 517–526. ACM, April 2011.

[15] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. DBpedia Spotlight: Shedding Light on the Web of Documents. In *Proceedings of the 7th International Conference on Semantic Systems (I-SEMANTICS)*, pages 1–8. ACM, September 2011.

[16] D. Metzler and C. Cai. USC/ISI at TREC 2011: Microblog Track. In *Working Notes, The Twentieth Text REtrieval Conference (TREC 2011) Proceedings*. NIST, November 2011.

[17] G. Miller et al. WordNet: A Lexical Database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

[18] M. Pennacchiotti and A.-M. Popescu. A Machine Learning Approach to Twitter User Classification. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 281–288. AAAI Press, July 2011.

[19] D. Rafiei, K. Bharat, and A. Shukla. Diversifying Web Search Results. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 781–790. ACM, April 2010.

[20] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 851–860. ACM, April 2010.

[21] K. Tao, F. Abel, C. Hauff, and G.-J. Houben. Twinder: A Search Engine for Twitter Streams. In *Proceedings of the 12th International Conference on Web Engineering (ICWE)*, pages 153–168. Springer, July 2012.

[22] K. Tao, F. Abel, C. Hauff, G.-J. Houben, and U. Gadiraju. Supporting Website: datasets and additional findings., November 2012. `http://wis.ewi.tudelft.nl/duptweet/`.

[23] J. Teevan, D. Ramage, and M. R. Morris. #TwitterSearch: A Comparison of Microblog Search and Web Search. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining (WSDM)*, pages 35–44. ACM, February 2011.

[24] J. Weng and B.-S. Lee. Event Detection in Twitter. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*. AAAI Press, July 2011.