Named Entity Normalization in User Generated Content

Valentin Jijkoun jijkoun@science.uva.nl
Mahboob Alam Khalid mahboob@science.uva.nl
Maarten Marx marx@science.uva.nl
Maarten de Rijke mdr@science.uva.nl

ISLA, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam

ABSTRACT

Named entity recognition is important for semantically oriented retrieval tasks, such as question answering, entity retrieval, biomedical retrieval, trend detection, and event and entity tracking. In many of these tasks it is important to be able to accurately *normalize* the recognized entities, i.e., to map surface forms to unambiguous references to real world entities. Within the context of structured databases, this task (known as record linkage and data de-duplication) has been a topic of active research for more than five decades. For edited content, such as news articles, the named entity normalization (NEN) task is one that has recently attracted considerable attention. We consider the task in the challenging context of user generated content (UGC), where it forms a key ingredient of tracking and media-analysis systems.

A baseline NEN system from the literature (that normalizes surface forms to Wikipedia pages) performs considerably worse on UGC than on edited news: accuracy drops from 80% to 65% for a Dutch language data set and from 94% to 77% for English. We identify several sources of errors: entity recognition errors, multiple ways of referring to the same entity and ambiguous references.

To address these issues we propose five improvements to the baseline NEN algorithm, to arrive at a language independent NEN system that achieves overall accuracy scores of 90% on the English data set and 89% on the Dutch data set. We show that each of the improvements contributes to the overall score of our improved NEN algorithm, and conclude with an error analysis on both Dutch and English language UGC. The NEN system is computationally efficient and runs with very modest computational requirements.

Categories and Subject Descriptors

H.3.1 [**Information Systems**]: Information Storage and Retrieval—*Content Analysis and Indexing*

AND'08, July 24, 2008, Singapore.

General Terms

Algorithms, Experimentation, Evaluation

Keywords

Named entities, Wikipedia, User generated content

1. INTRODUCTION

The task of record linkage (RL) is to find entries that refer to the same entity in different data sources. This task has been investigated since the 1950s—usually, entries are considered with their attributes (e.g., person with phone, address) [21]. The task proved important because data sources have varying ways of referring to the same real-world entity due to, e.g., different naming conventions, misspellings or use of abbreviation. The task of reference normalization is to analyze and detect these different references [6, 7]. When we consider the special case of this problem for natural language texts, we have to recognize entities in a text and resolve these references either to entities that exist within the document or to real-world entities. These two steps constitute the *named entity normalization* (NEN) problem.

We consider the NEN task within the setting of user generated content (UGC), such as blogs, discussion forums, or comments left behind by readers of online documents. For this type of textual data, the NEN task is particularly important within the settings of media and reputation analysis (which motivated the work reported here) and of intelligence gathering. Many strategies deployed in these areas revolve around the idea of determining and tracking the impact of an event, i.e., determining the number, intensity and orientation of responses and identifying the stakeholders and other actors and entities involved.

The specific scenario on which we focus concerns the analysis of data that is increasingly common: online texts decorated with unedited comments left behind by web users. Examples include news sites (such as BBC news), and discussion and collaboration forums (such as `linuxforum.com`). These comments contain valuable information that complements the original text that triggered them, but the sheer volume and their (usually) flat organization makes them hard to comprehend. Hence, tools are needed that help organize the list of comments, by clustering them, summarizing them, computing aggregate information, creating hyperlinks between them, etc.

Let's consider an example. In the data set that we use for evaluation purposes in this paper (see Section 4 for details), one of the news stories is about racing driver Michael

Figure 1: An excerpt from a BBC news article, with the excerpts of three comments (out of total of 39). Named entities are underlined.

- *News item*: Michael Schumacher wins his sixth victory in eight races—and tightens his grip on another Championship title. Do you think the title race is over? Have Your Say. Michael Schumacher extended his lead to 43 points after Juan Pablo Montoya's Williams broke down with 12 laps to go. (...)
- *Comments*:
  1. Ferrrari and Schumacher are now beyond the point where anyone can stop them (...)
  2. (...) Ralf, Montoya or DC need to win all the remaining seven races without Michael getting any points (...)
  3. (...) Both Williams' drivers could be giving Schumi more of a challenge if their cars were reliable, as could Coulthard at McLaren (...)

Schumacher (Figure 1). To normalize the named entities in this news item and the comments it triggers, we need to resolve them to real world entities. We notice that there are two types of reference in the data. One is *within-document reference*, e.g., in the comments in Figure 1, *Michael* and *Schumi* are used to refer to *Michael Schumacher* as mentioned earlier. The second kind of references are *references to real-world entities*, e.g., in the second comment, *DC* is used to refer to *David Coulthard*. Notice that resolving references of the latter type involves *named entity disambiguation*: in the context of Figure 1, *DC* is not used to refer to "Daimler-Chrysler," "direct current" or the number 600.[1]

The main challenge in normalizing named entities (NEs) occurring in the comments on a news story is that commentators often do not use the full name of an already mentioned NE, use nicknames, misspell words or creatively pun with them. For example, in one of the examples in our Dutch data set (a news article with 90 comments), singer *Anneke Grönloh* is referred to in 11 different ways, including variants such as *Anneke Grohnloh*, *anneke gr ?hnloh*, *Mw. Gronloh*, *Anneke Kreunlo*, *Mevrouw G.*, etc. Other examples of creative language use include *G@@Gle* and *Bu$h*. Besides, commentators often introduce additional NEs not even mentioned in the triggering news story, and some of the NEs used may actually refer to earlier comments. All in all, this turns NEN on UGC into a challenging problem.

NEN has been considered before, on structured data and on edited content. Of particular relevance to us is recent work by Cucerzan [4], comparing methods for NEN on edited content. In the present paper we apply similar methods to user generated content. We find several main sources of errors: NE recognition errors (incorrect boundaries of named entities as well as missing NEs), multiple ways of referring to the same entity, and ambiguous (out of context) references. We present five improvements to the baseline NEN algorithm to address these error types, namely: trimming, joining and ngramming NEs, approximate name matching, identification of missing references and name disambiguation. We assess the overall performance of the improved system as well as the individual contributions of the improvements. In this paper, we aim to create a named entity normalization algorithm for use in Dutch/English language media and reputation analysis settings that performs well on user generated content (UGC). We use Wikipedia, the largest encyclopedia to date, to assign unique identifiers to real world entities in the entity normalization process. For NEs not found in Wikipedia, we use the most complete variant of the name found in the text as the identifier.

The main contributions of the paper are: presentation and analysis of the problem of NEN in UGC, an algorithm for addressing the problem, and evaluation and analysis of the algorithm. Our algorithm was developed for Dutch and using Dutch data, but experiments with an English data set indicate that it is well applicable to other languages. Moreover, the algorithm is computationally efficient.

The remainder of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we present a baseline algorithm for named entity normalization in user generated content, and describe an improved version based on an error analysis. In Section 4 we present our experimental setup and in Section 5 we present and analyze the results of our evaluation. A set of conclusions in Section 6 completes the paper.

2. RELATED WORK

Name matching and disambiguation has been recognized as an important problem in various domains. Borgman and Siegfried [2] present an overview of motivations, applications, common problems and techniques for name matching in the art domain; see [17] for recent experiments with classification for name matching. A dual problem, personal name disambiguation has also attracted a lot of attention, and a number of unsupervised methods were proposed [14, 18].

A similar problem has been known in the database community for over five decades as the *record linkage* or the *record matching* problem [8, 21]. However, there the task is more general: matching arbitrary types of records, not just person names or other types of named entities. Another type of research focuses on identification, disambiguation and matching of text objects other than named entities, specifically, temporal expressions [1].

Related problems occur in a different task: discovering links in text. Like NEN, this task involves identifying and disambiguating references to entities, and has also attracted attention of the research community [10, 15].

Research on named entity extraction and normalization has been carried out in both restricted and open domains. For example, for the case of scientific articles on genomics, where gene and protein names can be both synonymous and ambiguous, Cohen [3] normalizes entities using dictionaries automatically extracted from gene databases. For the news domain, Magdy et al. [13] address cross-document Arabic person name normalization using a machine learning approach, a dictionary of person names and frequency information for names in a collection. Cucerzan [4] considers the entity normalization task for news and encyclopedia articles; they use information extracted from Wikipedia combined with machine learning for context-aware name disambiguation; the baseline that we use in this paper (taken from [11]) is a modification (and improved version) of Cucerzan [4]'s baseline. Cucerzan [4] also presents an extensive literature overview on the problem.

[1] See `http://en.wikipedia.org/wiki/DC`

Recent research has also examined the impact of normalizing entities in text on specific information access tasks. Zhou et al. [22] show that appropriate use of domain-specific knowledge base (i.e., synonyms, hypernyms, etc.) yields significant improvement in passage retrieval in the biomedical domain. Similarly, Khalid et al. [11] demonstrate that NEN based on Wikipedia helps text retrieval in the context of Question Answering in the news domain.

Finally, in recent years, there has been a steady increase in the development or adaptation of language technology for UGC. Most of the attention has gone to blogs (see [16] for a recent survey on text analytics for blogs). Online discussion fora are more closely related to the data with which we work; recent research includes work on finding authoritative answers in forum threads [9, 12], as well as attempts to assess the quality of forum posts [20]. To the best of our knowledge, discussion threads as triggered by news stories of the kind considered here have not been studied before.

3. AN ALGORITHM FOR NAMED ENTITY NORMALIZATION IN USER GENERATED CONTENT

In this section we present a baseline algorithm for NE-normalization based on [4, 11], perform an error analysis and describe five improvements to the baseline, each accounting for a specific type of error identified.

3.1 Baseline algorithm

Algorithm 1, our baseline algorithm, takes as input a pair $\langle A, R\rangle$ where A is the triggering news article and R is the list of comments on A in reverse chronological order. It returns an *entity model*, i.e., a list of triples $\langle s, n, p\rangle$, where s is a surface form (i.e., a named entity as it occurs in text), n is the normalized form of s (e.g., a title of the corresponding Wikipedia article), and p is the character position of s in the document. For example, one of the entity triples from the text in Figure 1 is $\langle$*Schumi*, *Michael Schumacher*, 57$\rangle$.

Line 1 of Algorithm 1 performs the NE recognition, i.e., it identifies NEs of types PERSON, LOCATION, ORGANIZATION or MISC (miscellaneous). Lines 2 and 4 do the preprocessing: we remove all noisy NEs, i.e., short (length $\leq$ 2 characters) or stopword-only NEs, with the exception of (capitalized) abbreviations, and remove diacritics (e.g., replacing *ö* with *o*).

Next, on line 5 we normalize each found NE using the function shown as Algorithm 2. This normalization algorithm treats NEs that are person names differently from other NE types (lines 2 and 3). Specifically, for persons we further remove common titles (such as *Mr*, *Mrs*) and perform *within-document reference resolution*, as detailed in Algorithm 3.

Algorithm 2 continues (line 5) by trying to link the NE to a Wikipedia article, calling the function `findWikiEntity` shown in Algorithm 4. If even after this step the NE is not normalized, we take the string itself as its normalized form (lines 6–7 of Algorithm 1).

The function `ResolveRefInDoc`, described in Algorithm 3, examines the list of entities already found and normalized earlier in the document, and finds matches based on first or last names.

The function `findWikiEntity`, described in Algorithm 4, first tries to match the input reference string with a Wikipedia article title (either exact match or case-insensitive, in this order).[2] If we find a matched Wikipedia page title, `WT`, we check whether the page is a Wikipedia redirection page (line 2). In case of a redirect, we take the title of the target Wikipedia page instead. Then we check if `WT` refers to a Wikipedia disambiguation page,[3] i.e., it lists a number of possible candidate pages for a given term. If this is the case, we select one of them, disambiguating between candidates using a heuristic from [11]: we select the candidate that has the highest number of incoming links in Wikipedia.

Algorithm 1 Compute the entity model of a document

Require: a text document DOC
1: $\text{REFS}_{doc} \Leftarrow$ NE-Recognition(DOC)
   {REFS_{doc}: list of $\langle$NE, type, position$\rangle$ triples}
2: $\text{REFS}_{doc} \Leftarrow$ Remove-NoinsyNEs(REFS_{doc})
3: **for** each $\langle$NE, type, position$\rangle \in \text{REFS}_{doc}$ **do**
4: REF $\Leftarrow$ Removing diacritics(NE)
5: REF-norm $\Leftarrow$ NormalizeNE($\langle$REF, type$\rangle$, REF-NORMS_{doc}) {see Algorithm 2}
6: **if** REF-norm = NULL **then**
7: REF-norm $\Leftarrow$ NE
8: **end if**
9: $\text{REF-NORMS}_{doc} \Leftarrow \langle$REF, REF-norm, position$\rangle$
10: **end for**
11: **return** REF-NORMS_{doc}

Algorithm 2 NormalizeNE: named entity normalization

Require: a pair $\langle$NAME, Type$\rangle$, a list REF-NORMS
1: **if** Type = PERSON **then**
2: NAME $\Leftarrow$ RemoveTitles(NAME)
3: REF-norm $\Leftarrow$ ResolveRef-InDoc(NAME, REF-NORMS) {see Algorithm 3}
4: **end if**
5: **if** REF-norm = NULL **then**
6: REF-norm $\Leftarrow$ findWikiEntity(REF) {see Algorithm 4}
7: **end if**
8: **return** REF-norm

3.2 Error analysis of the baseline algorithm

For development purposes, we annotated one news story (about two personalities in Dutch show business, Anneke Grönloh and Paul de Leeuw) and the first 90 accompanying comments.[4] We found 166 entities in the development data. Then, we compared the performance of our baseline algorithm against this gold standard, and performed an error analysis. Below we list the most common types of errors.

Recognition errors The boundaries of some NEs are recognized incorrectly: NEs include noise or are split in the middle, e.g., <ne>*Gronloh!!!*</ne>, and <ne>*Paul*</ne> *de* <ne>*Leeuw*</ne>. Commentators use

[2]The order matters, e.g., "MAC" and "Mac" are two different Wiki entities, the former is an abbreviation and the latter may refer to Mac OS X.

[3]I.e., a page that uses the Wikipedia Disambiguation template, or whose title ends in "_(disambiguation)" or "_(surname)."

[4]This development document was not included in the test set described in Section 4 below.

Algorithm 3 ResolveRef-InDoc: within-document coreference resolution

Require: a string NAME, a list REF-NORMS
{REF-NORMS: list of ⟨REF, REF-norm, position⟩ triples}
1: **for** each ⟨REF, REF-norm, position⟩ ∈ REF-NORMS **do**
2: **if** NAME = REF or NAME = REF-norm **then**
3: **return** ⟨REF, REF-norm, position⟩
4: **else if** NAME = firstName(REF) or NAME = lastName(REF) or NAME = firstName(REF-norm) or NAME = lastName(REF-norm) **then**
5: {firstName is the first token of the input string, and the lastName is the rest}
6: **return** ⟨REF, REF-norm, position⟩
7: **end if**
8: **end for**

Algorithm 4 findWikiEntity

Require: a string NAME
1: WT ⇐ findWikiTitle(NAME) {If NAME matches with a Wikipedia page's title}
2: **if** isRedirectPage(WT) **then**
3: WT ⇐ getTargetPage(WT)
4: **end if**
5: **if** isDisambiguationPage(WT) **then**
6: **return** Disambiguate(WT)
7: **end if**
8: **return** WT

capitalization and punctuation in non-standard ways, confusing the NE recognizer, which produces results like <ne>*Gronloh ,Maak*</ne> or <ne>*Gronloh .Die* </ne>. We found 27 such errors in the development data.

Multi-references Commentators sometimes do not bother with punctuation, and the NER recognizes a group of adjacent names as a single NE. For example, <ne> *Anneke Gronloh Paul de Leeuw*</ne>, actually contains references to two persons. The development data contains only one such error.

Variants Commentators often use partial (first or last) names or nicknames of the entities mentioned in the trigger article. Users also misspell names in comments or use creative puns. The development data contains 14 variants of 4 entities.

Missing NEs Often, capitalization is absent in Dutch UGC. Our NE recognizer misses all those variants like *anneke*, *gronloh*, *paul* and *anneke gr"ohnloh*. In the development data, 16 occurrences of entities were not detected.

Incomplete entities Commentators introduce new named entities without using full names, e.g., *Pronk* (refers to Dutch ex-minister Jan Pronk). Finding full names of such entities is difficult because we don't know what is in the mind of the commentator. In the development data, only one incomplete entity was found.

3.3 Improving named entity normalization

Based on the types of errors just listed we suggest five improvements of our baseline algorithm:

1. *Pre-processing NEs*: We clean the entities and fix some of the NE boundary recognition errors. Specifically, we perform the following two steps before calling the function `NormalizeNE`.
   - Clean up: remove non-alphabetical characters from both sides of an NE;
   - Gluing NEs: (Dutch-specific) to improve recognition of Dutch multi-word last names such as "van Gogh" or "ten Cate," we glue two adjacent person names if they are separated by one of the standard Dutch infixes (*van, ten, de, van der, van ten, van t, vt*).
2. *N-gram NE normalization*: we use ngramming to split NEs that cannot be normalized. This is a modification of part of the baseline algorithm (lines 6–8 of Algorithm 1). Specifically, if an NE cannot be normalized, we split the NE into a set of all overlapping word $n$grams (all possible values of n) and try to normalize each ngram using the function `NormalizeNE`. We proceed from longer to shorter ngrams, and when one of the ngrams is normalized successfully, all other ngrams that overlap with it are ignored.
3. *Person-name matching*: we improve our within-document coreference resolution (Algorithm 3), handling common variants of person names as they are used in UGC. Specifically, we use a set of heuristics (based on first and last names) and inexact string matching (based on Levenshtein edit distance) to identify variants of person names. This is a modification of lines 4–6 of Algorithm 3. The details of the improved name matching algorithm are given in the Appendix.
4. *Finding missing NEs*: we add a post-processing step to the baseline algorithm, in order to improve the recall of NEs. Specifically, after the recognition and normalization has been done (line 10 of Algorithm 1), we look for phrases which have not been recognized as NEs, but which are similar to those already recognized and normalized. We consider text segments outside the recognized NEs, and use a procedure similar to improvement 2 above: we split the text segments into word ngrams and try to resolve each ngram within the documents (calling the function `ResolveRef-InDoc`, Algorithm 3). Again, we proceed from longer to shorter ngrams, and ignore ngrams overlapping with those we detect as NEs. For efficiency, in this step we only consider ngrams that contain at most $n+2$ tokens, where n is the maximal number of tokens of a normalized form of NEs in the document.
5. *Normalizing new entities:* Wikipedia disambiguation pages do not cover all possible ambiguous entities, e.g., incomplete entity references such as "Pronk" (a Dutch surname). As noted in [4, 11], anchor texts in Wikipedia often cover these cases. Following [11], if an NE exactly matches the anchor text of some hyperlink in Wikipedia, we take as its normalized form the title of the Wikipedia article (the target of the hyperlink) with the highest number of incoming links. This is a modification of the function `Disambiguate` of Algorithm 4.

In the following sections we describe experiments used to test the effect of the improvements listed above.

4. EXPERIMENTAL SETUP

4.1 Research questions

In Section 3 we have described the baseline NEN algorithm and five improvements. Now we turn to set up our experiments in order to answer the following research questions:

RQ1 How does the baseline NEN algorithm from the literature perform on UGC?

RQ2 What is the overall performance achieved by the combined improvements proposed in Section 3.3?

RQ3 How important are the proposed improvements, individually? That is, if we leave out one of them, how much do we lose in terms of effectiveness?

RQ4 To what extent is our NEN algorithm work language dependent? I.e., does it achieve comparable scores across languages? We do not have the resources to answer this question for all possible language pairs, but compare the performance of our NEN algorithm on two related languages: Dutch and English.

While our main interest lies with the effectiveness of our NEN algorithm, we also include some results on efficiency.

4.2 Test data

As there is no standard data set available for assessing NEN on user generated comments, we decided to create our own data set. Given our application scenario (Dutch language media and reputation analysis), we selected five articles with comments from online versions of two leading Dutch national newspapers.[5] To be able to answer RQ4, we added five BBC[6] articles with comments from the "Have your say" section. For both languages, we selected articles in such a way as to cover the main named entity types (persons, locations, organizations and miscellaneous), and a variety of spelling variants of names. We expected that commentators vary more with names of people from show business than with politicians: document of both types were included. We choose documents with more than 50 comments, in order to have a larger chance for many variants and some internal discussion among the commentators. There are 69.6 persons, 55.8 location, 44.4 organization and 50 miscellaneous entities on average per document (news article plus comments) in the data set. On average, each real world entity was mentioned 5.6 and 5.3 times in the Dutch and English data sets, respectively. In the entire annotated data set 29% of the entities were referred to using more than one surface form and 5% had five or more distinct forms. See Table 1 for details.

We asked two assessors to extract and normalize all references to NEs in the articles and comments. For this purpose we split the input in such a way that each token of the data occurs on a line; on the same line, following the token, an NE tag (if applicable) and the normalized form had to be provided by our assessors. Assessors were asked to provide a Wikipedia article title as the normalization result, if possible, or perform within-document normalization otherwise.

[5] *De Telegraaf* `http://www.telegraaf.nl` and *Algemeen Dagblad* `http://www.ad.nl`.
[6] `http://news.bbc.co.uk/`.

Table 1: Evaluation data: length in words of article with comments (#W); number of comments (#C); total number of named entities (#NE); number of unique real world entities referred to (#RWE).

Category	#W	#C	#NE	#RWE
Dutch language evaluation set				
Persons (politics)	8,602	196	508	67
Persons (show business)	3,705	90	166	22
Locations	7,133	175	214	64
Organizations	2,516	47	190	34
Products	2,642	50	172	32
English language evaluation set				
Persons (politics)	4,265	62	241	43
Persons (show business)	2,230	30	80	16
Locations	4,050	67	257	49
Organizations	2,353	42	137	29
Products	2,324	39	163	26

4.3 Evaluation measures

The media analysis use case that motivated the work in this paper, suggests two important analysis dimensions: (1) who/what is mentioned and (2) how often. Our evaluation measures reflect this interest. For both dimensions we define separate notions of recall and precision.

Measuring real-world entities.

Let UNF_{gt} and UNF_{nen} denote the sets of real-world entities discovered by the assessors and by our NEN algorithm, respectively. We then define

$$\textit{entity-recall} = \frac{|UNF_{nen} \cap UNF_{gt}|}{|UNF_{gt}|} \quad (1)$$

$$\textit{entity-precision} = \frac{|UNF_{nen} \cap UNF_{gt}|}{|UNF_{nen}|}. \quad (2)$$

It is natural to order the sets of real world entities UNF_{gt} and UNF_{nen} by the number of mentions of each real world entity. Using that order we define *entity-precision@n* as the number of correctly identified entities within the first n returned entities, divided by n.

Measuring occurrences.

Let EM_{gt} and EM_{nen} denote the sets of triples (reference-string, normalized-form, position) discovered by the assessors and by our NEN algorithm, respectively. We define the set of true positives TP as: $EM_{nen} \cap EM_{gt}$. Now define the standard notions of precision and recall:

$$\textit{recall} = \frac{|TP|}{|EM_{gt}|} \quad (3)$$

$$\textit{precision} = \frac{|TP|}{|EM_{nen}|}. \quad (4)$$

We also want to measure the quality of normalization separately from the recognition step. Let FP_{norm} be the set of triples ⟨reference-string, normalized-form, position⟩ which are correctly recognized but incorrectly normalized. Thus $FP_{norm} = \{\langle e, n, p\rangle \in EM_{nen} \mid (e, n', p) \in EM_{gt} \text{ and } n \neq n'\}$. We measure the accuracy of normalization using the following metric:

$$\textit{accuracy} = \frac{|TP|}{|TP| + |FP_{norm}|}. \quad (5)$$

Because the last three measures work on individual NE occurrences we use those to evaluate the effect of each of our five improvements.

4.4 Significance testing

We perform significance testing only for accuracy (Eq. 5). There we compare two lists of triples and check for each triple if either they match perfectly with each other or there is no match. Thus we have a list of binary comparison numbers. In this case the McNemar test is appropriate. We look for significance at the $p = 0.01$ level.

5. EXPERIMENTAL EVALUATION

For evaluation purposes, we used the test set described in Section 4.2. We used two named entity recognition tools in our NEN algorithm: [19] for Dutch and [5] for English. For our NEN algorithm we used the Dutch Wikipedia of November 2006 and the English Wikipedia of August 2007.

5.1 Results

The overall accuracy of the baseline NEN algorithm was 80% and 65% for Dutch news articles and news articles plus comments, respectively, and 94% and 77% for English news articles and news articles plus comments, respectively. This answers our first research question, RQ1: the baseline NEN algorithm performs worse on UGC than on edited text (news only).

In order to answer the remaining research questions, we evaluated the baseline system, the full system with all improvements, and the full system with individual improvements switched off. Table 2 lists the results for the Dutch and English data. Table 3 shows the *entity-recall*, *entity-precision* and *entity-precision@10* measurements. `Baseline` denotes the baseline NEN algorithm, `FS` denotes the combined improvements, and $\texttt{Ignore}_k$ denotes the system with the kth improvement disabled ($k = 1, \ldots, 5$, see Section 3.3).

Table 2: Precision, recall and accuracy of baseline, full system (FS) and derived versions of the NEN algorithm. $\texttt{Ignore}_k$ is FS without the k-th improvement. Significant improvements over the baseline are marked with ▲ (p = 0.01)

	recall%	**precision**%	**accuracy**%
Evaluation results on the Dutch language data set			
Baseline	46	45	65
$\texttt{Ignore}_1$	70	60	87▲
$\texttt{Ignore}_2$	64	53	79▲
$\texttt{Ignore}_3$	80	65	88▲
$\texttt{Ignore}_4$	70	**66**	88▲
$\texttt{Ignore}_5$	70	54	78▲
FS	**81**	62	**89**▲
Evaluation results on the English language data set			
Baseline	70	72	77
$\texttt{Ignore}_1$	85	84	**90**▲
$\texttt{Ignore}_2$	79	75	83▲
$\texttt{Ignore}_3$	84	80	87▲
$\texttt{Ignore}_4$	84	**85**	89▲
$\texttt{Ignore}_5$	85	81	88▲
FS	**89**	82	**90**▲

Tables 2 and 3 provide an answer to our second research question, RQ2. `FS` outperformed the baseline for both Dutch and English, according to all evaluation metrics. Also, `FS` is highly recall-oriented (81% vs. 46% for Dutch, and 89% vs. 70% for English) and more accurate than the baseline (89% vs. 65% for Dutch, and 90% vs. 77% for English).

Table 3: Entity-precision, Entity-recall and Entity-precision@10 of the baseline and the full system.

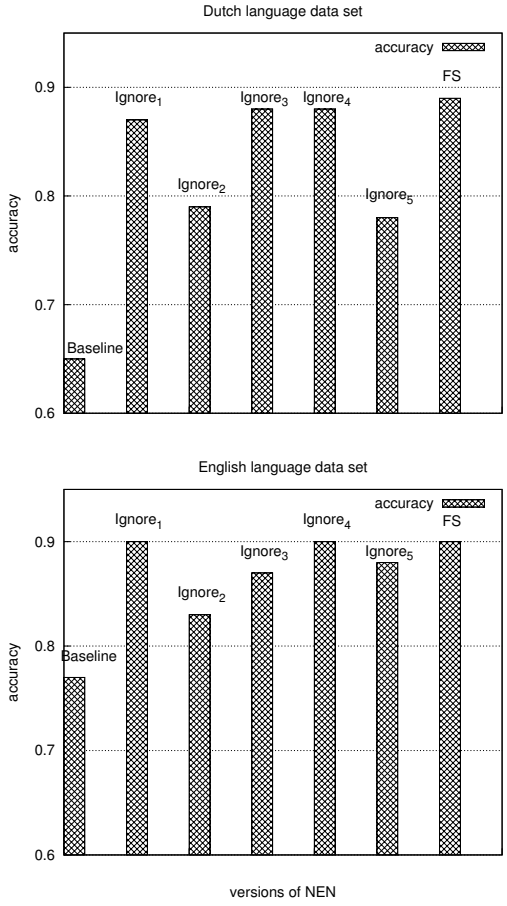
	e-recall	**e-precision**	**e-precision@10**
Evaluation results on the Dutch language data set			
Baseline	51	21	68
FS	**69**	**34**	**85**
Evaluation results on the English language data set			
Baseline	73	54	68
FS	**78**	**67**	**84**

In order to answer our third research question, RQ3, we zoom in on the evaluation results in Figure 2, where we compare the reduced versions of the full system $\texttt{Ignore}_k$ with the full system `FS` itself. When we ignore the first improvement (*Pre-processing NEs*), $\texttt{Ignore}_1$, we find no effect on the accuracy for the English data set, but the accuracy drops a little on the Dutch data set. The reason is that comment threads in the English data set tend to be cleaner than in the Dutch articles. The results in Table 2 also show that this improvement is important in increasing recall for both data sets. When we ignore the second improvement (*N-gram NE Normalization*), the third improvement (*Person-name matching*), or the fifth improvement (*Use Wikipedia link text for disambiguation*), the NEN algorithm performs worse than the full system `FS`, not only in terms of accuracy, but also in terms of recall and precision. These three improvements are really important for the performance of our NEN algorithm. When we ignore the fourth improvement (*Finding missing NEs*), we find the highest precision (bold face numbers in Table 2). The reason is that in this improvement we greedily extract all kinds of reference strings. When we split the text into ngrams and use our *Person-name matching* algorithm to handle variants of NEs, a string is sometimes incorrectly matched with an NE. E.g., "grapes" was considered as the last name of "Bill Gates." Finally, the NEN algorithm with all combined improvements and each of its derived versions (denoted by Ignore_k) are significantly better than the baseline at $p = 0.01$.

We now switch to our fourth research question, RQ4, that concerns the performance on different languages. Apart from the language specific rule concerning infixes in Dutch last names (part of the first improvement, in Section 3.3), the only difference between the Dutch and English language version of our NEN method is the corpus we use for normalization, namely, Wikipedia. We examine the difference in accuracy between the baseline NEN algorithm, the NEN algorithm with combined improvements, and its derived algorithms, for Dutch and English. Consider Figure 2. First, while it is hard to draw very firm conclusions given the small number of articles we were able to annotate, we see that, mostly, the different versions of the system perform comparably for the two languages. Ignore_5 (*Use Wikipedia link text for disambiguation*) seems to be a clear exception: on the Dutch data set the accuracy of the NEN algorithm is affected more strongly than on the English data set. The reason for this seems clear: the English version of Wikipedia constitutes a much larger knowledge base than the Dutch version, with ∼88,000 vs. ∼13,000 disambiguation pages.

Finally, we now look at how well our full NEN system han-

Figure 2: Accuracy of different versions of the NEN algorithm for Dutch and English.



dles the types errors discussed in Section 3.2. For the single development article, the full system corrected 22 our of 27 recognition errors, 1 out 1 multi-references were correctly split and resolved, 13 variants out of 14 were normalized correctly, 15 out of 16 missing entities were detected, and 1 out of 1 incomplete entity was resolved correctly. We also examined the same error types on the test data (containing 1,980 entities). Table 4 compares the performance of the baseline and the full system for different error types.

Table 4: Sources of errors in the test data generated by the baseline and the full NEN system.

Error type	Baseline	Full system
Recognition errors	227	16
Missing entities	210	85
Multi-references	28	2
Variants	54	3
Incomplete entities	13	9

5.2 Efficiency

On standard desktop hardware, the baseline NEN algorithm took on average 4.5s per document (article plus comments), and the full system `FS` and all other versions except Ignore_4 took 15.2s. The fourth improvement (*Finding missing NEs*) is the most expensive step of the NEN algorithm, accounting for approx. 10s of the running time of the full system. Hence, this improvement, although helpful to increase recall, is expensive in terms of running time.

5.3 NEN on real world data

We applied the full NEN system to the articles and user comments collected during 3 months (February 14 to May 14 2008) from 9 Dutch online newspapers. In total, we collected 38,629 articles with 88,651 comments. We further analyzed 7,980 articles that had at least one user comment. For this subset, on average we found 11.1 comments per article.

On average, our NEN system identified 8.2 normalized entities (NoEs) in the content of an article, and 15.0 NoEs in an article together with all its comments. In the content of the 7,980 articles with comments, on average there were 1.65 distinct surface variants for each NoE. When the articles are considered together with comments, we found 1.94 variants per NoE. Moreover, in the latter set, for 28% of NoEs we found more than one variant, and for 4%—more than 5 variants. These numbers show that NEN is indeed an important problem for user generated content.

We looked at NoEs with the largest number of variants. These were: *Nederland* (The Netherlands) with 704 variants, *Geert Wilders* (controversial Dutch politician) with 350, *Verenigde Staten* (United States) with 341, *Rotterdam* with 282. Below we list the most frequent variants for *Verenigde Staten*, as identified by our NEN algorithm, with their English description and frequency:

Amerikaanse	American	670
VS	US	353
Amerika	America	271
Verenigde Staten	United States	172
Amerikanen	Americans	149
USA	USA	122
NS	The Dutch national railway	65
US	US	42
Amerikaans	American	41
Amerikaan	American	35
Witte Huis	White House	33
De VS	The US	22
America	America	21

6. CONCLUSION

Our aim in this paper was to create a named entity normalization algorithm for use in Dutch language media and reputation analysis settings that performs well on user generated content (UGC). For this purpose we started with a baseline NEN system from the literature, and found that it performed much worse on UGC than on edited news: 65% vs. 80% accurracy on a Dutch language data set and 77% vs. 94% accuracy on an English language data set.

We identified the following main sources of errors of the baseline system when applied to UGC: NE recognition errors (incorrect boundaries of named entities or missing NEs), multiple ways of referring to the same entity, and ambiguous (out of context) references. We addressed these issues by proposing five improvements to the baseline NEN algorithm. Our experimental results showed that all improvements are important in increasing recall, precision and accuracy of the algorithm. While helpful in increasing the recall, the improvement that we introduced to cover missing NEs

is expensive in terms of running time. The overall system can run on multiple languages, and the main source of differences in performance between languages seems to be the size of the underlying corpus against which named entities are normalized, Wikipedia.

In future work we will attempt to further improve the performance of our NEN algorithm by using context-aware named entity disambiguation, creating small entity-specific language models. In addition, we want to improve the underlying NER tools we use and to consider other measures of string similarity than we have used so far (edit distance) to handle misspellings of the person names better.

7. ACKNOWLEDGEMENTS

This research was supported by the Netherlands Organization for Scientific Research (NWO) under project numbers 220-80-001, 017.001.190, 640.001.501, 640.002.501, 612.066.-512, and by the Dutch-Flemish research programme STEVIN under project DuOMAn, and by the E.U. IST programme of the 6th FP for RTD under project MultiMATCH contract IST-033104.

8. REFERENCES

[1] D. Ahn, J. van Rantwijk, and M. de Rijke. A cascaded machine learning approach to interpreting temporal expressions. In *Human Language Technologies 2007: Proc. ACL 2007*, pages 420–427, 2007.

[2] C. L. Borgman and S. L. Siegfried. Getty's synoname and its cousins: A survey of applications of personal name-matching algorithms. *Journal of the American Society for Information Science*, 43(7):459–476, 1992.

[3] A. M. Cohen. Unsupervised gene/protein named entity normalization using automatically extracted dictionaries. In *ACL-ISMB Workshop on Linking Biological Literature, Ontologies and Databases*, pages 17–24, 2005.

[4] S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL '07*, pages 708–716, 2007.

[5] F. de Meulder and W. Daelemans. Memory-based named entity recognition using unannotated data. In *Proceedings of CoNLL-2003, Edmonton, Canada*, pages 208–211, 2003.

[6] A. Doan and A. Y. Halevy. Semantic-integration research in the database community. *AI Mag.*, 26(1):83–94, 2005.

[7] X. Dong, A. Halevy, and J. Madhavan. Reference reconciliation in complex information spaces. In *Proc. SIGMOD '05*, pages 85–96, 2005.

[8] A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios. Duplicate record detection: A survey. *Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.

[9] D. Feng, E. Shaw, J. Kim, and E. Hovy. Learning to detect conversation focus of threaded discussions. In *HLT-NNACL*, 2006.

[10] S. Fissaha Adafre and M. de Rijke. Discovering missing links in Wikipedia. In *Proceedings of the Workshop on Link Discovery: Issues, Approaches and Applications (LinkKDD-2005)*, 2005.

[11] M. Khalid, V. Jijkoun, and M. de Rijke. The impact of named entity normalization on information retrieval for question answering. In *Proc. ECIR 2008*, 2008.

[12] J. Kim, G. Chern, D. Feng, E. Shaw, and E. Hovy. Mining and assessing discussions on the web through speech act analysis. In *Proceedings of the Workshop on Web Content Mining with Human Language Technologies at the 5th International Semantic Web Conference*, 2006.

[13] W. Magdy, K. Darwish, O. Emam, and H. Hassan. Arabic cross-document person name normalization. In *CASL Workshop '07*, pages 25–32, 2007.

[14] G. S. Mann and D. Yarowsky. Unsupervised personal name disambiguation. In *Proceedings of the 7th Conference on Natural Language Learning at HLT-NAACL 2003*, pages 33–40, 2003.

[15] R. Mihalcea and A. Csomai. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the 17th ACM Conference on Conference on Information and Knowledge Management*, pages 233–242, 2007.

[16] G. Mishne. *Applied Text Analytics for Blogs*. PhD thesis, University of Amsterdam, Amsterdam, 2007.

[17] C. Phua, V. Lee, and K. Smith. The personal name problem and a recommended data mining solution. In *Encyclopedia of Data Warehousing and Mining (2nd Edition)*. 2006.

[18] Y. Song, J. Huang, I. G. Councill, J. Li, and C. L. Giles. Efficient topic-based unsupervised name disambiguation. In *Proceedings of the 2007 Joint Conference on Digital libraries*, pages 342–351, 2007.

[19] E. F. Tjong Kim Sang. Memory-based named entity recognition. In *Proceedings of CoNLL-2002, Taipei, Taiwan*, pages 203–206, 2002.

[20] M. Weimer, I. Gurevych, and M. Mehlhauser. Automatically assessing the post quality in online discussions on software. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 125–128, 2007.

[21] W. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Bureau of the Census, Washington, DC., 1999.

[22] W. Zhou, C. Yu, N. Smalheiser, V. Torvik, and J. Hong. Knowledge-intensive conceptual retrieval and passage extraction of biomedical literature. In *SIGIR '07*, pages 655–662, 2007.

APPENDIX

Person name matching algorithm

Algorithm 5 takes three arguments: a source string, a target string and a flag indicating whether the target string follows a person title (e.g., Mr, Mrs, etc.). The algorithm checks whether the target string is a variant of the source string.

Algorithm 5 Person name matching algorithm

Require: a source string S, a target string T, a flag L
if T = S, or T = firstName(S), or T = lastName(S) **then**
  return matched
end if
if lastName(S) $\neq$ NULL **then**
  if lastName(T) = NULL **then**
    if L = TRUE and lastName(S) startsWith T **then**
      return matched
    else if firstChar(T) = firstChar(lastName(S)) **then**
      return isSimilar(lastName(S), T) {isSimilar(str1, str2) returns true if $\frac{editDist(str1,str2)}{length(str2)} \leq 0.34$}
    end if
  else if firstName(S) startsWith firstName(T) **then**
    if lastName(S) startsWith lastName(T) **then**
      return matched
    else
      return isSimilar(S, T)
    end if
  else if lastName(S) startsWith firstName(T) **then**
    return isSimilar(lastName(S), T)
  end if
else if lastName(T) = NULL and firstChar(S) = firstChar(T) **then**
  return isSimilar(S, T)
end if