Entity recognition for duplicate filtering

J.A. Cordero Cruz S.E. Garza S.E. Schaeffer

Abstract

We propose a system for automatic detection of duplicate entries in a repository of semi-structured text documents. The proposed system employs text-entity recognition to extract information regarding time, location, names of persons and organizations, as well as events described within the document content. With structured representations of the content, called "metamodels", we group the entries into clusters based on the similarity of the contents. Then we apply machine-learning algorithms to the clusters to carry out duplicate detection. We present results regarding precision, recall, and F-value of the proposed system.

1 Introduction

It is now possible to easily create and share text content, which results in vast repositories of information ready to be queried and analyzed. The content of such repositories undergoes constant change: updates, deletions, and insertions of documents are frequent. Therefore, it becomes a complex task to monitor that no *duplicate* entries (that is, entries with redundant content considering the other entries present in the repository) are created as new documents arrive and the old ones are being edited.

In this work we propose a system for detecting duplicates among text documents that contain mentions of geographical locations, instances of time, and events. The operation of the proposed system can be outlined as follows: first, the words of the document are *labeled* to identify the entities contained in that document; the labeling marks locations, times, names of people and organizations, and description of events that have occurred. Then, a structured representation called a *metamodel* is created from the labeled contents, and finally, clustering algorithms are applied to the metamodels to reduce the number of comparisons necessary to identify *duplicates* among the metamodels.

As a case study to test the proposed system we chose the citizen reports received at the CIC (Center of Citizen Integration; in Spanish, *Centro de Integración Ciudadana*) in the metropolitan area of Monterrey, Mexico, available at `http://www.cic.mx`. The CIC reports describe, among other incidents, traffic accidents, road conditions, and need of maintenance of public infrastructure. Active citizens create reports through a web application (also available on mobile platforms) called Tehuan (available at `http://www.tehuan.cic.mx`) or by

```
*ACCIDENTE* Leones y 18 Av carril de alta MTY #mtyfollow 19:53 via @vigila2 cc @spvmty
```

Figure 1: An example of a CIC report via Twitter, reporting an accident at the intersection of streets called "Leones" and "18" that is blocking a lane.

sending a message on Twitter (cf. `https://twitter.com`) — called a *tweet* — either mentioning the account `@Cicmty`. Upon reception, the CIC staff structures (in large part manually) the received report and inserts it into a repository.

The CIC reports include information regarding the event or condition that is being reported, the geographical location (street address or even GPS coordinates of the reporting mobile device), and the time that the event took place (possibly described verbally, but always at least as a time stamp of the report's reception). Regardless of the origin of the report, the text of the resulting document is condensed and the language used is abbreviated. Figure 1 shows an example of a Twitter-originated CIC report regarding a car accident, in Spanish. We consider two CIC reports to be *duplicates* if a human observer would interpret them as referring to a single event upon examining them both.

The remainder of this article is structured as follows: in Section 2 we discuss the identification of names within text, after which in Section 3 we discuss related literature. Section 4 details our proposed solution and then discuss the case study more closely in Section 5. Finally, in Section 6 we conclude the present work and discuss opportunities for future work.

2 Named entity recognition

The goal of *named entity recognition* (NER) is to extract words from text and classify them into predefined categories known as *entities*. Possible entities of interest include names of persons, names of places, and dates. There are several methods for NER and in this work we use a simple version based on hidden Markov models (HMM) [5, 10, 13], the implementation of which we now explain.

A document $\mathbf{x} = x_1 x_2 \dots x_n$ is represented as a sequence of n words x_i. The task is to assign for each x_i a *label* y_i, resulting in a labeling sequence $\mathbf{y} = y_1 y_2 \dots y_n$, where all labels y_i belong to a predefined set $K = \{e_1, e_2, \dots, e_k\}$. The labels in $\mathbf{y}$ are chosen by maximizing the *joint probability* between a given text $\mathbf{x}$ and a labeling sequence $\mathbf{y}$:

$$P(\mathbf{y}, \mathbf{x}) = P(\mathbf{x} \mid \mathbf{y}) P(\mathbf{y}), \tag{1}$$

where $P(\mathbf{x} \mid \mathbf{y})$ is the *conditional probability* of generating the text $\mathbf{x}$ given the labeling sequence $\mathbf{y}$ and $P(\mathbf{y})$ corresponds to an *a priori* probability distribution over the labeling sequence $\mathbf{y}$ [13, 16].

Using a second-order HMM the computation of $P(\mathbf{y}, \mathbf{x})$ is simplified to

$$P(\mathbf{y}, \mathbf{x}) = \prod_{i=1}^{n+1} P(y_i \mid y_{i-1}, y_{i-2}) \prod_{i=1}^{n} P(x_i \mid y_i). \tag{2}$$

Now, calculating the parameters $P(y_i \mid y_{i-1}, y_{i-2})$ and $P(x_i \mid y_i)$ of the HMM is easy as these are based on unigrams, bigrams, and trigrams (that is, sequences of a single label, two labels, and three labels, respectively) [4], as

$$P(s \mid u, v) = \frac{c(u, v, s)}{c(u, v)} \text{ and } P(x \mid s) = \frac{c(s \rightsquigarrow x)}{c(s)}, \tag{3}$$

where, for a given set of labeled words, $c(u, v, w)$ is the number of occurrences of the label trigram (u, v, w), whereas $c(u, v)$ is the number of occurrences of the label bigram (u, v), and $c(u)$ the number of occurrences of the label unigram (u); the number of times that the unigram $c(s)$ corresponds to the word x is denoted by $c(s \rightsquigarrow x)$.

Once these parameters are computed, the label sequence is obtained with the *Viterbi* algorithm [6, 10], shown as Algorithm 1; the `STOP` label of the algorithm is introduced to allow the algorithm to operate with word sequences of different lengths [12].

Algorithm 1 Pseudocode of the Viterbi algorithm.

Require: a text sequence $x_1 \dots x_n$, parameters $P(s \mid u, v)$ and $P(x \mid s)$.
  $\forall (u, v)$ such that $(u \neq *) \vee (v \neq *)$, assign $\pi(0, *, *) = 0$
  for $k = 1 \dots n$ **do**
    for $u \in K, v \in K$ **do**
      $\pi(k, u, v) = \max_{w \in K}(\pi(k-1, w, u) \times P(u \mid w, u) \times P(x_k \mid v))$
      $bp(k, u, v) = \arg\max_{w \in K}(\pi(k-1, w, u) \times P(u \mid w, u) \times P(x_k \mid v))$
    end for
  end for
  Assign $(y_{n-1}, y_n) = \arg\max_{(u,v)}(\pi(n, u, v) \times P(\texttt{STOP} \mid u, v))$
  for $k = (n-2) \dots 1$ **do**
    $y_k = bp(k+2, y_{k+1}, y_{k+2})$
  end for
  return the labeling sequence $y_1 \dots y_n$

3 Related work

The proposed system bears similarity with existing work on Twitter analysis. Tao et al. [19] propose a method for detecting near-duplicate tweets by first determining if two tweets are considered duplicates and then assigning a level of duplicity to the pair of tweets varying from exact copy to somewhat overlapping, using five levels. Natural similarity measures to attempt on tweets include the *edit distance* (also known as the Levenshtein distance) [5, 8, 11], the proportion of shared words, as well as the proportion of shared hashtags. NER is used to obtain semantic characteristics such as the proportion of shared entities. Also, the message time stamps and the similarity of the Twitter accounts that sent the tweet are considered. The duplicate detection in itself is carried out as *logistic regression*, given the similarity data.

Another work is that of Sankaranarayanan et al. [17], extracting and analyzing news in Twitter. First the system filters the news messages from the rest of the tweets, then groups the news tweets to obtain those related to a same news story and then detects the type of news that is being reported. The filtering of news versus not news is done by a *naïve Bayesian classifier* [3, 9] that has been previously trained with a set of tweets marked as either "news" or "noise". The grouping of the news tweets is done based on the text contents as well as metadata, extracting the topic and the location (again with NER).

Agarwal et al. [1] focus on detecting local news on Twitter that report fires at factories as well as strikes. Their system operates in four stages: first, the messages that contain information relevant to the topics of interest are filtered using *regular expressions* and supervised classifiers, after which the resulting tweets are compared with those of the last 24 hours, grouping the ones that correspond to the same event, and then NER is used to extract characteristics such as the duration, the location, and the type of the event, and finally groups that have very similar characteristics are merged.

Systems that work with duplicate detection in larger text documents (and hence can base the detection on a much larger set of data per element) include the DUDE system[1] of University of Michigan for technical papers. A framework for finding duplicates among XML documents with a known schema is presented by Weis and Naumann [20], whereas a similarity calculation for text documents is presented by Schleimer et al. [18].

As the CIC reports are extremely brief, we do not expect systems designed for longer documents to be able to function well on our data set. Also, the order in which information is presented is not relevant for the CIC reports to be considered duplicates, whereas for example detecting whether one text document copies fragments of another would require several words to follow one another in a near-identical manner to detect duplicity — a CIC report is usually shorter than a normal sentence in written text. The work of Gong et al. [7] is intended for short texts, but does not incorporate the element for determining the similarity of points of time expressed in the text that is necessary for the CIC reports.

4 Proposed solution

We propose a four-stage process for the duplicate detection. In this section, we discuss the details of each stage as well as the steps involved.

Preprocessing into metamodels: The preprocessing aims to structure the document for posterior analysis. Information is *extracted* from the document. For example, the CIC reports are available in XML and JSON and a corresponding parser is applied to access the elements of the document and obtain the textual content. *Filters* are then applied to eliminate undesired features such as non-ASCII characters or stop words, as well as

[1] Available online at `http://sigda.eecs.umich.edu/DUDE/`.

substitutions such as replacing abbreviations with full words. Using NER, *labels* are then assigned to the remaining words. The label categories used depend on the type of documents being processed. Based on the labeled sequence of words, sets of words are created by joining those that received the same label. These are stored along with the corresponding label in a template to create the metamodel.

Metamodel clustering: The metamodels are classified into clusters of similar metamodels (this can be done either incrementally as an online clustering or globally as a one-time static clustering). The goal is to reduce the set of metamodels with which the duplicate detection is later carried out.

Classifier training: Within the clusters of metamodels created by the previous step, classifiers are trained to distinguish between duplicate and non-duplicate metamodels, taking into account data regarding the location, time, and type of event.

Duplicate detection: The trained classifiers of each cluster are then presented with pairs of metamodels over all pairs of the cluster if this is a global, static analysis, and between a new metamodel and the metamodels already included in the cluster in the case of an online, incremental processing. If two metamodels are classified as duplicates, then the corresponding input documents are considered to be duplicates.

5 Case study

In this section we discuss the application of our proposed duplicate-detection system on the CIC citizen reports (discussed already in Section 1). The reports of the CIC were downloaded in the JSON format from the developer API of the CIC (available at `http://www.developers.cic.mx/api`); an example is shown in Figure 2. The CIC reports contain the following fields: `ticket` is a unique ID assigned to each report received, `content` contains the description of the reported event, `created_at` is a time stamp of the report creation, `address_detail` is the address with some typical fields (if available), and `categories` are predefined categories of the CIC for types of reports.

In this work, we downloaded reports from the following categories (with an abbreviation indicated in parenthesis, derived from the Spanish name used by the CIC for each category): accidents (`acc`), street lights (`alu`), traffic lights (`sem`), road damage (`bac`), sewer lids (`alc`), public events (`eve`), road work or closure (`obr`), and situations of risk (`sit`). Within the categories, we employ the different criteria for defining whether two reports are to be considered duplicates. For example, traffic-accident reports are considered as duplicates when the text describing the location is similar and the time lapse between the reports does not exceed 15 minutes, whereas if the time lapse is higher (even with the location being the same), the reports are considered distinct. This is not the same for missing or damaged sewer lids, for example: the same lid may be reported several

```
"ticket": "#7YPC",
"content": "*ACCIDENTE* En gonzalitos altura de vuelta izquierda a Insurgentes MTY #mtyfollow 17:37",
"created_at": "2013-08-04T17:49:56-05:00",
"address_detail": {
   "formatted_address": "Gonzalitos 655, Sin Nombre de Colonia 31, Monterrey, NL, México",
   "zipcode": "64000",
   "county": {
      "long_name":"Monterrey", "short_name":"Monterrey",
   },
   "state": {
      "long_name": "Nuevo León", "short_name":"Nuevo León"
   },
   "neighborhood": {
      "long_name": "Centro", "short_name": "Centro"
   }
},
"categories": ["ACCIDENTE"]
```


Figure 2: An example of a CIC report in JSON.

hours apart or on different days. Hence the classifiers are trained separately for each metamodel cluster.

5.1 Preprocessing

For the preprocessing phase, we employ the Python library `json`[2] to extract the following fields: `'ticket'`, `'content'`, `'created_at'`, `'categories'`, and `'address_detail'` for each report. From the date, the UTC date and time was parsed into fields for year, month, day, hour, minute, and second. Then, we apply the filters to clean up the data; we mention some examples of the filters used:

- Replace all accented characters with their ASCII equivalent.
- Eliminate everything that begins with `http`.
- Eliminate special characters such as `*`, `:`, `?`, `;`, `-`.
- Eliminate (with regular expressions) all hashtags and Twitter accounts.

At this point *no* stop-word elimination has yet been applied, as the prepositions are important for correct identification of place names in Spanish with NER: expressions such as `entre Avenida P. Livas y Las Americas` (between two specific avenues), `en Paseo de los Leones` (at a specific street), `rumbo a Lazaro Cardenas` (near a specific avenue). The categories used for the NER labeling of the reports are the following: places (`LOC`), time (`TIME`), persons (`NAME`), organizations (`ORG`), and event descriptions (`DESC`); a label for irrelevant information (`IRR`) was also employed, as done in the work of Ratinov and Roth [14]. The Vitebri algorithm (Algorithm 1 on page 3) was employed to assign

[2]Available at `https://docs.python.org/2/library/json.html`.

```
ACCIDENTE en Ave. Garza
Sada sin lesionados 6:30
pm MTY NL gracias
```

(a) Cleaned text.

```
[ACCIDENTE, DESC], [en, LOC], [Ave., LOC], [Garza,
LOC], [Sada, LOC], [sin, DESC], [lesionados, DESC],
[6:30, TIME], [pm, TIME], [NL, LOC], [gracias, O]
```

(b) Labeled text.

```
<metamodel>
   <desc>ACCIDENTE sin lesionados</desc>
   <loc>en Ave. Garza Sada</loc>
   <time>6:30 pm</time>
</metamodel>
```

(c) Metamodel.

Figure 3: Cleaned text sequence, the corresponding labeled sequence, and the resulting metamodel.

the labels. Given the labeling, the metamodel is composed (as described in Section 4). Figure 3 shows an example of an input, the resulting labeling, and the created metamodel.

The clean-up carried out upon creating the metamodel out of the labeled sequence involves the elimination of stop words, making all words lowercase, replacing plural nouns by their singular forms, and elimination of word repetition.

The information in the metamodels is then accessed by querying on three text elements: `tinfo` that describes the reported event (formed by cleaning the content of the labels `DESC`, `NAME`, and `ORG` of the metamodel), `tloc` that indicates the location (simply the cleaned-up content of the label `LOC` in the metamodel), and `ttime` that states the time stamp at level of minutes as UNIX time. Time differences are measured (for purposes of evaluating their similarity) as

$$\Delta_{t_i,t_j} = 1 - \big(1 + \log_{10}(|t_i - t_j|)\big)^{-1}, \tag{4}$$

where t_i is the UNIX time of the first metamodel and t_j that of the second.

5.2 Classifier training

Eight support vector machine classifiers [3] are trained to detect duplicate pairs of metamodels, one per CIC category. The training commences by creating h training triples

$$\mathcal{T} = [(m_1^1, m_2^1, \delta^1), (m_1^2, m_2^2, \delta^2), \ldots, (m_1^h, m_2^h, \delta^h)], \tag{5}$$

where for the ith triplet, m_1^i y m_2^i are two (distinct) metamodels and $\delta^i \in [0, 1]$ is a binary decision variable: zero indicates that they are not duplicates whereas one indicates that the two metamodels are considered duplicates of one another. Each triple is then processed:

1. The fields `desc`, `loc`, and `time` are accessed for both metamodels.

2. Two weighted vectors are created for both metamodels using *term frequency - inverse document frequency* or tf-idf [12]: vector $\mathcal{I}_j$ is based on `tinfo` and vector $\mathcal{L}_j$ is based on `tloc` for metamodel $j \in \{1, 2\}$. The vocabulary employed for the terms was created manually from a sample of 1,784 metamodels and another distinct sample set was used for training each classifier.

3. The *cosine similarity* [2], defined for two vectors $\mathbf{v}$ and $\mathbf{w}$ as

$$(\mathbf{v} \cdot \mathbf{w})/(|\mathbf{v}||\mathbf{w}|), \tag{6}$$

is computed for $\mathcal{I}_1$ versus $\mathcal{I}_2$ (we denote the result by $\rho_{\mathcal{I}}$) and also for $\mathcal{L}_1$ versus $\mathcal{L}_2$ (yielding $\rho_{\mathcal{L}}$).

4. The time difference is computed (cf. Equation (4)); we denote this simply by Δ when the two metamodels used to obtain the time stamps are implicitly clear.

Then, a characteristic matrix of dimension $h \times 3$ is created together with a $1 \times h$ column vector:

$$\mathbf{X} = \begin{bmatrix} \rho_{\mathcal{I}}^1 & \rho_{\mathcal{L}}^1 & \Delta^1 \\ \vdots & \vdots & \vdots \\ \rho_{\mathcal{I}}^h & \rho_{\mathcal{L}}^h & \Delta^h \end{bmatrix} \text{ and } \vec{y} = \left[\delta^1\, \delta^2 \ldots \delta^h\right]^{\mathrm{T}}. \tag{7}$$

Using these two, a classifier is then trained for a specific category with the scikit-learn[3] Python-library. The resulting classifier for category ℓ is denoted by $\mathcal{C}_\ell$.

5.3 Duplicate detection

The metamodel clustering for the case study is done simply based on the category assigned by the CIC (we have also carried out experiments using k-means variants to recover the categories based on document similarity with reasonable success). We hence apply the classifier $\mathcal{C}_\ell$ for each category $\ell \in \{\mathrm{acc}, \mathrm{alu}, \ldots \mathrm{sit}\}$ (done either using all pairs of metamodels within that category or upon the introduction of a new metamodel to a set of existing metamodels) for the test set of documents (those used for training and dictionary-creating are excluded). The test set contained 105 metamodels corresponding to the category `acc`, 20 to `alc`, 85 to `alu`, 90 to `bac`, 45 to `eve`, 45 to `abr`, 75 to `sem`, and 40 to `sit`. The pseudocode for the process is shown in Algorithm 2 for the case of adding a *single* new metamodel into a set of existing metamodels of the same category.

5.4 Results

Our results include the evaluation of the NER-labeler (alone) and the evaluation of, properly, the duplicate detection system. With respect to the former, the cleaned-up CIC reports contained a total of 123,583 words (3,823 reports). All these words were manually labeled using the labels discussed in Section 5.1: `LOC`, `TIME`, `NAME`, `ORG`, `DESC`, and `IRR`. Then, the parameters of the NER-labeler were computed. The labeler was tested with a set of 5,099 words extracted from a new set of CIC reports; we created one labeling with the trained labeler

[3] Available at `http://scikit-learn.org`.

Algorithm 2 Duplicate-detection algorithm outline.

Require: incoming metamodel m, existing metamodels M, trained classifier $\mathcal{C}$

$\mathcal{I} \leftarrow$ tf-idf(`tinfo`(m))
$\mathcal{L} \leftarrow$ tf-idf(`tloc`(m))
$t \leftarrow$ `ttime`(m)
$D = \emptyset$ (list of duplicates of m detected within M)
for $m' \in M$ **do**
  $\mathcal{I}' \leftarrow$ tf-idf(`tinfo`(m'))
  $\mathcal{L}' \leftarrow$ tf-idf(`tloc`(m'))
  $t' \leftarrow$ `ttime`(m')
  $\rho_{\mathcal{I}} \leftarrow \mathrm{sim}(\mathcal{I}, \mathcal{I}')$ with Equation (6)
  $\rho_{\mathcal{L}} \leftarrow \mathrm{sim}(\mathcal{L}, \mathcal{L}')$ with Equation (6)
  $\Delta \leftarrow \Delta_{t,t'}$ with Equation (4)
  $\delta \leftarrow \mathcal{C}(\rho_{\mathcal{I}}, \rho_{\mathcal{L}}, \Delta)$ (classifier output)
  if $\delta = 1$ **then**
    $D \leftarrow D \cup \{m'\}$ (add to results the detected duplicate)
  end if
end for
return D

and another manually, obtaining a 92% precision on the automated labeling with respect to the manual one. The success was notable in identifying places, times, and event descriptions, possibly attributable to the limited vocabulary employed in the CIC reports. As pointed out by Ritter et al. [15], existing NER tools tend to perform poorly on Twitter messages; we hence conclude that our labeler has a sufficient performance with the current precision.

For evaluating the reliability of the duplicate-detection system, we performed modifications on CIC reports to produce duplicates, then testing whether the modified duplicates were correctly identified by the classifiers. We used a total of 201 original reports and created two artificially modified duplicates for each, also creating two artificial non-duplicated by applying drastic modification. The modifications were made manually to ensure that the resulting reports make sense and that those intended as duplicates are in fact semantically similar whereas the non-duplicates have differences that permit a human observer to conclude that they are clearly distinct.

The set of metamodels thusly obtained was divided into a training set and a test set as follows: the original 201 metamodels were grouped according to their respective CIC categories (acc, alu, sem, bac, alc, eve, obr, and sit). With the K-iterations method [9], one half of each category was assigned to a training set and the other half to a test set. The modified metamodels (two duplicates and two non-duplicates) were then inserted in the same category and set as their corresponding original.

The classifier training was repeated ten times to study possible variations in the end result; Figure 4 illustrates the precision (on average 55%), recall (on average 84%), and F-value (on average 66%) obtained for the duplicate detector. Recent related work on detecting duplicate tweets by Tao et al. [19]

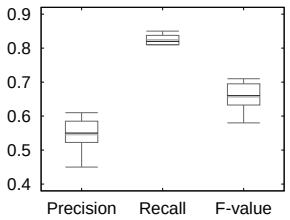

Figure 4: Box-whiskers plots of the statistics of three performance measures over 10 repetitions of the classifier training.

obtained 48% precision and 43% recall, in the light of which our system seems to perform quite satisfactorily given the similarity between the input data in their work and ours. As the authors know of no other system for duplicate detection adaptable to the CIC context, we do not present a comparison between our method and another one; for example, using the system of Tao et al. [19] for comparison would be unfair as the mere textual similarity that suffices for tweets is expected to perform poorly on the CIC reports that may in fact be written by two different people, simply describing the same indicent in different words.

6 Conclusions

We have presented an approach for detecting duplicates within document repositories, based on named entity recognition and supervised classification. The proposed method is tested on a case study using citizen-reported urban incidents in the metropolitan area of Monterrey, Mexico. The similarity between two reports is evaluated in terms of locations, times, events, and names present in the documents. The computational results obtained are better than expected from the performance of state-of-the-art solutions for similar data sets.

Improvements to the present system, left as future work, include parsing verbal expressions of time (phrases like "last Friday" or "at noon") and estimating geographical coordinates (latitude and longitude) based on textual address information (for example through the Google Maps API) to estimate the distance between locations when GPS coordinates are not included in the reports.

The integration of the proposed system as an automated step at CIC upon report reception is left as future work — presently the staff attempts to notice duplicates as a human effort and tend to struggle around personnel-shift changes. Failure to notice a duplicate document may result in a CIC staff member calling the fire department after a colleague already reported the same fire a few moments earlier before heading home.

As future work, also the generalization of the system towards other types of repositories such as scientific publications (to detect attempts of double submission of a single work as well as plagiarism) is of interest. This would require the design of a label set for the NER phase and a redefinition of what similarity means for this type of documents.

References

[1] P. Agarwal, R. Vaithiyanathan, S. Sharma, and G. Shroff. Catching the long-tail: Extracting local news events from Twitter. In *Proc. of the 6th International AAAI Conference on Weblogs and Social Media*, pages 379–382, Palo Alto, CA, USA, 2012. AAAI.

[2] R. Baeza-Yates and B. Ribeiro-Neto. *Modern information retrieval*. ACM Press, New York, NY, USA, 1999.

[3] C. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, NY, USA, 2006.

[4] K. Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proc. of the 2nd Conference on Applied Natural Language Processing*, pages 136–143, Stroudsburg, PA, USA, 1988. Association for Computational Linguistics.

[5] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley-Interscience, New York, NY, USA, second edition, 2000.

[6] R. Esposito and D. Radicioni. CarpeDiem: Optimizing the Viterbi algorithm and applications to supervised sequential learning. *The Journal of Machine Learning Research*, 10:1851–1880, Dec. 2009.

[7] C. Gong, Y. Huang, X. Cheng, and S. Bai. Detecting near-duplicates in large-scale short text databases. In *Proc. of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, pages 877–883, Berlin/Heidelberg, Germany, 2008. Springer-Verlag.

[8] P. Hall and G. Dowling. Approximate string matching. *ACM Computing Surveys*, 12(4):381–402, 1980.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, NY, USA, second edition, 2009.

[10] D. Jurafsky and J. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

[11] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, 1966.

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

[13] N. Ponomareva, P. Rosso, F. Pla, and A. Molina. Conditional random fields vs. hidden Markov models in a biomedical named entity recognition task. In *Proc. of International Conference Recent Advances in Natural Language Processing*, pages 479–483, Borovets, Bulgaria, 2007. RANLP 2007 Organising Committee.

[14] L. Ratinov and D. Roth. Design challenges and misconceptions in named entity recognition. In *Proc. of the 13th Conference on Computational Natural Language Learning*, pages 147–155, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[15] A. Ritter, S. Clark, Mausam, and O. Etzioni. Named entity recognition in tweets: An experimental study. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[16] S. Ross. *A first course in probability*. Pearson Prentice Hall, Harlow, England, 2010.

[17] J. Sankaranarayanan, H. Samet, B. Teitler, M. Lieberman, and J. Sperling. TwitterStand: News in tweets. In *Proc. of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 42–51, New York, NY, USA, 2009. ACM.

[18] S. Schleimer, D. S. Wilkerson, and A. Aiken. Winnowing: local algorithms for document fingerprinting. In *Proc. of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85, New York, NY, USA, 2003. ACM.

[19] K. Tao, F. Abel, C. Hauff, G.-J. Houben, and U. Gadiraju. Groundhog Day: Near-duplicate Detection on Twitter. In *Proc. of the 22nd International Conference on World Wide Web*, pages 1273–1284, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[20] M. Weis and F. Naumann. DogmatiX tracks down duplicates in xml. In *Proc. of the 2005 ACM SIGMOD international conference on Management of data*, pages 431–442, New York, NY, USA, 2005. ACM.