Distant supervision of relation extraction in sparse data

Intelligent Data Analysis 23 (2019) 1145–1166
DOI 10.3233/IDA-184238
IOS Press

Distant supervision of relation extraction in sparse data

Bijan Ranjbar-Sahraei[a], Hossein Rahmani[a,b,*], Gerhard Weiss[a] and Karl Tuyls[c]
[a]*Department of Data Science and Knowledge Engineering, Maastricht University, Maastricht, The Netherlands*
[b]*School of Computer Engineering, Iran University of Science and Technology, Teheran, Iran*
[c]*Liverpool University, UK*

Abstract. To extract structured knowledge from unstructured text sources we need to understand the semantic *relationships* between entities. State-of-the-art relation extraction techniques take advantage of the abundance of data on the web. However, in domains with sparse data such as social networks which have limited occurrences of entities and relationship patterns, bootstrapping techniques and pattern detection methods are inefficient and inaccurate. In this paper, we introduce REDS, a Relation Extraction approach based on Distant Supervision. REDS extracts the named entities from text documents and assigns a fingerprint to each potential relationship among the named entities. Then, it queries a knowledge repository for similar matches to each fingerprint. An assessor uses the query results and the data statistics to measure the validity of the relationships corresponding to the queried fingerprints, and labels each potential relationship with the predicted type. In addition to handling the relation extraction in presence of data sparsity, REDS uses an information retrieval framework that makes it scalable and capable of dealing with noisy data. We implement and test REDS on a non-English historical archive consisting of unstructured notarial acts and structured civil registers; By means of manual evaluations REDS achieves precision of 0.90.

Keywords: Relation extraction, distant supervision, identity resolution

1. Introduction

Named Entity Recognition (NER) and relation extraction are among the main sub-problems of information retrieval [14]. NER locates a word or a phrase that refers to a particular named entity within a text [36], and relation extraction focuses on recognizing relations among the named entities in unstructured text [3]. Both sub-problems have been studied extensively in literature [3,36,48] where supervised and unsupervised techniques have been used to tackle these problems. Among these techniques, supervised learning is capable of extracting both the named entities and relations from unstructured data [30,54,61]. However, the need for manually labeled data renders its application inefficient in real-world settings. An alternative approach is to benefit from the abundance of data on the web. Data redundancy in the web allows for accurate statistical estimates [2,46]; for instance, the state-of-the-art open information extraction tools such as KnowItAll [27] and Open IE [5,26] are designed to collect huge amounts of data from the web. They use bootstrapping techniques and/or data statistics to precisely extract information. For instance, as shown by Ravichandran and Hovey [46] by searching the web for

*Corresponding author: Hossein Rahmani, Department of Data Science and Knowledge Engineering, Maastricht University, Maastricht, The Netherlands. E-mail: h_rahmani@iust.ac.ir.

"Mozort 1756" or "Newton 1642" a large amount of patterns indicating the year of birth of a person are extracted without the need to a training set. Another example is how KnowItAll [27] searches the web for specific phrases like "... such as ..." or "... including ..." for extracting new classes of objects.

In contrast to data sources such as Wikipedia and Freebase which contain vast amount of data with high redundancy, *sparse datasets* have limited occurrences of entities and textual patterns. In absence of *manually labeled training sets* and *data redundancy*, distant supervision uses additional knowledge repositories to perform information extraction. This approach is very promising in dealing with sparse data [34], though there exists challenges in dealing with uncertainties. For instance, a single entity might be referred to with different names due to spelling errors and name variations, or multiple entities might be referred with identical names. Therefore, it is mandatory to use entity resolution techniques [17,18] and in particular *identity resolution* approaches [6,8,57] prior to extraction of relations.

In this paper, we introduce the notion of *relationship fingerprint*; a relationship fingerprint, or in short a fingerprint, is a noise-tolerant condensed representation of a relationship between two or more individuals; each fingerprint maps the important attributes of the individuals involved in the relationship to a unique tuple. Fingerprints can be indexed and searched in an efficient way. REDS uses a combination of orthographic and dictionary lookup features to extract named entities from the unstructured data. It constructs the candidate relationships from the tuples of named entities existing in each single unstructured document. Fingerprints corresponding to the candidate relationships are generated, and the existing knowledge repository is queried for each fingerprint. If the knowledge repository contains any *Levenshtein neighbor* of the fingerprint the relation between the named entities is revealed. Besides, the statistics of data are used to assess the confidence level of the discovered relation and predict its type. As an additional step, by using the extracted relations as a training set, we learn a classifier to extract relations from the portions of unstructured data for which no knowledge repository exists. In short, the contributions of this paper can be summarized as follows: 1) Augmentation of relation extraction procedure with an identity resolution method; 2) Scalability of relation extraction for large datasets due to using an inverted index; 3) High precision of relation extraction due to using relationship fingerprints, and 4) no need to any initial training set and being language independent.

The rest of this paper is organized as follows: In Section 2, we introduce the existing work on the identity resolution and relation extraction approaches. Section 3 defines, formally, the problem we are addressing and also introduces the data setup used for examples and experiments. Then, the architecture of REDS approach and its components are described in Section 4 by means of various examples. In Section 5, we report on the experiments and evaluations of implementing REDS on a large corpus; Section 6 discusses the advantages and limitations of this work in practice. Section 7 concludes.

2. Related work

This work is mainly based on two concepts of *identity resolution* and *relation extraction*. We, first introduce related work on identity resolution and discuss the scalability problem in many of the existing work. Then we review some of the existing work on relation extraction, and position our work with respect to the established methods.

2.1. Identity resolution

Dealing with the *identity resolution* problem is the first step in mining and analysis of various social networks. Due to appearence of multiple Online Social Networks (OSNs) in recent two decades, integrating these networks has become an interesting but challenging research topic. Applications of the

identity resolution in OSNs include targeted advertisements [10,53], building expertise networks [39,40] and refining the individuals' contact lists [6]. In addition to the OSNs which provide networked data, the genealogical data sets also provide information about a large group of social agents and their interactions in the course of the centuries [49]. Prior to analysis of historical archives, the implementation of identity resolution is a mandatory task, as social agents are mentioned in different documents in absence of any unique identification number. The name variations, errors and missing data are among the challenges in dealing with such datasets.

The first intuitive approach in identifying a group of so-called *references* which refer to the same entity is to compare the reference attributes. In a social network setting, such attributes might include the *given name* and *family name*, *gender*, *dates* and *locations*. Vosecky et al. [56], and Motoyama and Varghese [35] studied such biographical attributes for searching and matching individuals across multiple OSNs. Köpcke et al. [31] gave an extensive comparison of some of the non-learning and learning based approaches which mainly use the biographical attributes. Efremova et al. [23] studied a baseline approach based on the biographical attributes to show that in presence of name variations, missing data and errors, the precision of matchings is very low. In their work, they witnessed different entities having very similar and in many cases identical bibliographic attributes. Li et al. [32] considered historical profiles from different unreliable resources and proposed a source-aware temporal matching algorithm for the identity resolution task. Balaji et al. [4], first, discussed the currently used blocking methods in data cleaning phase of entity resolution task and then, proposed a new ensemble blocking method to combine the results of different blocking methods. Kong et al. [15] proposed an unsupervised approach, called EMAN, to match entities across two or more heterogeneous data sources using locality sensitive hashing.

To improve the accuracy of identity resolution, researchers have exploited the information on social relationships in the network. Considering the neighbor nodes in a social network can significantly improve the precision of the matchings. For instance, Bartunov et al. [6] combined the usage of profile attributes and social relationships to identify all references to an entity. Buccafurry et al. [13] also used the common neighbors of two references to predict location of the *me* links that connect two references to the same entity. Rahmani et al. [43] used random-walk agents to count the number of paths in the network between two similar references which pass similar neighbor nodes. Existence of such paths corresponds to a high probability that two references refer to the same entity. Bhattacharya and Getoor [7] proposed a relational clustering algorithm to use both the reference attributes and the connections among them.

Using the network relationships increases precision of the identity resolution algorithms; however can cause computational issues. For instance, Bartunov et al. [6] explained that they can apply their identity resolution algorithm in small subgraphs, and face a major challenge to scale it up to large social graphs. This type of limitation is studied in detail by Christen and Gayler [18]. They propose the use of inverted indexing techniques, as a common technique in information retrieval, for real-time, fast and scalable identification of entities. They show that inverted index approaches are up-to one hundred times faster than some of the traditional entity matching techniques.

In this work, similar to the work by Christen and Gayler [18] we address the identity resolution problem in the information retrieval framework. However, instead of matching the references we aim at matching the relationships between them. By choosing the relationships as the targets of the identity matching problem, we achieve a low computationally complexity and still preserve a high accuracy.

2.2. Relation extraction

A vast amount of information available to us comes in form of the unstructured data, which can not be efficiently linked to the other sources of information in its original form [33,58]. Therefore,

most of the identity resolution approaches are not applicable in multi source applications which have both structured and unstructured data coexisting. It is important to develop techniques to convert the unstructured data to structured form by extracting the named entities and existing relations. For this, the unstructured data, which is in form of text, needs to be segmented into meaningful chunks of strings [16], and in turn the chunks which correspond to named entities should be identified (i.e., NER [36,45,51,55]). In the second step, the relations between these entities should be extracted (i.e., the topic of relation extraction [19,29,60]).

Using supervised learning for extracting information from a text imposes many restrictions such as the human effort needed to prepare the training set, and being domain specific. In order to solve these restrictions for text segmentation, Agichtein and Ganti [1] proposed the use of existing structured data in the data warehouse to learn an automatic segmentation system called CRAM. A very similar approach was used by Borkar et al. [11] in developing a tool called DATAMOLD which was able to automatically segment text. Unsupervised NER has also been well researched as in [21,28,37], where important information of the text can be extracted in a recursive manner.

However, solving the relation extraction in the unstructured data, which is the main focus of this paper, is more challenging than the text segmentation and NER. We divide the existing work on relation extraction, which is proposed to overcome the restrictions of supervised relation extraction, into three groups: **G1**) iterative relation extraction based on frequent named entities; **G2**) relation extraction based on frequent patterns, and **G3**) relation extraction based on distant supervision (the proposed approach in this paper is in this group).

In **G1**, starting from a small seed of entity pairs, with known relations, all occurrences of the pairs in data are found, which allows for extracting the patterns for those relations. These patterns help in finding more entity pairs and in turn the entity pairs increase the number of retrieved patterns. Iteratively, a large set of entity pairs and patterns to detect them can be obtained. The work of Brin [12] and the Snowball system [2] are examples of this approach.

In **G2**, the frequent patterns seen in the database provide a relational dataset. This dataset is then used by clustering algorithms to extract the informative patterns. For instance, Shinyama and Sekinea [52] proposed a *Preemptive* information extraction approach, in which first, the basic patterns which frequently appear in a text were extracted. Then a clustering technique was used to cluster the sets of similar patterns. In another attempt, Banko et al. [5] and Yates et al. [59] introduced the OpenIE (Open Information Extraction) approach and the TextRunner as a scalable implementation of OpenIE, in which first a learner automatically labels a sample corpus and then the labeled data is used to train a Naive Bayes classifier. By applying the classifier on large amounts of data many relational tuples can be extracted.

While both **G1** and **G2** have very useful real-world applications, they rely on abundance of data and frequent occurrences of entities and relation patterns, respectively. Therefore, such approaches have limitations in certain databases where the entities are not reported frequently and the relation patterns have high variety. In order to resolve this limitation, **G3** suggests to use a second source of information for extracting the relations. This second source which contains a large collection of entity tuples with known relations helps in finding large sets of patterns in the unstructured data corresponding to known relations. Work of Mintz et al. [34], Nguyen and Moschitti [38] and Riedel et al. [47] are in **G3**.

The work proposed in this paper contributes to **G3** by introducing REDS; it uses a knowledge repository of structured form for predicting the relations in the original unstructured data. In order to increase the precision and recall of the relation extraction technique it uses the concept of identity resolution to disambiguate the entities at the time of linking. Compared to the existing work in **G3**, REDS takes into account the possible spelling errors and name alternatives. The use of fingerprints for building an inverted index for the knowledge repository makes the searching process very fast and scalable.

Table 1
Brief description of notations used in our formal problem definition

Notation	Description
$\mathcal{U}$	Unstructured dataset
$\mathcal{K}$	External knowledge repository
d	Each document d in unstructured dataset
d.text	Text representation of each document $d \in \mathcal{U}$
d.refs	Set of references that are mentioned in d.text
r	Each reference $r \in d$.refs is a string representing a person name
d.rels	Set of relations that are existed in d.text
rel	Each relation *rel* $\in d$.rels is defined over a set of references

3. Preliminaries

In this section, we first formally define the problem at hand, and then we introduce the datasets that are used in different experiments throughout this paper.

3.1. Problem definition

Consider an unstructured dataset $\mathcal{U}$. Each document $d \in \mathcal{U}$ is represented by a text d.text, where a set of references d.refs are mentioned in d.text. Each reference $r \in d$.refs is a string representing a person name. Also a set of relations[1] d.rels exist in d.text. Each relation *rel* $\in d$.rels is defined over a set of references such that *rel*.refs $\subset d$.refs, it has a type *rel*.type and a pointer *rel*.cid to the certificate (i.e., the unstructured text document) it is extracted from.

Additionally, there exists a knowledge repository $\mathcal{K}$ that consists of a set of relations. Each relation *rel* $\in \mathcal{K}$ is defined over a set of references *rel*.refs, has a type *rel*.type and a pointer *rel*.cid to the certificate it is extracted from. Table 1 summarizes the notations used in our formal problem definition.

We assume that $\mathcal{U}$ and $\mathcal{K}$ are sparse datasets (i.e., each real entity is referred in $\mathcal{U}$ and $\mathcal{K}$ for a limited number of times). Besides, we assume name variations and spelling errors exists in both datasets. We define our problem as following.

Problem: Given a knowledge repository $\mathcal{K}$, a given document $d \in \mathcal{U}$ and a set of references $S_r = \{r_1, r_2, \ldots, r_m\}$ that are mentioned in consecutive order in d.text, find type of relation *rel* $\in d$ with *rel*.refs $= S_r$, and assess its confidence level.

In order to solve this problem, we propose a novel method for interconnecting the two datasets $\mathcal{U}$ and $\mathcal{K}$. To this end, we define the relation fingerprints that make it possible to directly compare the candidate relations of $\mathcal{U}$ with the explicit relations in $\mathcal{K}$.

3.2. Data setup

The genealogical dataset used in this paper consists of two different corpora: 1) the corpus of civil registers in form of structured data, and 2) the corpus of notarial acts in form of unstructured data. These corpora are provided by the Brabant Historical Information Center (BHIC).

Corpus of civil registers: this corpus contains three main types of certificates, namely "Birth", "Death" and "Marriage" certificates. Table 2 lists the content features for each certificate type. As shown

[1] In this paper the term *relation* refers to the way in which two or more references are connected (e.g., marriage relation, or co-occurrence relation) and is different from the concepts of relations in relational databases.

Table 2
Features of each certificate type in corpus of civil registers

Birth certificate	FIRSTNAME, LASTNAME, GENDER, BIRTHDATE, BIRTHPLACE, FATHERFIRSTNAME, FATHERLASTNAME, MOTHERFIRSTNAME, MOTHERLASTNAME
Death certificate	FIRSTNAME, LASTNAME, GENDER, BIRTHDATE, BIRTHPLACE, DEATHDATE, DEATHPLACE, FATHERFIRSTNAME, FATHERLASTNAME, MOTHERFIRSTNAME, MOTHERLASTNAME
Marriage certificate	GROOMFIRSTNAME, GROOMLASTNAME, GROOMAGE, BRIDEFIRSTNAME, BRIDELASTNAME, BRIDEAGE, GROOMFATHERFIRSTNAME, GROOMFATHERLASTNAME, GROOMMOTHERFIRSTNAME, GROOMMOTHERLASTNAME, BRIDEFATHERFIRSTNAME, BRIDEFATHERLASTNAME, BRIDEMOTHERFIRSTNAME, BRIDEMOTHERLASTNAME

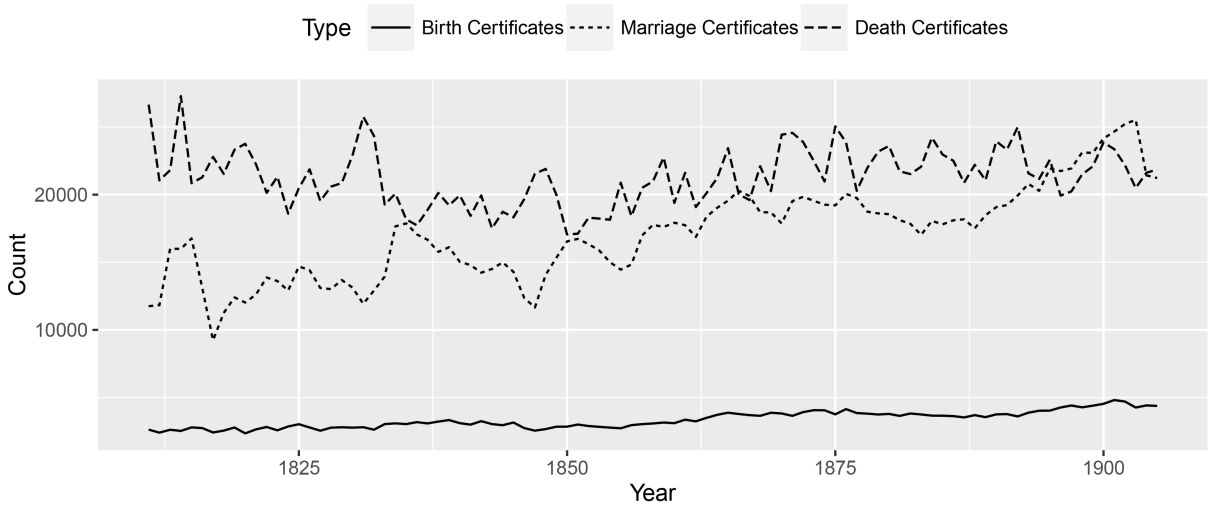

Fig. 1. Number of civil registers per year of issue.

in Table 2, Birth certificates include three individual references (i.e., child, father and mother). The Death certificates include four individual references (i.e., deceased, father, mother and relative of deceased). Finally, the Marriage certificates include six references (i.e., groom, bride and parents of each).

This corpus has been very dynamic in the sense that BHIC is continuously digitizing the handwritten archives by using the help of many volunteers. For instance the number of digitized documents has increased from 1,600,000 documents in 2012 to more than 2,000,000 documents in 2015. This amount is increasing everyday.

Figure 1 shows the number of digitized certificates per year for the most recent version of data in 2015. As can be seen in this figure, the amount of birth certificates has been less than number of marriage and death certificates. One of the reasons is the policies of BHIC and the fact that the regions for which marriage and death certificates are turned into structured data spans a larger area than the ones corresponding to birth certificates.

We use a Relational database model [20] to integrate and persist the three discussed certificate types considering Entity, Referential and Domain integrity constraints [25]. We choose the Relational database model since this model is widely used, easy to apply and is highly maintainable. [25] discusses in details the Relational database model and its advantages over other databases models.

Corpus of notarial acts: This corpus contains free text documents from various categories such as property transfers, inheritance reports, and loan declarations.

This dataset contains 226,751 records related the the period 1458 to 1900. The record contain written

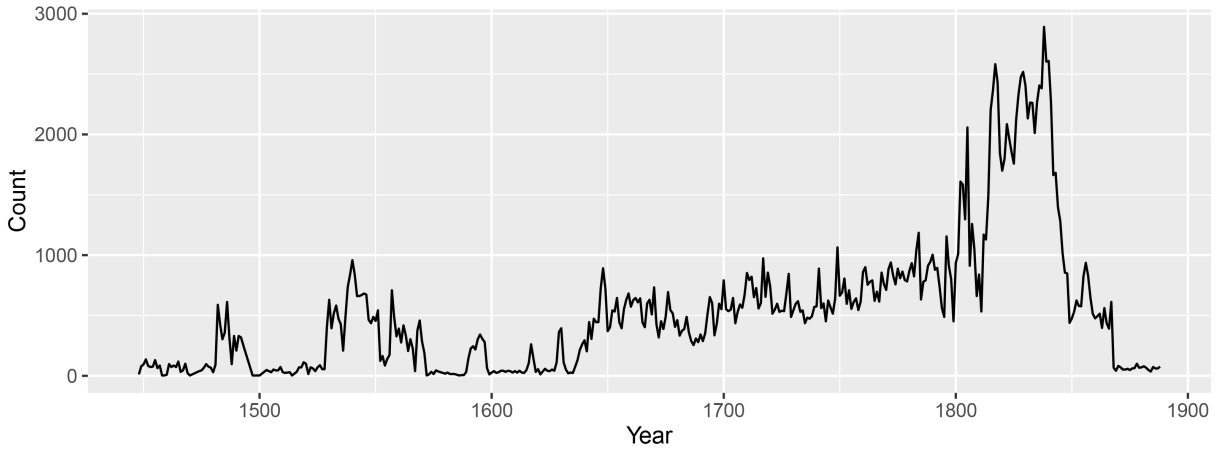

Fig. 2. Number of notarial acts per year of issue.

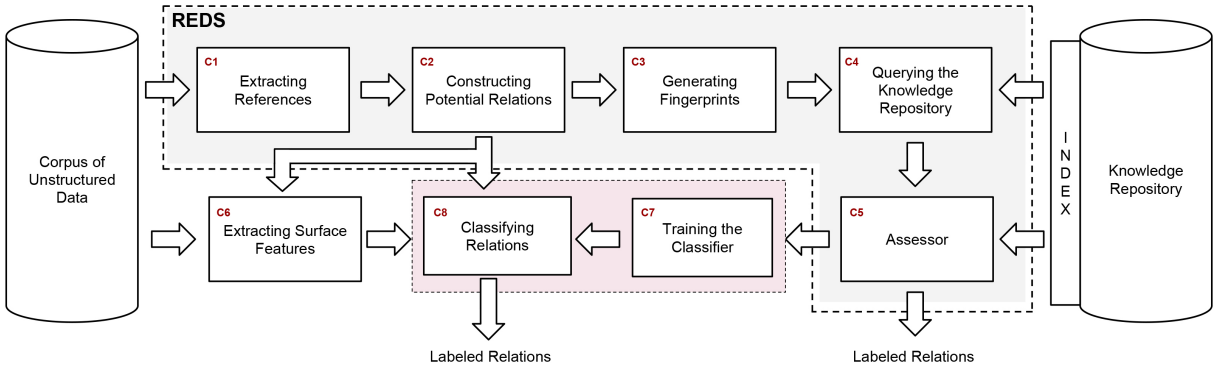

Fig. 3. The REDS components used for extracting relations from the unstructured data.

narration of facts drawn up by civil-law notaries, notary public or an alderman (Schepen in Dutch) authenticated by signature and official seal. Each record contains the main text of narration, place and date of issue and some other details.

Figure 2 shows the number of notarial acts per year. 86,000 notarial acts (i.e., 37% of all notarial acts) are issued after 1800 and overlap with the year of issue in civil registers.

4. The REDS method

Figure 3 illustrates the inputs, outputs and internal components, C1 to C8, of the proposed relation extraction approach. As its input, REDS has access to a corpus of unstructured data, on the left side of Fig. 3 and a knowledge repository, on the right side of Fig. 3. For an arbitrary text document chosen form the unstructured data corpus, references are extracted in C1 and potential relations are constructed in C2, followed by fingerprint generation in C3. Then, in C4, the knowledge repository is queried for each fingerprint. The result of this query can be used to detect the relations in text after being assessed by the assessor component C5, and also in combination with surface features of the unstructured data that are extracted in C6, it can be fed into a classifier for training purposes handled in C7. In C8 the classifier can be used to extract relations by using surface features of the text.

Next, we formally define each of these components C1 to C8 and use examples from the MiSS[2] dataset for better understanding of each component.

4.1. Indexing the knowledge repository

We first describe how to build an inverted index for the knowledge repository $\mathcal{K}$. Please note that the knowledge repository by itself acts as a *forward* index. For each relation in $\mathcal{K}$ we know which references appear in the relation. However, REDS needs an *inverted* index $\mathcal{I}$ that provides a list of references and for each reference the relations it appears in. For creating an inverted index, we parse the knowledge repository and while parsing we use the condensed representation of references, described below, as the index terms.

A person full name might contain prefixes before the given name or family name, middle names and other additional components. Each of these components are subject to change in presence of uncertainties and name variations. Therefore we use the notion of condensed representation of a reference that eliminates the unnecessary information of a full name.

Definition 1. Let r be a reference. The condensed representation c_r is a single string that is the concatenation of the given name and family name of r after lowercasing every word and filtering out the prefixes.

Depending on the quality of data and the domain of use, the condensed representation of a reference in Definition 1 can be further customized. For instance, each given name or family name can be standardized using a list of standard names. However, such customizations are beyond the scope of this paper; the interested reader can refer to the work of [9,22].

For each condensed representation, we store all the relation ids that the corresponding references appear in. Also, as we need to keep one additional information for each relation, the type of relation is stored too; this will help us with extracting the type of relations at the query time.

Therefore, to build the inverted index, for each relation from the knowledge repository we do the following. For each reference in the relation, we compute the condensed name of the reference. Then, we build the inverted index for the relation by adding the condensed name to the index $\mathcal{I}$ as a key and the tuple of relation id and relation type as the value. Apparently, if a key with the same name exists the tuple of relation id and relation type will be appended to the existing value.

Example: Table 3 shows an example of three civil registers. The knowledge repository and inverted index corresponding to these civil registers are shown in Table 4, in which the "husband-wife" relations are listed and indexed in $\mathcal{I}$.

4.2. Extracting references (C1)

Consider a text document $d \in \mathcal{U}$. REDS takes the raw text in d.text and splits it into words using a tokenizer. Next, it tags each token by its part-of-speech tag. As we're interested in named entities that refer to individuals' names, a combination of shape features (i.e., uppercase, titlecase, or lowercase) and lexical features (e.g., whole word, prefix/suffix) are used to extract the given names and family names, name prefixes and locations. Then it detects multi-token chunks that refer to a person full name. These chunks are added to the set of references d.refs.

[2]Mining Social Structures from Genealogical Data, part of the CATCH program funded by the Netherlands Organization for Scientific Research (NWO).

Table 3
Three civil certificates each containing "husband-wife" and "parent-child" relations. Two family names "Cornelesen" and "Cornelessen" are variations of the same family name. Some extra information such as place and date are not shown

Certificate 1	Certificate 2	Certificate 3
cid: 1	**cid**: 2	**cid**: 3
type: Marriage certificate	**type**: Marriage certificate	**type**: Death certificate
Groom: Gerardus Willems	**Groom**: Hendrikus Cornelesen	**Deceased**: Maria Cornelesen
Bride: Geertuij Cornelesen	**Bride**: Maria Meerwijck	**Father of deceased**: Cornelis Cornelesen
Father of Groom: Willem Willems	**Father of Groom**: Cornelis Cornelesen	**Mother of deceased**: Johanna Gijsbers
Mother of Groom: Geertruij Hagen	**Mother of Groom**: Johanna Gijsbers	**Spouse of deceased**: –
Father of Bride: Cornelis Cornelessen	**Father of Bride**: Wilhelmus Meerwijck	
Mother of Bride: Johanna Gijsbers	**Mother of Bride**: Anna Camphoven	

Table 4
A knowledge repository consisting of 7 "husband-wife" relations (in short 'HW') shown in dictionary format. In total 11 keys are extracted for the inverted index $\mathcal{I}$. By using the inverted index the references and the relations they appear in can be quickly retrieved

Relations in $\mathcal{K}$	Inverted index $\mathcal{I}$
rel_1 = {refs: {"Gerardus Willems", "Geertuij Cornelesen"}, type: 'HW', cid: 1}	"Gerardus Willems" → {[rel_1, 'HW']}
rel_2 = {refs: {"Willem Willems", "Geertruij Hagen"}, type: 'HW', cid: 1}	"Geertuij Cornelesen" → {[rel_1, 'HW']}
rel_3 = {refs: {"Cornelis Cornelessen", "Johanna Gijsbers"}, type: 'HW', cid: 1}	"Willem Willems" → { [rel_2, 'HW']}
rel_4 = {refs: {"Hendrikus Cornelesen", "Maria Meerwijck"}, type: 'HW', cid: 2}	"Geertruij Hagen" → { [rel_2, 'HW']}
rel_5 = {refs: {"Cornelis Cornelesen", "Johanna Gijsbers"}, type: 'HW', cid: 2}	"Cornelis Cornelessen" → { [rel_3, 'HW']}
rel_6 = {refs: {"Wilhelmus Meerwijck", "Anna Camphoven"}, type: 'HW', cid: 2}	"Johanna Gijsbers" → {[rel_3, 'HW'], [rel_5, 'HW'], [rel_7, 'HW']}
rel_7 = {refs: {"Cornelis Cornelesen", "Johanna Gijsbers"}, type: 'HW', cid: 3}	"Hendrikus Cornelesen" → { [rel_4, 'HW']}
	"Maria Meerwijck" → {[rel_4, 'HW']}
	"Cornelis Cornelesen" → { [rel_5, 'HW'], [rel_7, 'HW']}
	"Wilhelmus Meerwijck" → {[rel_6, 'HW']}
	"Anna Camphoven" → {[rel_6, 'HW']}

Example: Let d be a document with d.text given as following[3].

> "Johana Gijsbers weduwe Cornelius Cornelesen wonende te Erp heeft verkocht aan Hendrik Geert van der Steen land te Erp in de Melvert sectie A. 89 en 90."

This text can be broken up into words by using punctuations and whitespaces into 27 simple tokens, as shown below.

> Johana_(t1) Gijsbers_(t2) weduwe_(t3) Cornelius_(t4) Cornelesen_(t5) wonende_(t6) te_(t7) Erp_(t8) heeft_(t9) verkocht_(t10) aan_(t11) Hendrik_(t12) Geert_(t13) van_(t14) der_(t15) Steen_(t16) land_(t17) te_(t18) Erp_(t19) in_(t20) de_(t21) Melvert_(t22) sectie_(t23) A._(t24) 89_(t25) en_(t26) 90_(t27)

We introduce the part of sentence tags (GFN) for Given and Family Names, (FNP) for Family Name Prefixes, (LOC) for Locations, (LOP) for Location Prefixes and (UNK) for words with Unknown part of sentence. A Person Reference chucker (PR-Chunker) can detect the person references in the text by searching for the chunks that correspond to a person references, by mainly following the sequence of (GFN) and (FNP) tags. As a full name contains at least two words as the given and family names, no

[3]The translation of this Dutch text in English would be: *Johana Gijsbers widow of Cornelius Cornelesen living in Erp has sold to Hendrik Geert van der Steen a land in Erp in the Melvert sections A. 89 and 90.*

singular (GFN) tag can be accepted as a PR-chunk. Therefore, a full name is considered as a sequence of (GFN) tags or two of such sequences connected via one or more (FNP) tags.

Therefore d.text can be tokenized as following.

> Johana_(GFN) Gijsbers_(GFN) weduwe_(UNK) Cornelius_(GFN) Cornelesen_(GFN) wonende _(UNK) te_(LOP) Erp_(LOC) heeft_(UNK) verkocht_(UNK) aan_(UNK) Hendrik_(GFN) Geert_(GFN) van_(FNP) der_(FNP) Steen_(GFN) land_(LOP) te_(LOP) Erp_(LOC) in_ (UNK) de_(FNP) Melvert_(UNK) sectie_(UNK) A._(UNK) 89_(UNK) en_(UNK) 90_(UNK)

4.3. Constructing potential references (C2)

For each subset of references that are in consecutive order of each other a potential relation *rel* can be considered. Depending on the application at hand, relations can consist of two, three or more references in consecutive order.

Example: In analysis of MiSS dataset the focus is very often on "husband-wife" relations that consist of two references. We assume that every two consecutive person references, in the unstructured data, potentially have relations with each other. Therefore, for each unstructured document $d \in \mathcal{U}$ with m references $r_1, r_2, \ldots, r_m$, we generate $m-1$ pairs of references as $\textit{rel}_1 = (r_1, r_2)$, $\textit{rel}_2 = (r_2, r_3), \ldots,$ $\textit{rel}_{m-1} = (r_{m-1}, r_m)$.

The PR-Chunks in this example are $r_1 =$ "Johana Gijsbers", $r_2 =$ "Cornelius Cornelesen", and $r_3 =$ "Hendrik Geert van der Steen". Therefore, the reference pairs lead to two candidate relations $\textit{rel}_1$.refs = {"Johana Gijsbers", "Cornelius Cornelesen"} and $\textit{rel}_2$.refs = {"Cornelius Cornelesen", "Hendrik Geert van der Steen"}.

4.4. Generating fingerprints (C3)

Using Definition 1, we define the fingerprint of a relation *rel* as following.

Definition 2. Let *rel* be a relation over a set of references *rel*.refs. The fingerprint

$$\begin{aligned}\mathcal{F} &= \texttt{GenerateFingerprint}(\textit{rel}) \\ &= \left\{c_r \middle| r \in \textit{rel}.\text{refs}\right\}\end{aligned} \tag{1}$$

is defined as the set of condensed representation of each reference $r \in$ *rel*.refs.

Example: Using Definitions 1 and 2, REDS generates the following two fingerprints $\mathcal{F}_1 =$ {"Johana Gijsbers", "Cornelius Cornelesen"} and $\mathcal{F}_2 =$ {"Cornelius Cornelesen", "Hendrik Steen"}.

4.5. Querying the knowledge repository (C4)

REDS needs to answer two query types: 1) the exact matching queries: These queries contain some strings, and we want to find all the relations which contain references with condensed name equal to one of those strings. 2) the Levenshtein matching queries. These queries contain some strings and an integer a distance threshold $L_d \geqslant 0$. To answer these queries we have to find every relation which contains a reference for which the Levenshtein distance between the condensed name of the reference and one of the strings doesn't exceed L_d. The former type of query is a simpler case of the latter type when $L_d = 0$. Therefore, from now on we focus on the latter type of query. In the Levenshtein matching queries we have to search the index list $\mathcal{I}$ for the *lexical Levenshtein neighbors* of the query string. To this end, we use the scalable and fast *Levenshtein automata* proposed by [50]. If the Levenshtein automata finds

the Levenshtein neighbors of the query term, then the tuples of relation ids and relation types will be retrieved otherwise an empty set will be returned. We define a function for finding the Levenshtein neighbors of a string as following.

Definition 3. Let s be a string and $\mathcal{I}$ be the index list of $\mathcal{K}$. By using a Levenshtein automata we search for every $i \in \mathcal{I}$ where the Levenshtein distance between s and i doesn't exceed L_d. We define the following function.

$$\mathcal{N} = \texttt{GetLevenshteinNeighbors}(s, \mathcal{I}, L_d) \tag{2}$$

where $\mathcal{N}$ is a list of tuples including the relation ids and relation types.

4.6. Assessor (C5)

The Assessor component uses the statistics from data to measure the confidence level of the relation and also predicts its type by using the indexed knowledge repository $\mathcal{K}$. Following comes some definitions we need to assess the relation *rel* using its fingerprint $\mathcal{F}$.

Definition 4. Let $\mathcal{F} = \{s_1, s_2, \ldots, s_n\}$ be a relation fingerprint and $L_D \geqslant 0$ be the maximum acceptable Levenshtein distance for Levenshtein neighbors. Let $\mathcal{N}_i$ be the set of relations that contain the Levenshtein neighbors of s_i

$$\mathcal{N}_i = \texttt{GetLevenshteinNeighbors}(s_i, \mathcal{I}, L_d)$$

then we define $\texttt{Support}(\{s_i\})$ to be the number of occurrences of Levenshtein neighbors of s_i in $\mathcal{I}$ as

$$\texttt{Support}(\{s_i\}) = |\mathcal{N}_i|.$$

Let $\mathbf{N}_c = \bigcap_{i=1,2,\ldots,n} \mathcal{N}_i$ be the common tuples among all $\mathcal{N}_i$s, then we define $\texttt{Support}(s_1, s_2, \ldots, s_n)$ which is the number of those relations that every $s_1, s_2, \ldots, s_n$ has a levenshtein neighbor in it as

$$\texttt{Support}(\{s_1, s_2, \ldots, s_n\}) = \left|\mathbf{N}_c\right|.$$

We use the support of fingerprint elements and the set of common levenshtein neighbors in Definition 4 to compute the confidence level and predict the type of a relation. These two will be defined as following.

Definition 5. Let $\mathcal{F} = \{s_1, s_2, \ldots, s_n\}$ be a fingerprint relation. Then confidence level of $\mathcal{F}$ is defined as

$$\texttt{GetConfidenceLevel}(\mathcal{F}) = \frac{\texttt{Support}(\{s_1, s_2, \ldots, s_n\})}{\texttt{Support}(\{s_1\}) + \texttt{Support}(\{s_2\}) + \cdots + \texttt{Support}(\{s_n\})}$$

Let $\mathbf{N}_c$ be the set of tuples of relation id and relation type according to Definition 4. Let $\mathbf{N}_c$.type be a list of all relation types seen in this $\mathbf{N}_c$. Then we use $\mathcal{F}$ to define the predicted type as

$$\texttt{GetPredictedType}(\mathcal{F}) = \texttt{MostFreq}(\mathbf{N}_c.\text{type})$$

where the $\texttt{MostFreq}(\textit{list})$ function, returns one of the elements that is most frequent in the *list*.

We define the threshold level θ such that the relation *rel* is considered to be true if $\texttt{GetConfidenceLevel}(\mathcal{F}) > \theta$.

Algorithm 1 describes how REDS approach can be implemented to extract relations in an unstructured dataset $\mathcal{U}$.

Algorithm 1: Extraction of relations by using distant supervision. REDS generates the fingerprint of a candidate relation and queries the index $\mathcal{I}$ of the knowledge base $\mathcal{K}$ for this fingerprint. Based on the query answer if the relation is valid REDS measures the confidence level and predicts the type of the candidate relation. REDS returns sets of valid and invalid relations.

input : The unstructured dataset $\mathcal{U}$
input : The knowledge repository $\mathcal{K}$
output: Set of valid relations $Rel_{+}^{\mathcal{U}}$ extracted in $\mathcal{U}$
output: Set of invalid relations $Rel_{-}^{\mathcal{U}}$ extracted in $\mathcal{U}$

Build the inverted index $\mathcal{I}$ for $\mathcal{K}$
Initialize the threshold of the confidence level θ
Initialize the Levenshtein distance threshold L_d
$Rel_{+}^{\mathcal{U}} := \emptyset$
$Rel_{-}^{\mathcal{U}} := \emptyset$
foreach d **in** $\mathcal{U}$ **do**
  Extract the set of references d.refs
  Construct the set of potential relations d.rels
  foreach $rel \in d.rels$ **do**
    $\mathcal{F}$:= `GetFingerprint` (rel)
    rel.confidence := `GetConfidenceLevel`$(\mathcal{F})$
    if $rel.confidence > \theta$ **then**
      rel.type := `GetPredictedType`$(\mathcal{F})$
      insert tuple rel to $Rel_{+}^{\mathcal{U}}$
    else
      rel.type := $null$
      insert tuple rel to $Rel_{-}^{\mathcal{U}}$
    end
  end
end
return $Rel_{+}^{\mathcal{U}}$,$Rel_{-}^{\mathcal{U}}$

Example: Having an inverted index built for the knowledge repository, we can resolve the stream of queries that are generated to predict the relations in the unstructured data. For each fingerprint $\mathcal{F}$ generated from a relation in a notarial act we query the civil registers for the elements in the fingerprint allowing for a Levenshtein distance of maximum $L_d = 2$. We compute the confidence score of the relation by using Definition 5. For example, consider the three references extracted from the text document r_1 = "Johana Gijsbers", r_2 = "Cornelius Cornelesen", and r_3 = "Hendrik Geert van der Steen". Their condensed names are s_1 = "Johana Gijsbers", s_2 = "Cornelius Cornelesen", and s_3 = "Hendrik Steen" and their Levenshtein neighbors appear 18, 8 and 9 times in $\mathcal{I}$, respectively. Besides, the fingerprint $\mathcal{F}_1$ = {"Johana Gijsbers", "Cornelius Cornelesen"} appears 7 times (including the 3 times seen in the index of Table 4 in rel_3, rel_5, rel_7) in $\mathcal{I}$. The fingerprint $\mathcal{F}_2$ = {"Cornelius Cornelesen", "Hendrik Steen"} doesn't appear in $\mathcal{I}$. Therefore based on Definition 5, we have

$\texttt{Support}(s_1) = 18$ $\texttt{Support}(s_2) = 8$ $\texttt{Support}(s_3) = 9$
$\texttt{Support}(s_1, s_2) = 7$ $\texttt{Support}(s_2, s_3) = 0$
$\texttt{GetConfidenceLevel}(\mathcal{F}_1) = 0.33$ $\texttt{GetConfidenceLevel}(\mathcal{F}_2) = 0$

4.7. Extracting surface features (C6)

For generating the features for the training set we use the surface features of each document $d \in \mathcal{U}$ as described in following.

Definition 6. Let *rel* be a candidate relation defined over the m references $r_1, r_2, \ldots, r_m$. The feature vector

$$V_F = \texttt{ExtractSurfaceFeatures}(\textit{rel})$$

returns the t words before r_1, the words between every two consecutive references and t words after the last reference r_m, where t is an arbitrary number. Additionally, V_F includes the length of each set of words.

Example: We use the extracted relations by REDS to build a training set for classification of new potential relations as either positive (i.e., a "husband-wife" relation) or negative (i.e., no relation extracted). First, every relation fingerprint $\mathcal{F}$ with $\texttt{GetConfidenceLevel}(\mathcal{F}) > 0$ (i.e., $\theta = 0$) is labeled as a Positive instance and every relation fingerprint $\mathcal{F}$ with $\texttt{GetConfidenceLevel}(\mathcal{F}) = 0$ is labeled as a Negative instance. Second, we extract the surface features V_F: Using Definition 6 for each relation consisting of two references, V_F includes W_l which is the set of $t = 5$ words before the first reference, W_c which is the set of the words in between the two references, and W_r which is the set of $t = 5$ word after the second reference and the length of each one. The list of the words before and after each reference are cut when reaching another reference.

4.8. Training the classifier (C7) and classifying relations (C8)

REDS assesses the validity of potential relations in the unstructured data $\mathcal{U}$, and exports two sets of valid $Rel_+^{\mathcal{U}}$ and invalid relations $Rel_-^{\mathcal{U}}$. These two sets provide a training set which we can use to find the discriminative patterns of text capable of detecting the relations in absence of distant supervision. To this end, we label the set $Rel_+^{\mathcal{U}}$ as Positive instance and the remaining relations in $Rel_-^{\mathcal{U}}$ as the Negative instances. We assume the number of the False Positives (set of relations we discovered wrongly in the unstructured data) and False negatives (set of true relations in unstructured data that are not discovered by REDS) to be limited and not affect the discovery of discriminative patterns.

Using Definition 6 and the outputs of Algorithm 1 we can apply a proper machine learning method on the training instances to train classifier DT. DT can then predict the type of a relation based on the surface features of text. Once DT is trained, it can be applied on the text documents to extract relation with no need to distant supervision.

Example: Among the off-the-shelf machine learning methods, we apply the Decision Tree classifier since the patterns discovered by this method are easily interpretable by the domain experts [44]. The decision tree is built top-down from the very top node by using ID3 algorithm [41,42].[4] In each iteration, the information gain of each attribute of the feature set is computed and the attribute with the largest information gain is chosen to split the dataset. Once the decision tree is constructed it can be used to extract informative patterns for predicting the class label of a new feature set.

We use all the 10,000 Positive instances and then we select, randomly, 10,000 of the Negative instances to build a training set. We extract the surface features for each relation instance, and train a decision tree classier for predicting new relations. Figure 4 shows the decision tree with termination criterion of 12 leaves (12 leaves are chosen to keep the decision tree visualizable and interpretable. In practice a much larger decision tree can be built).

[4]The C4.5 algorithm, an extension of the ID3 algorithm, can also be used for this purpose. However, for this application we don't expect any significant change in the outcome.

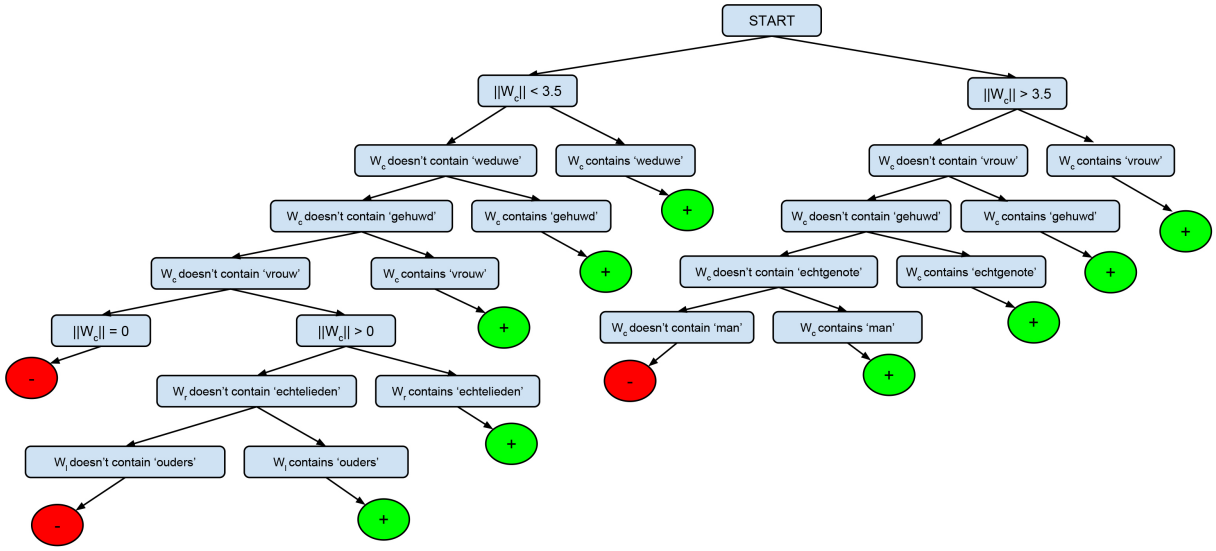

Fig. 4. A Decision tree built based on the ID3 algorithm to minimize the entropy. This tree generates 12 patterns for labeling a relation as Positive for a "husband-wife" relation or Negative otherwise. The discovered patterns show that in addition to the contents of W_l, W_c and W_r, the length of the text between two references $||W_c||$ plays an important role in extracting relations.

Every path from root to leaf represents a single pattern. This tree generates 12 patterns for labeling a relation as Positive (leading to green leaf) for a husband-wife relation or Negative (leading to red leaf) otherwise. For example, left most path indicates that if $||W_c|| < 3.5$, W_c does not contain any of "weduwe", "gehuwd", "vrouw" terms and finally, $||W_c|| = 0$ then, our decision tree predicts label Negative for the examined relation. From the other side, right most path indicates that if $||W_c|| > 3.5$ and W_c contains "vrouw", the prdicted label is Possitive.

Algorithm 2 shows how DT can be used to find the relations for which due to absence of evidence in $\mathcal{K}$, Algorithm 1 is incapable of detecting.

5. Empirical results

In this section, we report the results of implementing REDS on the dataset introduced in Subsection 3.2.

5.1. Effectiveness of fingerprints

First, we study the effectiveness of using proposed fingerprints by looking at data statistics. Consider a real person as entity e. In the civil registers, we expect references to e for his/her own birth, marriage and death certificates (3 times) and also birth, marriage and death of his/her children (3 times for each children). The death certificate of his/her spouse might refer to e as well (1 time for each marriage).

However, as no ground truth exists for this dataset, we can just make an estimation of number of times e is referred to. For an entity who has married once in his/her life and has three children we expect to find 13 references in the civil registers. Let r_i be one of the references referring to e with the condensed name s_i. We expect $\texttt{support}(\{s_i\}) \approx 13$. Also, considering the spouse of e called e', we expect e' to be referred in every civil register where e is referred to except for the birth of e. Let r'_i be a reference to e'

Algorithm 2: Using the DT classifier to extract relations in the unstructured data $\mathcal{U}$. DT which is trained based on outputs of Algorithm 1, uses the discriminative patterns of text to extract new relations between references.

input : The unstructured dataset $\mathcal{U}$
input : The trained classifier DT
output: Set of valid relations $Rel_{+}^{\mathcal{U}}$ extracted in $\mathcal{U}$
output: Set of invalid relations $Rel_{-}^{\mathcal{U}}$ extracted in $\mathcal{U}$

$Rel_{+}^{\mathcal{U}} := \emptyset$
$Rel_{-}^{\mathcal{U}} := \emptyset$
foreach d **in** $\mathcal{U}$ **do**
  extract the set of references d.refs
  use d.refs to construct the set of potential relations d.rels
  foreach *rel* $\in$ *d.rels* **do**
    $V_F :=$ `ExtractSurfaceFeatures`(*rel*)
    $classLabel :=$ DT (V_F)
    if *classLabel is valid* **then**
      rel.type = classLabel
      insert tuple *rel* to $Rel_{+}^{\mathcal{U}}$
    else
      rel.type = *null*
      insert tuple *rel* to $Rel_{-}^{\mathcal{U}}$
    end
  end
end
return $Rel_{+}^{\mathcal{U}}$, $Rel_{-}^{\mathcal{U}}$

with the condensed name s_i', thus we expect that $\texttt{support}(\{s_i, s_i'\}) \approx 12$. Considering the uncertainties in the dataset and missing data we expect to see less values for the support of single and paired condensed names in most of the cases. To study the two types of support functions, we look at the distribution of supports for single condensed names and compare it with the support distributions for pair of condensed names. First, we define the Complementary Cumulative Distribution Function (CCDF).

$$P_c(k) = \big|\{\alpha | \texttt{support}(\alpha) \geqslant k\}\big|.$$

The CCDF, $P_c(k)$, shows how often $\texttt{support}(\alpha)$ is above k.

We choose 1000 "husband-wife" relations randomly from the knowledge repository at hand (i.e., the relations in the civil registers). For each relation the fingerprint consisting of a pair of condensed names is extracted. Then, for each condensed name which appears in the fingerprint, we compute its support. We compute the supports for four different Levenshtein distance thresholds $L_d = 0, 1, 2, 3$. Figure 5a shows the CCDF for each threshold in logarithmic scale. We can see that by increasing the distance threshold the maximum support significantly increases. While 36 identical condensed names exist for $L_d = 0$, more than 100 condensed names are Levenshtein neighbors of each other for $L_d = 3$. This is in contrast to our expectations of having about 13 references to an entity. Figure 5b shows the CCDF for the support of fingerprints (i.e., pair of condensed names). By increasing the distance threshold the average support is increased however the maximum support is fixed around 12.

To further study the changes in maximum support value, for the Levenshtein distance threshold $L_d = 2$ we consider knowledge repositories with different sizes: 500, 800, 1000, 2000, 6000, 8000 and 10,000 relations. Figure 6 gives a comparison between the maximum support of single condensed names and paired condensed names for each of these knowledge repositories. In this figure, it is intuitively clear that by increasing the size of the knowledge repository the maximum number of Levenshtein neighbors of

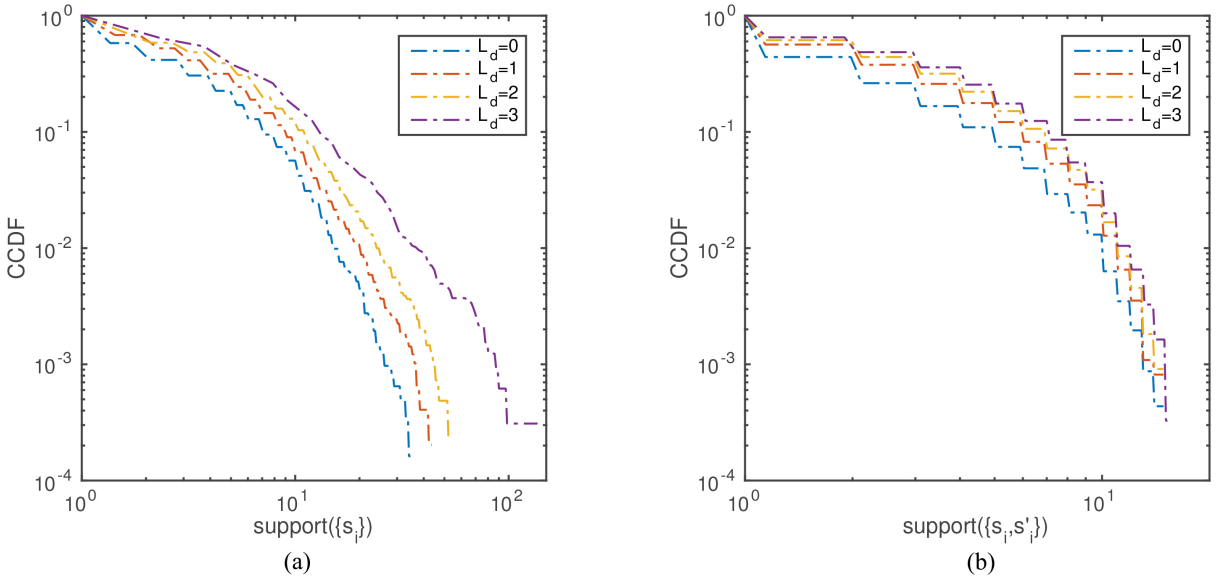

Fig. 5. The CCDF for the support of single and paired condensed names. In (a) by increasing the Levenshtein distance threshold L_d, the average support and maximum support significantly increase. However, in (b) while the average support increases, the maximum support is constant which shows the effectiveness of using fingerprints in identifying the real entities.

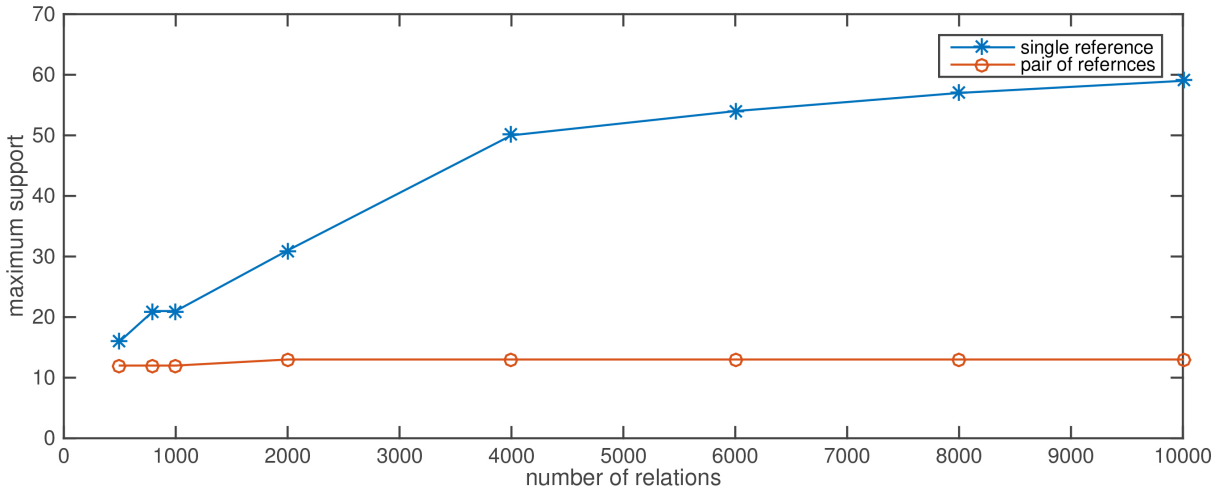

Fig. 6. The changes of maximum support of single and paired condensed names for knowledge repositories of different sizes. By increasing the size of knowledge repository the number of Levenshtein neighbors of single condensed names increases while the maximum number of paired condensed names in the same Levenshtein neighborhood are very robust to changes in size of the knowledge repository.

single condensed names continuously increases. However, for the paired condensed names (i.e., fingerprints) this maximum value is very constant. Therefore, in a large knowledge repository each condensed name and its close Levenshtein neighbors might refer to various entities, while the pair of these condensed names clearly refer to a pair of entities and are very robust to the size of knowledge repository. This result will be confirmed by showing a high precision of REDS in the next subsections.

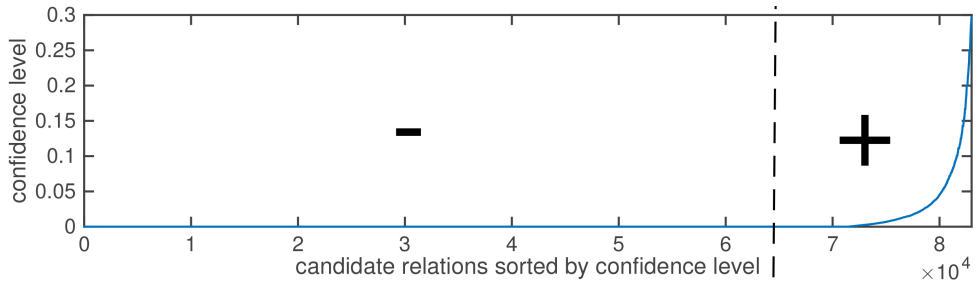

Fig. 7. Distribution of confidence scores described based on Definition 5. The reference pairs with zero score are labeled as Negative (i.e., left side of the dashed line) and the reference pairs with non-zero score are labeled as Positive (i.e., right side of the dashed line). The number of Negative instances is almost 6 times larger than the number of Positive instances.

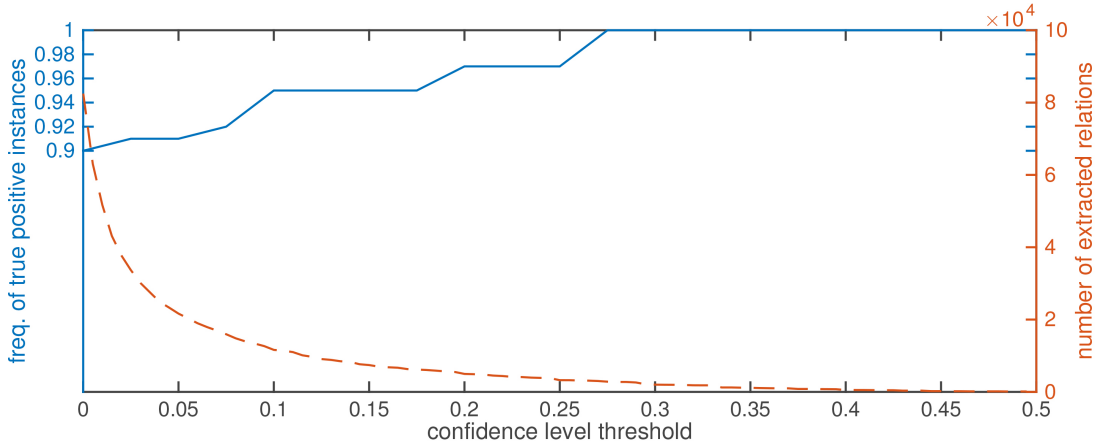

Fig. 8. Effect of confidence level threshold on precision of extracted relations and overall number of extracted relations. By increasing the threshold of confidence level θ, the number of extracted relations decreases drastically and the number of true positive instances compared to the false positives slightly increases. With threshold $\theta = 0$ REDS extracts 80,000 relations with precision 0.90, while with $\theta = 0.5$ REDS extracts only 100 relations with precision 1.0.

5.2. *Generating and assessing the potential relations*

Following the NER approach discussed previously, we extract, on average 4.3 PR-chunks, from 20,000 notarial acts. For more than 14,000 of these notarial acts at least one candidate relation is generated. In total 83,000 relations are extracted, among which about 12,000 receive positive confidence level and are labeled as Positive instances and 71,000 are labeled as Negative. Figure 7 shows the distribution of scores for these 83,000 extracted reference pairs.

5.3. *Evaluation of extracted relations*

In order to evaluate the discovered relations by REDS, experts in historical information are asked to check the extracted household relations from notarial acts and decide whether the pair of references have a "husband-wife" relation with each other or not. Manual evaluation of the extracted relations is a very time-consuming process for the domain experts. Clearly it is not possible for the domain experts to evaluate all the 80,000 discovered relations, in an acceptable time. Thus we randomly select 1,000 positive instances, while the confidence level threshold $\theta = 0$ is chosen. Among these instances, experts

Table 5
Measuring the accuracy of REDS and DT for extracted relations from the unstructured text. #Ins., #TP, #FP, #TN and #FN are the number of evaluated, true positive, false positive, true negative and false negative instances, respectively

	#Ins.	#TP	#FP	#TN	#FN	Precision	Recall	F-measure
REDS	1000	896	104	–	–	0.90	–	–
DT on Test-set 1	700	205	45	415	35	0.82	0.85	0.89
DT on Test-set 2	700	158	52	458	32	0.75	0.79	0.88

find 896 true positives and 104 false positives; the precision is then 0.90. Due to absence of any ground truth for this dataset, the measurement of recall is a very difficult task (for discussion on challenges in measuring recall we refer to work of [24]). We repeated this evaluation for other confidence level thresholds $0 < \theta \leqslant 1$ and the changes of precision are shown in Fig. 8. Distribution of extracted relations with respect to the confidence level threshold is also illustrated in Fig. 8. According to Fig. 8 although by increasing the confidence level the precision increases, large portion of the extracted relations have very low but positive confidence level. This confirms that choosing $\theta = 0$ is a right choice for this application as it contains the most amount of true positive relations with an acceptable precision.

5.4. Evaluation of classifier relations

Next, we evaluate the decision tree DT. First, we apply DT to the **Test-set 1** which is the same dataset of notarial acts between 1800 to 1912 which REDS was applied on. As discussed in Section 3.2, this dataset overlap with the years of external knowledge repository and contains 86,000 notarial acts for that period. Second, more importantly, we apply DT to **Test-set 2** consisting the notarial acts between 1700 to 1800, which do not have any time overlap with the available knowledge repository, and can be assumed as a real test set. The **Test-set 2** contains 140,751 notarial acts (See Fig. 2). We select randomly 700 relations in each case including the negative instances. Table 5 gives the evaluation results, precision, recall and accuracy of REDS and DT. In this table, as the references in civil registers don't have overlaps with the notarial acts in Test-set 2 the REDS can not be applied on this test set.

6. Discussion

The analysis provided in the previous section, showed that REDS is a precise approach which extracts 90% of the relations correctly and has the potential to generate a training set for supervised learning of a classifier. The classifier can extract new relations with precision of 0.75 and recall of 0.79. In Section 5.3, we showed that in our application the confidence level acts as a binary assessor, such that for non-negative levels we can accept the validity of the relation. We consider this as a result of data sparsity; the entities that are mentioned in the text documents are referred to for a handful of times. Therefore, co-occurrence of two references in at least one relation of the knowledge repository is an indication of a high probability for the validity of the potential relation which includes the same two references. We see this behavior of REDS as an advantage that allows Algorithm 1 to provide a fine-grained filtering of results such that there is no need to the ranking of results.

In Section 5.4, we also saw about 25% False Positives as the result of applying the trained decision tree on a test set. Here we mention two main reasons for prediction of invalid relations: First, NER module can make mistakes in forming the PR-Chunks, and in turn the queries on fingerprints return invalid results. Second and more importantly, due to the dynamics of the data in course of centuries, the

text grammar, word collections and topics of the documents change; the patterns that the decision tree is trained to detect do not appear in the text documents in the test set, thus the classifier is not capable of predicting correct relations. This is another proof for the importance of using distant supervision in detecting relations.

The proposed REDS approach doesn't make any assumption on quality of the knowledge repository. Although, a deduplicated and cleaned knowledge repository might improve the accuracy of REDS, the use of relation fingerprints provides a fast dedupliation of the knowledge repository on the fly. Thus, building an inverted index for the knowledge repository is the only preprocessing requirement of REDS.

7. Conclusions

In this paper, we addressed the problem of extracting relations from unstructured data. We discussed how the existing relation extraction approaches had limitations such as dependency on manually built training sets, being language dependent or requiring the abundance of data. Considering these limitations, we used a knowledge repository to provide distant supervision to a relation extraction engine. Additionally, unstructured datasets usually suffer from uncertainties such as spelling errors, name variations etc. To resolve these uncertainties, first, we introduced relationship fingerprint as a noise-tolerant condensed representation of a relationship between two or more individuals and then, we used Levenshtein distance to match the fingerprints in the external repository. By using an inverted index in the knowledge repository the proposed method in this paper, called REDS, is very fast and scalable. The proposed approach was implemented on a case study of genealogy, in which a collection of structured civil registers were used as the knowledge repository. The relations in civil registers provided distant supervision to the relation extraction in the unstructured notarial acts. In order to evaluate the precision of the proposed approach, the domain experts, manually, evaluated the outcomes and confirmed the high precision of this technique (precision $= 0.90$). Additionally, we used the relations extracted by REDS to discover the informative patterns between references in the notary acts. We used these patterns, extracted by a decision tree, to discover relations in the test set of unstructured notarial acts where no knowledge repository was available for. The proposed method succeeded to predict novel relations with F-measure of 0.88.

Acknowledgments

This research has been supported under the NWO CATCH program in the MISS project (project no. 640.005.003). The authors are grateful to the BHIC center for the support in data gathering, evaluations and direction.

References

[1] E. Agichtein and V. Ganti, Mining reference tables for automatic text segmentation, in: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, 2004, pp. 20–29.
[2] E. Agichtein and L. Gravano, Snowball: Extracting relations from large plain-text collections, in: *Proceedings of the Fifth ACM Conference on Digital Libraries*, ACM, 2000, pp. 85–94.
[3] N. Bach and S. Badaskar, A review of relation extraction, Literature review for Language and Statistics II, 2007.
[4] J. Balaji, F. Javed, M. Kejriwal, C. Min, S. Sander and O. Ozturk, An ensemble blocking scheme for entity resolution of large and sparse datasets, CoRR, abs/1609.06265, 2016.

[5] M. Banko, M.J. Cafarella, S. Soderland, M. Broadhead and O. Etzioni, Open information extraction for the web, in: *IJCAI*, Vol. 7, 2007, pp. 2670–2676.
[6] S. Bartunov, A. Korshunov, S.-T. Park, W. Ryu and H. Lee, Joint link-attribute user identity resolution in online social networks, in: *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, Workshop on Social Network Mining and Analysis*, ACM, 2012.
[7] I. Bhattacharya and L. Getoor, Collective entity resolution in relational data, *ACM Transactions on Knowledge Discovery from Data (TKDD)* **1**(1) (2007), 5.
[8] L. Bilge, T. Strufe, D. Balzarotti and E. Kirda, All your contacts are belong to us: automated identity theft attacks on social networks, in: *Proceedings of the 18th International Conference on World Wide Web*, ACM, 2009, pp. 551–560.
[9] G. Bloothooft, Corpus-based name standardization, *History and Computing* **6**(3) (1994), 153–167.
[10] S.R. Boal, Identity resolution for consumers with shared credentials, July 17 2013. US Patent App. 13/944,486.
[11] V. Borkar, K. Deshmukh and S. Sarawagi, Automatic segmentation of text into structured records, in: *ACM SIGMOD Record*, ACM, Vol. 30, 2001, pp. 175–186.
[12] S. Brin, Extracting patterns and relations from the world wide web, in: *The World Wide Web and Databases*, Springer, 1999, pp. 172–183.
[13] F. Buccafurri, G. Lax, A. Nocera and D. Ursino, Discovering missing me edges across social networks, Information Sciences, 2015.
[14] R.C. Bunescu and R.J. Mooney, A shortest path dependency kernel for relation extraction, in: *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2005, pp. 724–731.
[15] K. C., G. M., X. C., Q. W. and Z. A., Entity matching across multiple heterogeneous data sources, in: *Database Systems for Advanced Applications. DASFAA 2016. Lecture Notes in Computer Science*.
[16] F.Y. Choi, Advances in domain independent linear text segmentation, in: *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference*, Association for Computational Linguistics, 2000, pp. 26–33.
[17] P. Christen, Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection, Springer Science & Business Media, 2012.
[18] P. Christen and R. Gayler, Towards scalable real-time entity resolution using a similarity-aware inverted index approach, in: *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, Australian Computer Society, Inc., 2008, pp. 51–60.
[19] C. Christodoulopoulos and A. Mittal, Simple large-scale relation extraction from unstructured text, CoRR, abs/1803.09091, 2018.
[20] E.F. Codd, The Relational Model for Database Management: Version 2, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.
[21] A. Cucchiarelli and P. Velardi, Unsupervised named entity recognition using syntactic and semantic contextual evidence, *Computational Linguistics* **27**(1) (2001), 123–131.
[22] J. Efremova, B. Ranjbar-Sahraei and T. Calders, A hybrid disambiguation measure for inaccurate cultural heritage data, in: *The 8th Workshop on LaTeCH*, 2014, pp. 47–55.
[23] J. Efremova, B. Ranjbar-Sahraei, F.A. Oliehoek, T. Calders and K. Tuyls, A baseline method for genealogical entity resolution, in: *Workshop on Population Reconstruction*, 2014.
[24] J. Efremova, B. Ranjbar-Sahraei, H. Rahmani, F. Oliehoek, T. Calders, K. Tuyls and G. Weiss, Multi-source entity resolution for genealogical data, in: *Population Reconstruction*, Springer International Publishing, 2015, pp. 129–154.
[25] R.A. Elmasri and S.B. Navathe, Fundamentals of Database Systems, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1999.
[26] O. Etzioni, M. Banko, S. Soderland and D.S. Weld, Open information extraction from the web, *Communications of the ACM* **51**(12) (2008), 68–74.
[27] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D.S. Weld and A. Yates, Web-scale information extraction in knowitall:(preliminary results), in: *Proceedings of the 13th international Conference on World Wide Web*, ACM, 2004, pp. 100–110.
[28] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D.S. Weld and A. Yates, Unsupervised named-entity extraction from the web: An experimental study, *Artificial Intelligence* **165**(1) (2005), 91–134.
[29] Z. GuoDong, S. Jian, Z. Jie and Z. Min, Exploring various knowledge in relation extraction, in: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 2005, pp. 427–434.
[30] N. Kambhatla, Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations, in: *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, Morristown, NJ, USA, Association for Computational Linguistics, 2004, p. 22.

[31] H. Köpcke, A. Thor and E. Rahm, Evaluation of entity resolution approaches on real-world match problems, *Proceedings of the VLDB Endowment* **3**(1-2) (2010), 484–493.
[32] F. Li, M.L. Lee and W. Hsu, Profiling entities over time in the presence of unreliable sources, *IEEE Transactions on Knowledge and Data Engineering* **29**(7) (July 2017), 1522–1535.
[33] A. McCallum, Information extraction: Distilling structured data from unstructured text, *Queue* **3**(9) (2005), 48–57.
[34] M. Mintz, S. Bills, R. Snow and D. Jurafsky, Distant supervision for relation extraction without labeled data, in: *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, Association for Computational Linguistics, 2009, pp. 1003–1011.
[35] M. Motoyama and G. Varghese, I seek you: searching and matching individuals in social networks, in: *Proceedings of the Eleventh International Workshop on Web Information and Data Management*, ACM, 2009, pp. 67–75.
[36] D. Nadeau and S. Sekine, A survey of named entity recognition and classification, *Lingvisticae Investigationes* **30**(1) (2007), 3–26.
[37] D. Nadeau, P. Turney and S. Matwin, Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity, 2006.
[38] T.-V.T. Nguyen and A. Moschitti, End-to-end relation extraction using distant supervision from external semantic repositories, in: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers-Volume 2*, Association for Computational Linguistics, 2011, pp. 277–282.
[39] H. Purohit, A. Dow, O. Alonso, L. Duan and K. Haas, User taglines: Alternative presentations of expertise and interest in social media, in: *Social Informatics (SocialInformatics), 2012 International Conference on*, IEEE, 2012, pp. 236–243.
[40] H. Purohit, P.A. Dow, L. Duan and O. Alonso, Derivation and presentation of expertise summaries and interests for users, Mar. 12 2013. US Patent App. 13/797,914.
[41] J.R. Quinlan, Induction of decision trees, *Machine Learning* **1**(1) (1986), 81–106.
[42] J.R. Quinlan, C4. 5: programs for machine learning, Elsevier, 2014.
[43] H. Rahmani, B. Ranjbar-Sahraei, G. Weiss and K. Tuyls, Entity resolution in disjoint graphs: an application on genealogical data, *Intelligent Data Analysis* **20**(2) (2016).
[44] C.A. Ratanamahatana and D. Gunopulos, Feature selection for the naive bayesian classifier using decision trees, *Applied Artificial Intelligence* **17**(5-6) (2003), 475–487.
[45] L. Ratinov and D. Roth, Design challenges and misconceptions in named entity recognition, in: *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, Association for Computational Linguistics, 2009, pp. 147–155.
[46] D. Ravichandran and E. Hovy, Learning surface text patterns for a question answering system, in: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 2002, pp. 41–47.
[47] S. Riedel, L. Yao and A. McCallum, Modeling relations and their mentions without labeled text, in: *Machine Learning and Knowledge Discovery in Databases*, Springer, 2010, pp. 148–163.
[48] S. Sarawagi, Information extraction, *Foundations and Trends in Databases* **1**(3) (2008), 261–377.
[49] M. Schraagen and H.J. Hoogeboom, Predicting record linkage potential in a family reconstruction graph, in: *23th Benelux Conference on Artificial Intelligence (BNAIC2011)*, 2011, pp. 199–206.
[50] K.U. Schulz and S. Mihov, Fast string correction with levenshtein automata, *International Journal on Document Analysis and Recognition* **5**(1) (2002), 67–85.
[51] Y. Shen, H. Yun, Z.C. Lipton, Y. Kronrod and A. Anandkumar, Deep active learning for named entity recognition, CoRR, abs/1707.05928, 2017.
[52] Y. Shinyama and S. Sekine, Preemptive information extraction using unrestricted relation discovery, in: *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, Association for Computational Linguistics, 2006, pp. 304–311.
[53] J.M. Stibel and A.B. Stibel, Method and system for directly targeting and blasting messages to automatically identified entities on social media, June 24 2014. US Patent 8,762,473.
[54] M. Surdeanu and M. Ciaramita, Robust Information Extraction with Perceptrons, in: *Proceedings of the NIST 2007 Automatic Content Extraction Workshop (ACE07)*, 2007.
[55] E.F. Tjong Kim Sang and F. De Meulder, Introduction to the conll-2003 shared task: Language-independent named entity recognition, in: *Proceedings of the Seventh Conference on Natural language learning at HLT-NAACL 2003-Volume 4*, Association for Computational Linguistics, 2003, pp. 142–147.
[56] J. Vosecky, D. Hong and V.Y. Shen, User identification across multiple social networks, in: *Networked Digital Technologies, 2009. NDT'09. First International Conference on*, IEEE, 2009, pp. 360–365.
[57] D.J. Watts, P.S. Dodds and M.E. Newman, Identity and search in social networus patent appks, *Science* **296**(5571) (2002), 1302–1305.
[58] W.E. Winkler, Overview of record linkage and current research directions, in: *Bureau of the Census*, Citeseer, 2006.

[59] A. Yates, M. Cafarella, M. Banko, O. Etzioni, M. Broadhead and S. Soderland, Textrunner: open information extraction on the web, in: *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, Association for Computational Linguistics, 2007, pp. 25–26.

[60] D. Zelenko, C. Aone and A. Richardella, Kernel methods for relation extraction, *The Journal of Machine Learning Research* **3** (2003), 1083–1106.

[61] S. Zhao and R. Grishman, Extracting relations with integrated information using kernel methods, in: *ACL*, The Association for Computer Linguistics, 2005.