IJDAR manuscript No.
(will be inserted by the editor)

Domain-Specific Entity Extraction from Noisy, Unstructured Data Using Ontology-Guided Search

Sergey Bratus · Anna Rumshisky
Alexy Khrabrov · Rajenda Magar
Paul Thompson

Received: date / Accepted: date

Abstract Domain-specific knowledge is often recorded by experts in the form of unstructured text. For example, in the medical domain, clinical notes from electronic health records contain a wealth of information. Similar practices are found in other domains. The challenge we discuss in this paper is how to identify and extract part names from technicians repair notes, a noisy unstructured text data source from General Motors' archives of solved vehicle repair problems, with the goal to develop a robust and dynamic reasoning system to be used as a repair adviser by service technicians.

In the present work, we discuss two approaches to this problem. We present an algorithm for ontology-guided entity disambiguation that uses existing knowledge sources such as domain-specific taxonomies and other structured data. We illustrate its use in automotive domain, using GM parts ontology and the unit structure of repair manuals text to build context models, which are then used to disambiguate mentions of part-related entities in the text. We also describe extraction of part names with a small amount of annotated data using Hidden Markov Models (HMM) with shrinkage, achieving an f-score of approximately 80%. Next we used linear-chain Conditional Random Fields (CRF) in order to model observation dependencies present in the repair notes. Using CRF did not lead to improved performance, but a slight improvement over the HMM results was obtained by using a weighted combination of the HMM and CRF models.

Keywords Text Analysis · Language Models · Information Extraction · Ontology-Guided Search

Sergey Bratus, Rajenda Magar, Paul Thompson
Dept. of Computer Science, Dartmouth College, Hanover, NH USA
E-mail: {sergey.bratus,rajenda.magar,paul.thompson}@dartmouth.edu

Alexy Khrabrov
Thayer School of Engineering, Dartmouth College, Hanover, NH USA
E-mail: alexy.khrabrov@dartmouth.edu

Anna Rumshisky
Dept. of Computer Science, Brandeis University, Waltham, MA USA
E-mail: arum@cs.brandeis.edu

1 Introduction

In many specialized areas, domain knowledge is often codified in the form of specialized lexicons and ontologies. At the same time, working domain experts often keep records of their actions in the form of unstructured notes. Using domain knowledge recorded in such form presents a serious problem, since annotating large amounts of unstructured text for machine learning algorithms with domain concepts and semantic types requires extensive human labor.

For example, such is the situation with clinical notes from the electronic patient records. Despite some recent efforts [12], there is a distinct lack of annotated corpora for unstructured medical text. At the same time, a number of lexicons and ontologies are available in the medical domain. For example, the National Library of Medicine maintains a Unified Medical Language System (UMLS) which provides both taxonomy in the form of network of semantic types, and metathesaurus defining and linking relevant concept structures; a SNOMED nomenclature of clinical terms is available from the College of American Pathologists.

A similar situation exists in many manufacturing companies which often maintain relevant ontologies and thesauri for their products. The challenge we discuss in this paper is how to identify and extract part names from technicians repair notes, a noisy unstructured text data source from General Motors' archives of solved vehicle repair problems, with the goal to develop a robust and dynamic reasoning system to be used as a repair adviser by service technicians.

This extraction problem is alleviated by the presence of a comprehensive and actively maintained ontology, which contains a lot of semantic and lexical clues to disambiguating the said notes. *We proceed from the assumption that the writer abbreviates (and the reader therefore interprets) the notes based on a shared conceptual context, and that the same context is expressed in the ontology.* This allows us to build and compare context models derived from both text units and the ontology, and base semantic disambiguation on an information-theoretic basis, as described in Section 3.

We use the term ontology loosely, to refer to what is used as a *de facto* tool for knowledge organization and communication structuring by a large body of engineers and other specialists. Strictly speaking, "terminology" or "controlled vocabulary" might be more fitting. However, in our choice of the term, we want to stress the fact that the data set is closely linked with (and in practice may be inseparable from) the actual organization of knowledge regarding the unit structure and composition of the objects. In other words, the role played by it in reality is closest to that of a comprehensive taxonomy encompassing a *de facto* ontology. The structured organizational knowledge expressed in this dataset should not be considered trivial because the very existence of this data is due to real communication needs of large bodies of domain experts and represents a systematic effort to facilitate their collaboration.

General Motors has numerous structured knowledge sources that are maintained for various purposes; in particular, a taxonomy of part names organized by functional subsumption, which we use as the ontology. The unstructured text data we use comes from GM's massive archives of service technicians' notes on solved repair problems. The overall goal of the project described in this paper is to use these archives and resources to develop robust, dynamic Textual Case-Based Reasoning (TCBR) systems [1, 5] that can be used as repair advisers by service technicians and engineers. A core challenge is in the taxonomic indexing of repair text archives, so that smart search, e.g., ontology-guided search (OGS), can be used to match the description of the symptoms

of a new problem with those of solved problems in the archive. Since classifying and disambiguating text are key elements of the indexing process, it includes a strong element of natural language processing (NLP) and information extraction. For example, "GAS CAP" and "FUEL CAP" are synonyms and should be classified together, but "GAS" by itself requires disambiguation, because it has three distinct meanings in the context of three distinct subsystems: the *powertrain* (as fuel for the engine), the *heating and cooling system* (as refrigerant for the air conditioning), and the *fuel cell power system*, where hydrogen gas is used to generate electricity. Clearly, two kinds of methods must be developed: (1) methods to classify text by locating key phrases on lexical taxonomies, and (2) methods to disambiguate the text by using context to determine which regions of the taxonomies are more likely to be most relevant.

The types of solved-problem archives we have in mind consist of a few structured attributes along with technicians' notes in free-form text. Existing collections of text annotated by parts of speech, such as the Penn TreeBank, are of little use in analyzing this kind of text. It has a specialized vocabulary and abbreviations. The text blocks are brief, and commonly they do not obey the rules of standard English spelling or grammar. Therefore, certain standard NLP techniques have not been fruitful, and domain-specific alternatives seem necessary. It needs to be taken into account that we do not have the resources to hand-annotate such text in the domain-specific ways required. Since the text consists of ungrammatical fragments with no sentence boundaries for each fragment, it is doubtful whether annotating parts of speech right away would be the right approach. For example, consider the following actual sample from our notes:

```
CUST STATES THE RIGHT SIDE OF
THE HOOD IS SITTING
HIGHER THENA FOUND RT SIDE
OF HOOD LOOSE TIGHTEN
HOOD BOLT
```

For a domain expert, this segment yields the following sentences:

```
1. CUST STATES THE RIGHT SIDE OF THE HOOD
   IS SITTING HIGHER
2. THENA FOUND RT SIDE OF HOOD LOOSE
3. TIGHTEN HOOD BOLT
```

There is a number of anomalies in this segment, in addition to the absence of sentence boundaries. Notice that unstructured notes in different domains carry their own peculiarities. Consider the following excerpt from a free-text note taken from an electronic medical record:

```
At 8:30am SBP 80 and dopamine gtt increased
to 9.5mg/kg. PA catheter placed under fluro
with opening # PA 70/38 PWP 40 with V waves
to 50, CVP 28 with V waves to 38, CO/CI/SVR
7.5/3.5/437
```

While sentensification may be less difficult, syntactic anomalies remain a problem, as does extensive use of domain-specific acronyms and abbreviations. It is clear, however, that the same kind of semantic abbreviation takes place, which depends on a *shared concept* system. By considering such examples, we have been led to consider

approaches that do not depend on standard grammar or spelling, but do exploit the domain-specific structures.

The challenge is to use the existing (and maintained) knowledge resources, in lieu of annotated data that needs to be created. For example, there are several structured lexicons of part names. (They are actually names of categories of parts, but we will refer to them as part names.) Indexing with part names may be sufficient for many smart search applications. In other words, if we can find solved problems involving the parts referenced in the new problem, then we have reduced the number of relevant "cases" significantly.

These various domain-specific knowledge resources have good coverage, but they lack the structure and precision needed for TCBR and OGS. Therefore, we have looked at techniques for refining these knowledge resources to make them more consistent, less redundant, and more powerfully structured. We have also developed robust indexing (and search) techniques that are not destabilized by noisy data.

We note that both our ontology and reference text samples are mission-critical and actively maintained for the specific domain. Hence whereas one can rely on these properties being true in the automotive and medical domains, in other domains where manuals and procedures might not be as mission critical, our methods may not yield the results as good as can be obtained in the domains at hand.

2 Overview

Knowledge resources created for day-to-day purposes in a company can often be adapted for decision support. Uschold noted that a company's glossary or thesaurus could be adapted to become a semantic net [11]. Any manufacturing company, such as General Motors, has numerous lists and lexicons related to the names associated with the design, engineering, manufacturing, and servicing of its products. Our key example are the lists of automotive parts that are referenced in servicing vehicles. We might have focused on lexicons for design, engineering, or manufacturing, in developing decision-support tools for those areas. An example for manufacturing is the GM Variation-Reduction Adviser [6, 7].

For knowledge to function as an asset in a modern corporation, it must be explicit and machine-readable. "Explicit" means that it is "written down," as opposed to carried in people's heads or derivable via data analysis. Knowledge must be machine-readable because the sheer volume of data makes human processing impossible. It is commonly said that 80% of corporate data exists as text: program notebooks; problem summaries in warranty records, technical-assistance center logs, customer surveys, and various other archives of records, logs, and diaries. Most of this text is machine-readable only in a limited sense. For example, key-word search can find items by exact matching, but the slightest differences in expression can cause an item to be missed. Ideally, one would like to ask questions such as "Has this issue come up before?" or "How many times has this issue come up before?" and get the answer by searching, cross-referencing, and comparing these various archives. Current tools and methods cannot do this.

Repair records are usually partly structured, with important information captured as free text. We want a TCBR system that accepts a "query" consisting of the symptoms of a problem and responds with a prioritized list of repair records that have similar symptoms. We want to use the knowledge structures that are available and maintained

by the manufacturer. This is a very rich source of already-available knowledge. We would like to avoid a significant knowledge-acquisition effort specifically for the TCBR application, e.g. annotating text for training ML methods.

An archive of text records of solved repair problems poses major challenges when matching it to the codified contextual knowledge tree. One can easily imagine using such an archive to avoid repeating time-consuming diagnostic and repair experiences, but in practice it is difficult to use. This is because of

(1) assumed contextual information that is not written into the record,
(2) the paraphrase problem, and
(3) ambiguous language.

Let us consider our model TCBR application in more detail: A technician with an unsolved repair problem is searching the Warranty Data Archive to determine if his unsolved problem or a similar problem has arisen before. The technician enters the symptoms and wants the records returned in the "most relevant symptom" order. After some analysis of such repair records, we have determined that most symptoms have the form: "PART NAME is broken."

We have focused our efforts on identifying and classifying the part-names references in a repair record. These part names have become our surrogate for symptoms. Part name extraction is indicative of much broader NLP problems posed by the text outlined above, such as the material context omitted from the text, paraphrases missing from the grammar, ambiguous language, and so on. We focus on this particular problem because it appears to deliver a highest payoff in the industrial scenarios.

In fact, extracting entity names, such as part names, is the first step towards understanding of natural text as it immediately enables business analytics and other statistical applications even at its simplest. It also significantly improves search of document collections and, therefore, location of relevant documents. For example, in the automotive domain, much of the business activity revolves around making, ordering, or replacing parts. In the medical domain, symptoms and diagnoses play a similar role.

The approach we imagine is to use information extraction to create a structured index of the blocks of text included with the repair records. We want to identify and classify part names in text. It is through the classification of part names that both indexing and similarity are defined. The indexing process would include cleaning the text, extracting the part names, and mapping the part names into existing taxonomies of part names. Thus, the values for the slots in the index would be taken from one or more existing parts taxonomies. Similarity would be defined from the taxonomic structures. A characteristic of the domain is that these part names and their structures are constantly changing. Thus, the system would require a certain robustness.

An important consideration for TCBR is that the objects of interest be structured into *isa* and *part-of* taxonomies (and perhaps other relations). This is because indexing text leads to a desire to generalize and specialize to solve the paraphrase problem and the disambiguation problem. One natural context for an object in a taxonomy is the path from the node where the object is named to the root, as well as the descendants of the node. If the same name is used in several nodes (e.g., "GAS"), then the choice of node is the disambiguation of the name, and matching the textual context with the taxonomic context is one way to choose the most relevant node.

The kinds of lexicons and lists we used were:

– T, a taxonomy of part name categories. The relations defined by the links has varying meaning, usually *isa* or *part-of*.

- a list of standard abbreviations,
- an engineering glossary,
- L, a list of labor code descriptions, and
- a mapping of the elements of L to the elements of T.

We note that the ontology provided to us represents the summary of a large organization's efforts in structuring communications between its units, and, as such, resembles a coding tree (in the information theory sense) rather than a codification of synonymy, hyper/hyponymy, substitutability and other semantic relations captured by a lexical KBs such as WordNet [14]. Specifically, our ontology is meant to capture context relationships between terms and concepts represented by its nodes in a way that a domain expert would rely on when communicating with a colleague. Thus it is obvious to us that our ontology and WordNet are products of two different processes, the former deriving from an organizational process designed specifically to produce the corpus at hand as a remedy for concrete communication issues, and the latter being a general and broad attempt to structure language norms not tailored to any specific project needs. Even though we cannot quantify the difference in the product of these two processes exactly, we observe much more focused and uniform grouping of our domain-specific ontologies, with a much more fixed set of relationships, mostly containment, reflected in the tree structure, vs. a more general network-like structure of the *synsets*.

While many authors, e.g., [13], [14], [15], have used natural language processing or machine learning techniques for various language processing tasks with biomedical texts, we are unaware of any approaches to lexical disambiguation based on information-theoretic mappings to taxonomies such as presented here. The following describes several examples of this research. Demner-Fushaman et al. [14] describe how to improve the accuracy of text categorization based on a thesaurus by eliminating highly ambiguous terms from the thesaurus. Chapman et al. [13] use a na ive Bayes classifier to categorize chief complaints from free text fields of medical records into seven early presentations of disease categories, or syndromes, for use in an electronic syndromic surveillance system. Bundschus et al. [15] compare the use of several topic modeling techniques for use in (semi-) automated generation of metadata annotation for PubMed abstracts. By contrast, our approach uses both new data structures and new algorithms to organize the domain knowledge at hand.

In Section 3, we describe the algorithm for ontology-guided entity disambiguation. In Section 4, we describe the overall OGS-based TCBR application pipeline. In Section 5, we discuss part name extraction, which supports the TCBR application, and present experimental results. Section 6 provides a discussion of these results, and Section 7 concludes.

3 Ontology-Guided Entity Disambiguation

In unstructured expert-generated text, domain-specific entities (e.g. parts in automotive domain; diagnoses, test results, therapy protocols in patient records) are referred to by inherently ambiguous short noun phrase mentions that are disambiguated by human readers based both on textual context and on their extra-textual domain knowledge. Automatic indexing and efficient searching of unstructured noisy text corpora require that this disambiguation be performed automatically so that part mentions are extracted and annotated with the respective part identifiers.

In each domain, this knowledge is partly expressed in domain-specific taxonomies and concept systems. For example, medical domain ontologies may classify entities as body parts, clinical findings, procedures and treatments, and so on. In the GM automotive domain, structural ontologies of automobile parts classify parts by functionality, as well as by systems, subsystems and assemblies. Our method uses lexical features derived from these ontologies, together with lexical features from the context surrounding the automotive text mentions, to perform disambiguation.

The main idea behind the algorithm we propose is the following. We need to disambiguate each part mention to a particular node in the ontology tree. The elements of the noun phrase *NP*comprising the part mention may occur in the names and descriptions associated with a given set of nodes in the ontology tree. We make the simplifying assumption that the target part mention will disambiguate to one of these *candidate nodes*. We term the set of candidate nodes, along with the paths from each node to the root, a *tree cut* induced by the part mention. We associate a probability distribution with a given *tree cut* and assign a probability to each leave by assuming equiprobable branching at each node.[1] Entropy of a *tree cut* is then computed as the entropy of the associated probability distribution.

For each candidate node, we obtain its *tree context* by assembling the lexical items from the names and descriptions of the nodes on its path to the root. We then look at the context in which the target part mention occurs, and compute a set-theoretic intersect of this context with the *tree context* of the candidate node. Each term in the intersect will occur only in some candidate nodes, thereby inducing a further cut on the tree cut corresponding to the target part mention. We compute an association score between the candidate node and the target part mention by weighting each term in the intersect by the overall reduction in the entropy it induces on the original tree cut. Thus, for example, the term "assembly" may not be very informative, but it may reduce the set of candidate nodes for a particular part mention to a single node. Each term is also weighted according to its distance from the target entity in text and the tree path distance from the candidate node. The part mention is then disambiguated to the node that maximizes this association score.

More formally, for an entity E (part mention) associated with a noun phrase NP, we define its context C as a collection of features derived from the unit(s) of text containing the NP(the sentence, the paragraph, or other structural units if defined; for instance, the headings of the manual section and chapter for the NPs in a manual). For each node N of the ontology tree T we likewise define a collection of features C', derived from the lexical contents of N and its ancestors on the path to the root of the tree, as well as of its siblings and additional lexical units attached to the nodes of the ontology tree T.

For all candidate nodes N in T, we compute the score $Q(E, C, N, C')$ and select the node N with the highest score. The function Q is based on information-theoretic measures associated with the lexical units in the tree T, based in their occurrence in the nodes throughout T. Essentially, the measure associated with a single unit (word or stem) expresses the *uncertainty* about the identity of the node N containing that unit once the unit is known. For example, a word or stem that occurs in the names or descriptions of many nodes throughout T has a higher measure of uncertainty than

[1] Ideally, corpus counts could be used to estimate probabilities at each node, but that requires making further assumptions, such as equal distribution of senses for each ambiguous word.

a word or stem that occurs only in a few leaves or branches of T. These uncertainty scores of participating units are then combined to form Q. The definition of function Q is discussed below.

In our initial experiments we also considered adding sibling-child and associated metadata nodes in the tree to the nodes' tree context. While anecdotally such inclusion with an appropriate weight coefficient appears helpful, – in particular in the situations where the creators of the ontology apparently counted on the human reader deriving the context from the words previously used in siblings as a form of abbreviation – we do not currently possess convincing measurements in favor of this hypothesis. We note this omission of lexical units that would be helpful for automated processing is a general human tendency – the same tendency to eliminate redundancy that our entire model aims to capture. Generally, we found the context gathered on the upward path toward the root of the tree discriminating enough.

3.1 Information-Theoretic Context Association Score

Let tokenization of a paragraph P produce a sequence of tokens $\{t_i\}$. If sentence boundaries are available, we further subdivide this sequence into groups by their respective sentences. Let $\{u_i\}$ be the sequence of lexical units derived from $\{t_i\}$ after tokenization. The lexical units are derived by a transformation according to a dictionary D of abbreviations and multi-token terms; some tokens are expanded to several lexical units, some are stemmed, the stem replacing the token in its place in the sequence, some multi-token groups collapsed to a single unit. This transformation from sequences of tokens to sequences of standardized lexical units will be referred to as $U : \{t_i\} \to \{u_i\}$.

For an entity E that spans lexical units $u_k, \ldots, u_{k+l}$, we define the *primary context* C_E as the collection of units $u_1, \ldots, u_{k-1}, u_{k+l+1}, \ldots$. If E occurs in structured text such as a manual, we also add to this primary context the units of text structurally related to P, such as headings under which P occurs. As an extension of this approach, we assign to each lexical unit u in C_E a *weight* depending on u's position relative to E (e.g., decreasing with the distance from u to E, counted within P in the number of lexical units separating them, and outside P in the number of structural elements separating the current element from P). Denote this weight $d(u, E)$.

For a candidate node N of T, define the *ontology context* C'_N of N as consisting of lexical units derived according to the rules of the transformation U from the names and descriptions attached with the ancestors and siblings of node N. Each unit u coming from a node N_u is associated with a weight based on the distance between N and N_u, counted in the number of nodes separating N and N_u on the path to the root of T, with a special distance value fixed for siblings of N. Denote this weight $d_T(N_u, N)$.

We define the score $Q(E, C_E, N, C'_N)$ as follows:

$$Q(E, C_E, N, C'_N) =$$

$$= \sum_{u \in \mathfrak{C}(E,N)} d(u, E) \cdot d_T(N_u, N) \cdot (H(T_E) - H((T_E)_u))$$

where $\mathfrak{C}(E, N)$ is the set of terms in the set-theoretic context intersect $\mathfrak{C}(E, N) = (U(E) \cup C_E) \cap C'_N$, T_W is the tree cut induced on T by the set of lexical units derived from the W, and $H(T_W)$ is the entropy [13] measure associated with it as described above.

We considered using other parts of speech and their relations, such as verb phrases, but our texts are noun-rich and verb-poor. Examining other syntactic constructs in addition to *NP*s is an interesting further extension of this work.

4 Matching Noun Phrases to XML Ontology

In this section, we describe the application of the above algorithm in a test setting. The ontology used for the task was derived from Vehicle Partitioning and Product Structure (VPPS) taxonomy maintained within GM. VPPS is a taxonomy of part names organized by functional subsumption, which contains roughly 5000 nodes. It is supplemented by a lexicon of part name categories (UPC-FNA) that contains 55,000 entries. The text data we used comes from the low-noise technical text from the GM car user manual, as well as the noisy repair notes.

4.1 Data Formatting

We received the original VPPS and UPC-FNA data as large spreadsheets in the MS Excel format, with the tree structure being represented by way of cell indentation. We found this representation hard to validate and maintain, especially because our planned approach was to continually enhance the ontology tree based on the merging of these sources with added annotation. Moreover, while validating the original spreadsheets, we found some apparent format discrepancies and errors (e.g., mis-attached nodes), most probably due to past maintenance edits, an outcome we wished to avoid.

The parts lists and ontologies being "living documents" actively maintained and exchanged by different groups of specialists, we also anticipated providing the data owners with tools for future validation and management that would not conflict with our added annotations. We therefore decided to represent, process, and maintain the underlying main dataset combining VPPS and UPC-FNA (which serves as input to all our algorithms) as a single XML document, validated both through the standard XML parsing mechanisms and by our own scripts, and processed according to the DOM model.

The obtained matches of part mentions against the ontology must be output for human consumption and verification. When developing the pipeline, we output such matches in a simple ASCII markup, which also makes it easy to use in further scripting. We describe our results and markup below. We also output supporting data as YAML, a data interchange format similar to XML but less verbose.

4.2 System Description

In the experiments with part mentions from the GM car manuals, ontology-guided search is currently performed for any text on the paragraph level. Noun phrases are extracted from the sentences comprising each paragraph, filtered against the unigrams encountered in the tree, their entropies computed and sorted, and the corresponding tree nodes are shown for each paragraph along with the phrases.

As a part of preprocessing the ontology, lexical units for abbreviation expansions were added as attributes to their respective nodes, so abbreviation expansion is taken

care of by the design of the ontology. The paragraphs boundaries are obtained via title headings in the original SGML documents. The titles for each text unit are preserved, parsed, and used as a part of text context for each part mention. Each paragraph is sentensified, and each sentence is then passed through an English parser – currently Charniak's parser – and the *NP*s are found with their $NN|JJ$ modifiers. Measuring parsing performance was not a part of this phase of the project, but checking them by hand found virtually no errors in the technical text of the manuals, likely due to the simple structure of the technical text at hand. The *NP*s are filtered against the unigrams list of all the words encountered in the ontology, thus forming the set of interesting, or sensible, phrases, which we will refer to as *mentions*.

Top-scoring ontology nodes are then identified for each *mention* found in a paragraph, and a list of *mentions* (relevant and/or identifiable *NP*s), with the corresponding top disambiguation candidates for each *NP*, is associated with each paragraph. The resulting paragraph summaries are stored in a machine-oriented, human-readable YAML format, available for immediate inspection and verification after each automatic pipeline run. The produced document is then represented as the list of paragraph and sentence summaries. The pipeline steps are described below in the order of their actual execution.

- *Text Extraction from SGML.* Text and headings are extracted from SGML of a manual chapter. The title blocks are collected hierarchically and output as a path, slash-separated, for each paragraph.

- *Sentence Boundary Detection and Parsing.* Sentences and their immediate paragraph heading are extracted and parsed using Charniak's parser. Paragraph titles are split by / and each part is parsed separately.

- *Noun Phrase Extraction.* We read parse trees from Charniak's parser such as:

  ```
  (S1 (S (NP (DT The) (VB scan) (NN tool))
  (VP (VBZ displays) (NP (NNP On/Off))) (. .)))
  ```

  – and extract noun phrases (NPs) with a noun head `NN` and nominal or adjectival modifiers (`NN|JJ`), and record them along with syntax labels and the location of origin – sentence and paragraph.

- *Entropy Computation and Sorting.* For each extracted NP, we compute the association score with each of the candidate nodes, and record the top-scoring nodes with the corresponding scores, along with the node's location in the ontology. Association scores are generated for unigrams and bigrams encountered in the ontology tree.

- *Text Markup and Output.* Finally, we mark up the original text with the top-entropy phrases and top nodes for them.

The final markup of the text looks as shown in Figure 1.

The original sentences are printed one per line, with `Sn:` prefix, where `n` is the sentence number. The paragraphs are delimited with the lines of the form:

```
@Par Pn: /Hierarchical/Title/Heading
```

For example:

```
@Par P1: /Engine/Engine Electrical/Diagnostic Information and Procedures/Scan Tool
Output Controls

S1: The engine control module (ECM) commands the generator OFF by removing the 5-volt
reference signal from the L-terminal of the voltage regulator when you select OFF.

@Mention 1 [voltage regulator] (NN NN) :S2 =2.585 (6)

@Node 1
Tree context: /vpps/vehicle/information & controls/driver information/clusters/regulator
integrated circuit voltage
Score: 3.462

@Node 2
Tree context: /vpps/vehicle/electrical function/charging & energy
storage/generator/generator internal component/terminal voltage regulator
Score: 2.177

@Node 3
Tree context: /vpps/vehicle/electrical function/charging & energy
storage/generator/generator internal component/plate voltage regulator
Score: 2.177
...
```

Fig. 1 Text markup sample for the technical manual.

`@Par P2: /Engine/Engine Electrical/Diagnostic Information and Procedures/Scan Tool Output Controls`

Only the paragraphs containing the *mentions* for which a match was found contain the appropriate markup. Each *mention* block starts with its number, `@Mention 1`, `@Mention 2`, etc., and consists of a *mention* header line, containing the *mention* phrase and stats, for example:

`@Mention 1 [voltage regulator] (NN NN) :S1 =2.585 (6)`

where the interpretations of each markup element are as follows:

`@Mention 1`	this is the first matched *mention* in the paragraph
`[regulator voltage]`	the actual *mention*
`(NN NN)`	POS tags corresponding to the words left-to-right
`:S1`	originating sentence
`=2.585`	entropy for this *mention* in the ontology tree, assuming a uniform distribution over the candidate nodes
`(6)`	number of leaf nodes in the tree cut for this *mention*

We then show the top matched nodes containing the *mention*, including their path names, showing for each ancestor starting at the root, the slash-separated node name (if available) or node type:

```
@Node 1
Tree context: /vpps/vehicle/information \& controls/driver information/clusters/
              regulator integrated circuit voltage
Score: 3.462
```

Repair notes are processed in the same manner, using similar filtering mechanisms. In the absence of the hierarchical title heading structure, text-only context before and after the target NP is used in disambiguation. As NPs are extracted from each sentence, we retain only the content words when querying the ontology (skipping determiners,

conjunctions, possessive markers, etc.). A sample markup of disambiguated part mention from a repair note is shown in Figure 2. Note that in this example, the target NP "wiring harness" is highly ambiguous and occurs in 1539 nodes in the ontology tree.

```
S1: DRIVER'S SEAT WILL NOT GO TO THE MEMORY POSITION WHEN KEY IS INSERTED . REPLACE
BRAKE PEDAL SENSOR OLH TO REMOVE AND REPLACE RIGHT FRONT SEAT CUSHION TO REPAIR WIRING
HARNESS AND REROUTE

@Mention 1 [wiring harness] (NN NN) :S1 =10.588 (1539)

@Node 1
Tree context: /vpps/vehicle/interior/seats/front seat/harness assembly: driver seat
suspension air supply harness assembly
Score: 0.901

@Node 2
Tree context: /vpps/vehicle/interior/seats/front seat/harness asm
Score: 0.838

@Node 3
Tree context: /vpps/vehicle/electrical function/power & signal distribution/wire
harnesses/front seat cushion heater wiring harness assembly
Score: 0.683
...
```

Fig. 2 Text markup for a sample repair note.

5 HMM/CRF Approach to Part Name Extraction from Noisy Text

First, let us reiterate what we mean by part names. We are interested in categories of parts, such as "OIL PAN." There are many kinds of "actual" oil pans, and each is identified by a unique name (or rather a combination of data, including a part number, supplier, date of manufacture, and other identifying information). We find it convenient in this paper to refer to OIL PAN and other part-name categories as "part names," but we are always referring to categories. We want to be able to identify part names in text. We want to classify the discovered part names by mapping them to T. In order to accomplish this, we need to consider which words and two-word phrases of the name are semantically "informative" and which are not. In this paper we focus on identifying part names. In the case that we have grammatical text, we would expect to be able to extract noun phrases (NPs) from the text using NLP techniques as a starting point for identifying part names. Then we could determine by attempting to map each NP to T whether the NP is a part name or some other object (e.g., CUSTOMER).

However, the structure of part names can help us identify them, even when the text is fairly ungrammatical. For concreteness, imagine that a technician is having trouble with the "left outside rear view mirror." Naturally, he might look for this exact string, but a verbatim in a warranty record might describe the "LEFT OUTSIDE REAR VIEW MIRROR" in different ways. The most obvious variants are created simply by omitting some of the qualifying adjectives, which is often done when the context is assumed. Further, vehicles have a bilateral symmetry, so that many parts come in a left (driver's side, etc.) version and a right (passenger's side, etc.) version. Sometimes

the difference will be important to the problem-solving potential of a warranty record, but often it is not.

The semantic head of the above phrase is "MIRROR," which in this case is the syntactic head. Mirrors in vehicles are either "rear view" or "vanity," but vanity mirrors can be only inside. Thus, technicians might neglect to add the qualifying phrase "rear view" if they have already notes "outside." It might not even make sense for the search to favor these kinds of qualifying adjectives, because such qualifiers are often omitted in technician's notes. Thus, while if they are present, they add to the information gain of the phrase, if they are not present, there is no information loss. In other words, it means nothing; it is a consequence of the "assumed context," which is so common in such notes. A hurried technician might simply have written "OUTSIDE MIRROR" knowing that vanity mirrors are never outside and the left-right distinction is not important for the particular write up. The essential structure of the NP is "MIRROR," which is essentially a concept class, along with enough qualifiers to distinguish it from all other mirrors. There is (at least one) taxonomic tree implicit in this analysis.

The training data for part name identification consists of repair notes where part names have been labeled. For the evaluation, we hand-labeled 1,000 randomly sampled repair notes. We divided the repair notes into two sets of 500 each, one for evaluation during development and the other for final testing. We also used approximately 30,000 part name phrases and a lexicon of 1,600 part name words collected from these phrases during training. For testing we used five-fold cross validation.

The structure of our HMM consists of "target" and "non-target" states, s_i. Part names correspond to the "target" state of the HMM. The non-target states are Start, Prefix, Suffix, Background, and End. The Prefix state corresponds to a fixed-length sequence of words before the target words. Similarly, the Suffix state corresponds to a fixed-length sequence of words following the target words. The remaining words are thought of as being emitted by the Background state. The probabilities are estimated as ratios of counts. The transition probability $P(s_j|s_i)$ is calculated as the total number of (s_i, s_j) label pairs divided by the total number of s_i labels in the training data. The emission probability $P(w|s_i)$ is calculated as the number of w labeled as s_i divided by the total number of s_i labels. One of the issues we have had to face is getting sufficient labeled data for training. This has had implications for the effectiveness of HMM structure used to model our data [2, 3]. Our HMM is complex with as many states as possible and is able to capture the intricate structure of the data in use; however, it results in poor (high variance) parameter estimation because of the sparseness of training data. In contrast, simpler models with fewer states, while giving robust parameter estimates, are not expressive enough for data modeling. In order to strike a balance, we used a statistical smoothing technique called "shrinkage" to combine the estimates from these models of differing complexity. Freitag and McCallum [3] report positive results using "shrinkage" to perform information extraction using HMMs. Some states from a complex model are shrunk to a common state to form a new HMM structure – hence the term shrinkage. To further improve parameter estimates, states from the new HMM can be further shrunk to form another HMM with even fewer states, thus forming a shrinkage hierarchy. In our case, we shrunk the Prefix and Suffix states to a common Context state. We then employed another level of shrinkage, in which all the states were shrunk to a single state.

The recall and precision scores from the five-fold cross validation along with the F-score are presented below. Table 1 shows results for the fully expressive model alone, as well as for the optimal shrinkage mixture of three HMM models.

	fully expressive model	optimal shrinkage mixture
Average Recall	12.26	81.64
Average Precision	79.85	79.45
F-Score	21.26	80.53

Table 1 Five fold cross-validation results for (1) the fully expressive model and (2) optimal shrinkage mixture, with a context width of two.

The fully expressive model has poor recall though precision is good. This indicates that the model by itself is not sufficient to cover all part names. The F-score of the fully expressive model is low. On the other hand, using shrinkage with optimal mixture weights improves recall values substantially while maintaining high precision. The substantial improvement in recall indicates that the shrinkage mixture helps to smooth parameter estimates to expand coverage of part names not handled by the fully expressive model alone.

We have investigated the effectiveness of using HMMs with shrinkage for part name extraction and found that HMMs do well modeling the repair notes as shown by an F-score that hovers around 80%. Next, we sought to improve performance on the remaining 20% of the part names that were missed or incorrectly labeled by HMMs by introducing a more flexible model called Linear-Chain Conditional Random Field [9, 10]. We hypothesized that the errors not handled by HMMs could be handled using the observation dependencies found in repair notes. It is desirable to integrate these dependencies into the models to improve overall classification accuracy.

We used unigram and bigram features. Unigram features are obtained using the identity of the current word. For each word seen in the training data, the unigram feature value is assigned to 1. Bigram features use previous and current word.

Contrary to our original hypothesis, extraction using a CRF did not outperform HMM with Shrinkage, although CRF does perform substantially better than the HMM without shrinkage.

Average Recall	85.41
Average Precision	74.86
F-Score	79.78

Table 2 Five fold cross-validation results for CRF with a context width of two.

Analyzing the misclassifications of the HMM and the CRF, we noticed that there was a fair amount of difference in which items were misclassified. This suggested a weighted combination of the two techniques might improve performance. To combine the two models, we merged Viterbi search [8] in both the HMM and the CRF using a weighted combination.

Our results from the weighted combination of the HMM and the CRF are shown in Table 3. There is some improvement in overall score, but there is little improvement over either model alone.

Average Recall	84.35
Average Precision	79.64
F-Score	81.93

Table 3 Five fold cross-validation results for HMM+CRF with a context width of two.

6 Discussion

When we used fully expressive HMMs performance was poor, because of an insufficient amount of training data. With shrinkage performance dramatically improved. We thought that we could make further improvement using CRFs, since, unlike HMMs, CRFs would allow us to model observation dependencies. However, CRF did not outperform HMM with shrinkage. Since the two approaches misclassified different part names, it seemed that it might be possible to find an optimal way to combine the two approaches to obtain better performance than with either approach by itself.

E.g., the HMM and the HMM+CRF miss "on" in "on star antenna," reporting only "star antenna" as a part name, whereas the CRF correctly reports the entire part name. It turns out that "on" occurs in the training data mostly as a non-*target* word, so the HMM does not catch it. On the other hand, the CRF catches "on" because of the bigram "on star", which is unambiguously labeled as *target-target* in training. The HMM+CRF is not able to catch it, which means that on this particular observation the HMM label dominates the CRF label.

On the other hand, "pads" in "step pads" is missed by the CRF, but not by the HMM and the HMM+CRF. Here "step" is caught by all three methods since it is unambiguously labeled as *target* in training. But "pads" is not observed in training, so the CRF misses it. However, in the case of the HMM, the *target* label of "step" must have influenced the *target-target* transition between "step" and "pads", thus causing the HMM to not miss "pads" (*target-target* transition probability is considerable). The fact that the HMM+CRF does not miss "pads" must mean that in the combination the HMM label is stronger than the CRF label at this particular observation.

We produced our HMM+CRF combination by merging the Viterbi search used by both HMMs and CRFs. We did this by using the Expectation Maximization algorithm to estimate the optimal weights. We achieved some improvement in overall score, but did not make a substantial improvement over either model alone. The reason is that we use the same pair of weights for each token position during the Viterbi search. Ideally during the Viterbi search it would be desirable to provide a weight of 1 to the model that correctly labeled the token in question and a weight of 0 to the model that did not provide the correct label. If we had an algorithm that could correctly select the appropriate model at each token position, the best f-score we would get is 86.62%. (We found this score by tallying against the labeled version of the test set.) Finally, it is likely that all of the approaches that we tried would have done better had more training data been available.

With improved coverage of training data, both the HMM and the CRF would be able to perform better. In the case of the CRF, we can expect more training data to help improve bigram coverage for use in bigram features. A comprehensive bigram coverage adds robustness even in the presence of ambiguity problem posed by unigrams because bigrams are unlikely to be ambiguously labeled.

7 Conclusions

Textual Case-Based Reasoning is a promising technology for organizations with large amounts of textual information and taxonomies, such as the part name taxonomy described here, which can be used to aid the case-based reasoning application. When, as is the case here, that extraction is from noisy data, effective techniques must be found to extract information, here part names, to support TCBR.

While corpora linguistic techniques can perform well with large amounts of labeled training data, many organizations with textual data are not in a position to develop these large training sets. HMMs with shrinkage and CRF approaches, as described in this paper show how extraction can be successful with much smaller training sets. The benchmarks obtained with TCBR using relatively small amounts of annotated data should be used in a comparison against the algorithm that uses exclusively domain ontology and structured data resources to guide part name extraction and identification.

Many organizations represent institutional knowledge in the form of structured data which either exists as or is easily converted into hierarchical ontology. Such an ontology as we have shown can play an important role in disambiguation of noun phrase spans in short-hand professional notes. Our experience shows that such ontologies underline the language of professional communication and can therefore be efficiently used in their automatic understanding and augmentation. We presented a method for exploiting explicit ontologies to this effect. Automatic ontology construction techniques may find a new use in concert with our method to further improve accuracy of automatic context understanding.

8 References

[1] Bruninghaus, S. and Ashley, K. D. 2005. Reasoning with Textual Cases Proceedings of the International Conference on Case-Based Reasoning (ICCBR), 137-151.

[2] Freitag, D. and McCallum, A. 2000. Information Extraction with HMM Structures Learned by Stochastic Optimization. In Proceedings of the Seventeenth National Conference on Artificial Intelligence, AAAI, 584-589.

[3] Freitag, D. and McCallum, A. 1999. Information Extraction with HMMs and Shrinkage. In Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction, 31-36, July. AAAI Technical Report WS-99-11.

[4] Lafferty, J., McCallum, A., and Pereira, F. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proc. 18th International Conference on Machine Learning.

[5] Lenz, M. 1998. Textual CBR and Information Retreival: A Comparison. In Gierl, L. and Lenz, M. (eds.) Proceedings of the 6th German Workshop on Case-Based Reasoning, IMIB Series vol. 7, Inst. fuer Medizinische Informatik und Biometrie, University of Rostock.

[6] Morgan, A. P., Cafeo, J. A., Gibbons, D. I., Lesperance, R. M., Sengir, G. H., and Simon, A. M. 2003. The General Motors Variation-Reduction Adviser: Evolution of a

CBR System. ICCBR 2003, 306-318.

[7] Morgan, A. P., Cafeo, J. A., Godden, K., Lesperance, R. M., Simon, A. M, McGuinness, D. L., and Benedict, J. L. 2005. The General Motors Variation-Reduction Adviser. AI Magazine 26,3, 18-28.

[8] Rabiner, L. R. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In Proceedings of the IEEE, 77, 2.

[9] Sha, F. and F. Pereira. Shallow Parsing with Conditional Random Fields. Technical Report MS-CIS-02-35, University of Pennsylvania (2003)

[10] Sutton, C. and McCallum, A. 2006. An Introduction to Conditional RandomFields for Relational Learning. In Introduction to Statistical Relational Learning. Getoor, L. and BenTaskar, B. (eds.) MIT Press.

[11] Uschold, M. 2000. Creating, Integrating and Maintaining Local and Global Ontologies. Proceedings of the 14th European Conference on Artificial Intelligence ECAI 2000, Berlin, Germany.

[12] Roberts, A., R. Gaizauskas, M. Hepple, N. Davis, G. Demetriou, Y. Guo, J. Kola, I. Roberts, A. Setzer, A. Tapuria, et al. 2007. The CLEF corpus: Semantic annotation of clinical text. In AMIA Annu Symp Proc, volume 625.

[13] Cover, T.M., Thomas, J.A. Elements of Information Theory. John Wiley and Sons, 2006.

[14] Fellbaum, C., et al. WordNet: An Electronic Lexical Database. MIT Press, 1998.

[15] Chapman, W., Dowling, J.N., and Wagner, M.M. Classification of Emergency Department Chief Complaints Into 7 Syndromes: A Retrospective Analysis of 527,228 Patients. Annals of Emergency Medicine, vol. 46, no. 5, Nov. 2005.

[16] Demner-Fushman, D. UMLS content views appropriate for NLP processing of the biomedical literature vs. clinical text. Journal of Biomedical Informatics, Aug. 2010.

[17] Bundschus, M., Volker Tresp, V., and Hans-Peter Kriegel, H.-P. Topic models for semantically annotated document collections. In NIPS 2009 Workshop: Applications for Topic Models: Text and Beyond, 2009.