# A Novel Approach to Part Name Discovery in Noisy Text

**Nobal B Niraula, Daniel Whyatt, and Anne Kao**
Boeing Research and Technology
Huntsville, Alabama and Seattle, Washington, USA
{nobal.b.niraula, daniel.whyatt, anne.kao}@boeing.com

## Abstract

As a specialized example of information extraction, part name extraction is an area that presents unique challenges. Part names are typically multiword terms longer than two words. There is little consistency in how terms are described in noisy free text, with variations spawned by typos, ad hoc abbreviations, acronyms, and incomplete names. This makes search and analyses of parts in these data extremely challenging. In this paper, we present our algorithm, PANDA (Part Name Discovery Analytics), based on a unique method that exploits statistical, linguistic and machine learning techniques to discover part names in noisy text such as that in manufacturing quality documentation, supply chain management records, service communication logs, and maintenance reports. Experiments show that PANDA is scalable and outperforms existing techniques significantly.

## 1 Introduction

Part information plays a key role in manufacturing, maintenance, supplier management and customer support of any large complex system, such as an airplane, which may easily involve over 30,000 types of parts. Parts can be described by part numbers or nomenclature. Furthermore, a given part serving the same function can often be supplied by multiple suppliers, who may use different part numbers and do not always use the same nomenclature to describe functionally equivalent parts. In addition, part names are very frequent in the service descriptions and notes written by mechanics and engineers around the world. Due to time constraints, working conditions (maintenance mechanics do not work in an office environment), time crunch, and job focus (primarily getting the aircraft ready for on-time takeoff, not writing perfect English), compounded by the fact that many of those involved are not native speakers of English, the data often contains a high percentage of non-standard spellings and ad hoc shorthand notations and typos. Table 1 exemplifies these issues with real sample maintenance records.

| |
|---|
| *lh side sidewall panel between sta 1600 and 1700 (pos 41l and 42l) worn* |
| *new electrical panel fitted iaw amm task xx-xx-* |
| *lwr ctr display selected lt inop on display selector pnl.* |

Table 1: Sample maintenance records containing parts (*highlighted*), typos, and ad hoc abbreviations (*lh* for left, *pnl* for *panel*, *lwr* for *lower*, etc.)

In order to pinpoint types of issues involved in manufacturing, maintenance support, or supply chain management, it is crucial to identify the specific part involved. Importantly, a robust and scalable approach for extracting parts from text of the nature described above should never rely on simple matching from a list of predefined part names. It should also have a way of exploiting abundant free text data amassed over years.

While information extraction is a well-studied field, typically information extraction focuses on people, organization, time, location, event and their relationship. Part name extraction is much less studied. Part name extraction has the following unique properties. First, the language of part names as well as their context in these data sources is very domain-specific. This means that, not only is there nothing analogous to special word lists like people's first and last names or city, state and country names, but general language resources like WorldNet (Miller, 1995) and Freebase (Freebase, 2018) are also virtually useless. In addition, part names are often longer than the names of people, organizations, or locations and can be as long as 5-



*Proceedings of NAACL-HLT 2018*, pages 170–176
New Orleans, Louisiana, June 1 - 6, 2018. 

7 words (e.g. *variable speed constant frequency cool fan*). Furthermore, it is often the case that a substring Y (e.g. *landing gear*) of a part name X (e.g. *main landing gear*) is also a valid part name. This creates a challenge for a system to identify X instead of Y as the part being mentioned. In addition, as noted above, the free text data containing the part names are often non-professionally authored and contain a high percentage of spelling variants (both ad hoc abbreviations and typos) and domain-specific acronyms. The spelling challenges do not just occur in part names, but occur throughout the free text, posing challenges for either traditional grammar-based parsing or n-gram approaches.

In this paper we present PANDA (Part Name Discovery Analytics), a fast and scalable method that exploits statistical, linguistic and supervised machine learning techniques in a unique way such that minimal human supervision is sufficient to discover thousands of part names from noisy text.

## 2 Related Work

Basic information extraction methods typically rely on language models and hand-crafted rules. The n-gram approach derives common strings from a large corpus and, together with hand-crafted rules, makes for an easy-to-implement way of inferring entities (Chandramouli, Subramanian, & Bal, 2013). A more sophisticated approach would define rules based on regular expressions over text content or their parts-of-speech tags to extract information. Noun-phrase identification is a typical approach under this category (Vilain and Day, 2000). Rule-based and language model-based systems are very effective in cases when the entities of interest follow specific patterns. However, when the text is very noisy, generating hand-crafted rules and patterns is cost-prohibitive and not feasible.

To our knowledge, there is only one previously published work specifically focusing on part name extraction (Chandramouli, Subramanian, & Bal, 2013). The authors propose an n-gram based approach which extracts part names from service logs. Given a list of basic part types (e.g. *valve*), they generate bigrams and trigrams ending with those part types and consider them as part candidates. The candidates are ranked using a mutual information metric. Furthermore, the authors found that Part-of-Speech (POS) based filtering improved the quality of prediction. While this work is unsupervised and easy to implement, it has important limitations. First, it cannot predict any new part types, because it relies on predefined part types. Therefore, any part types which are not already known will be missed. Secondly, their system cannot extract part names which have more than three tokens. In our data, part names consisting of more than three tokens occur frequently (i.e. *left main landing gear, horizontal stabilizer trim actuator*). Importantly, all n-gram based approaches suffer from the pervasive misspellings and abbreviations in noisy data. They may not be able to extract *outflow vlv*, *trim act* or *door switche*, as the respective part types *valve*, *actuator* and *switch* are misspelled or written in a non-standard way.

To enable more flexibility and more power in extracting entities, machine learning methods, especially supervised learning methods, have become a natural choice in modern day information extraction. Typical machine learning methods considered include Hidden Markov Models (HMM) (Skounakis, Craven and Ray, 2003; Freitag and McCallum, 1999), and Conditional Random Fields (CRFs) (McCallum, 2002). The supervised systems learn a set of rules or models from the supplied hand-tagged samples for the training phase of machine learning. Once a new model is built based on training data, the model can be applied to new documents to extract entities. Rules and models learned by supervised techniques are effective for extracting information from the same genre of documents they are trained on, but they may perform poorly when applied to a different genre. In addition, acquiring the right training examples can be very expensive. Part information extraction is one such area which requires SME (Subject Matter Expert) knowledge and is thus not suited to crowd sourcing.

Recent approaches that address the scalability problem in training data associated with supervised machine learning include weakly-supervised methods (Pasca, 2007), bootstrapping techniques (Vilain and Day, 2000; Maedche, 2003), and active learning (Thompson, Callif and Mooney, 1999; Williams et al., 2015). Active learning starts with bootstrap samples creating an initial model and uses that model to select the most informative examples in order to minimize the annotation cost required to generate training examples. A new model is created with the new examples and the process continues until a stopping criteria is met. However, such iteration still requires SME involvement. When the free text data contain a high degree of

noise, and the number of parts involved is in the tens of thousands, it is not clear how fast an active learning approach will converge. Besides, SMEs are very expensive and reducing their efforts in part name extraction process, as do ne by our method PANDA, offers a huge cost saving for a company.

Studies show that open-ended and domain independent information extraction systems do not work well for domain-specific information extraction (Etzioni et al., 2004). As such, existing approaches can be expected to perform poorly for part extraction, a domain specific information extraction problem. To this end, we propose a system that is tuned to extract parts from the target natural language text (e.g. maintenance logs). The proposed system is robust so that it can operate on noisy text and yet scalable as it demands very minimal supervision.

## 3 Part Name Discovery Analytics

The intent of Part Name Discovery Analytics (PANDA) is primarily to extract part information in the domain of aircraft to support vehicle health management by exploiting hundreds of thousands of free text records. However, its use can extend to any large data sets containing part name mentions. The primary design philosophy is to utilize machine learning capabilities and at the same time exploit linguistic knowledge of how part names are constructed in English. This allows discovery of new parts and at the same time minimizes the expensive training process required for supervised machine learning. Dealing with highly noisy data is a key requirement of this domain. Therefore, a non-learning based method would not meet our requirements. As we show later, PANDA learns to infer new part names from the noisy text.

We leverage the linguistic fact that the most important term in a multiword part name is the head noun (the "Head"), and in English, the Head is the last term in a multiword term. These Heads are terms such as *panel*, *valve*, *switch* etc. Although most people who are somewhat familiar with this domain can easily come up with 10-20 examples of these Heads, it is important to note that there is no knowledge base anywhere that contains all of them. By utilizing linguistic knowledge, we can automatically provide the most effective training examples to the machine learning algorithm, as well as greatly minimize SME review in providing crucial feedback for the machine learning process.

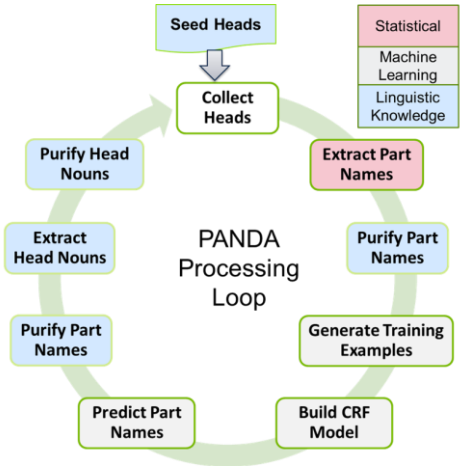


Figure 1: PANDA Processing Loop.

At a high level, PANDA cleverly shuttles between Heads and Heads plus modifiers, which are the full part names of interest. The requirement for SME's attention are focused on Heads. Since SME's need only review Heads, and not the full parts associated with each Head, the training is highly efficient.

Fig. 1 shows the architecture of PANDA. It consists of a loop which starts with seed Heads, a small set of basic part names such as *gear, panel, switch*, etc. The collected Heads are used to predict the part names in the Extract Part Names step (Section 3.1). The extracted part names are "purified" using several filtering mechanisms (Section 3.2). The purified parts are used to generate training examples (Section 3.3) for a CRF model (Section 3.4) which, in turn, is used to predict new part names in the data set (Section 3.5). The predicted part names are again purified (Section 3.2) and new Heads are extracted (Section 3.6). The extracted Heads themselves are also purified (Section 3.7). Finally, the purified Heads are added back to the earlier list forming a larger initial set of Heads. The loop is repeated until a stopping criteria is met (Section 3.8). Parts predicted by CRF and trie in the last run are collected as the final output.

### 3.1 Trie-based Part Name Prediction

The purpose of this step is to use part Heads and automatically generate complete part names that we need later to generate training examples for a machine learning model. To do this, we construct a data structure called a trie (Trie, 2018) from a large corpus such that the first level nodes are the given part Heads (e.g. *gear*) and their descendant nodes are the tokens appearing before them in the data set (see Fig. 2 below). This type of trie is computed by

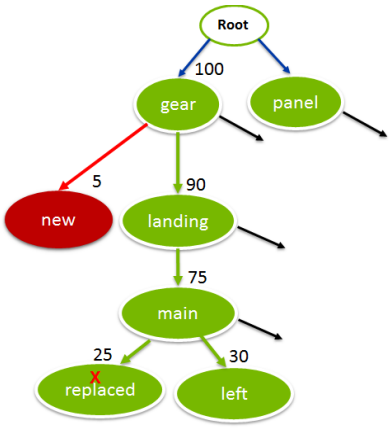


Figure 2: Sample trie generating parts *main landing gear* and *left main landing gear*.

scanning the tokens in reverse order, and is highly efficient. We then traverse the trie in depth-first fashion as long as it satisfies the minimum frequency criteria. We collect the potential part name sequence as we traverse. To further ensure better examples are used in the training example generation process, we could have a constraint to only use the Heads having certain minimum frequency.

### 3.2 Purify Part Names

As is the case of all machine learning methods, a significant amount of bad training samples may negatively impact the resulting model. Thus, to further improve the quality of part names predicted in the previous step, this step applies a number of heuristics based on POS features. For instance, a part name must not start with a verb or an article. If it does, we remove them and considered the remaining chunk as part name (e.g. *replaced main landing gear* becomes *main landing gear*). Similarly, PANDA requires that all part name tokens must either be nouns or adjectives.

### 3.3 Generate Training Examples

The goal of this phase is to generate training data for a machine learning model by annotating the data in the corpus with the part names resulting from the previous steps. Since the goal is to leverage patterns in the part names and their context to discover new part names, additional features need to be provided. PANDA currently employs k-previous and k-next word tokens and their POS tags as well as the POS tags of part names themselves as these features. The POS features can be generated using a POS Tagger such as Brill Tagger (Brill, 1992). Fig. 3 shows a sample annotated record with the part name left main landing gear and corresponding POS-tags.

| DT | JJ | NN | NN | NN | MD | RB | VB |
|---|---|---|---|---|---|---|---|
| The | left | main | landing | gear | will | not | retract |

Figure 3: A sentence annotated with a part name.

### 3.4 Train a Sequence Model

The goal of this phase is to use the annotated corpus as training data to generate a model that identifies part names in the data. Any sequence model that extracts sequences of tokens, such as a CRF (Lafferty, McCallum and Pereira, 2001) or Long Short Term Memory network (Gers, Schmidhuber and Cummins, 1999), can be used at this phase. We use CRF in our experiments.

### 3.5 Predict Part Names

The goal of this phase is to use the sequence model trained in Section 3.4 on the corpus to extract new potential part names. The newly predicted parts are collected and purified using the approach presented in Section 3.2.

### 3.6 Extract Head Nouns

This step is to extract Heads from the newly identified potential part names in the machine learning output. It extracts the last token of the supplied part and returns that as the Head. For instance, it returns *cap* for *oil filter cap.*

### 3.7 Purify Head Nouns

The goal of this step is for PANDA to validate Heads generated in the previous phase. The feedback can be done with a human-in-the-loop (a SME). The SME will review all generated Heads and classify them into different categories, typically *Good* (e.g. *antenna*) or *Bad* (e.g. *inoperable*). Optionally, an additional category *Borderline* (e.g. *unit*) can be used. However, only *Good* Heads are used in the next iteration of the loop to generate additional new part names. *Borderline* heads will not be used to generate new part names for the training purpose, but will be accepted as potentially valid heads at the last run.

### 3.8 Stopping Criteria for the Loop

PANDA supports various types of stopping criteria. It can be stopped after a certain number of iterations or after a certain number of parts are generated, or after it reaches a certain ratio of bad vs. good new parts generated.

### 3.9 Part Prediction using PANDA

The parts collected after the final run can be used to extract the part names in new incoming records. Alternatively, the final CRF model can also be used to predict the parts. Or a combination of both of these can be used.

## 4 Experiments and Results

We conducted experiments using three major key data sources in the aerospace domain: (a) *Maintenance Logbook (MLB)* includes key maintenance condition, maintenance action and parts involved for issues identified on an aircraft. (b) *Schedule Interruption (SI)* includes records generated by dozens of major airlines at airports all over the world. It contains reasons for significant delays in departure or landing, often due to the condition of one or more parts/systems. (c) *Communication Systems (CS)* includes professional help desk type correspondence between an aircraft manufacturing company and airline operators. MLB and SI are very noisy (as shown in Table 1) compared to CS. All data sets contained multiple records, had comparable sizes of 1 million tokens each, and were subject to the same preprocessing steps and POS-tagging.

We ran PANDA on SI data set using 36 seed head nouns. We set PANDA to identify full part names of length up to 5 tokens with minimum frequency threshold of 1, to capture maximum recall, and allowed it to run till no new good Head was generated. The results are presented in Table 2. PANDA stopped after iteration 7 generating 9374 parts. SME's feedback to predicted Heads in each iteration as Good, Bad and Borderline heads, defined in Section 3.7, are also presented in the table. Starting from 36 initial Heads, PANDA was able to extract 382 (= 317 Good + 65 Borderline) new part Heads. This demonstrates PANDA's ability to infer new part Heads which are not known initially. This is crucial because all Heads are not known in advance and hundreds of new full parts may be associated with a single new Head.

Table 2 also shows the total number of parts collected up to a given iteration. It extracted 9374 full parts at the end of the final iteration but only required annotations of 780 part Heads. Since the annotation task only involves annotating the Heads and not the full parts, the annotation is very fast. As a reference, this whole experiment took less than 2 hours to complete. This demonstrates the scalability of PANDA in that it requires minimal human input in the training phase of machine learning. Since previously annotated Heads can be reused in subsequent experiments, PANDA will run even faster in the later experiments.

| | Good | Bad | Borderline | Total Parts |
|---|---|---|---|---|
| **Seed** | 36 | - | - | - |
| **Iter 1** | 98 | 97 | 19 | 3370 |
| **Iter 2** | 127 | 146 | 30 | 7853 |
| **Iter 3** | 48 | 95 | 9 | 8794 |
| **Iter 4** | 27 | 27 | 3 | 9020 |
| **Iter 5** | 12 | 18 | 2 | 9249 |
| **Iter 6** | 3 | 11 | 2 | 9283 |
| **Iter 7** | 2 | 4 | 0 | 9374 |
| **Total (1-7)** | 317 | 398 | 65 | 9374 |

Table 2: PANDA results showing annotation counts for Good, Bad and Borderline heads and total full parts on SI data set

| | Heads | Parts |
|---|---|---|
| **Baseline** | 36 | 979 |
| **PANDA** | 382 | 9374 |

Table 3: Baseline and PANDA extracted parts using same 36 seed heads

Next we sought to evaluate the quality of full parts generated by PANDA. However, no gold data set is currently available for that purpose. Also, evaluation in terms of recall by annotating all parts is not feasible, as annotating all 9374 full parts would be very costly. Therefore, we randomly selected 1000 parts for evaluation. PANDA scored 80.9% accuracy on this evaluation. This clearly shows that, though a SME only provides feedback on Heads during the training process, PANDA is still able to extract full part names from noisy data with a high degree of accuracy.

### 4.1 PANDA VS Baseline

As noted in Section 2, the only known algorithm in the literature to extract part names from free text is by Chandramouli, Subramanian, and Bal (2013). It considers parts as n-grams ending at provided heads and ranks them by a collocation measure. We implemented their best performing algorithm that purifies parts with POS tags as baseline. We ran both the baseline and PANDA to extract full part names of length up to 5 words from the SI data, with a minimum acceptable collocation value of 25 for the baseline. The results are shown in Table 3.

Since the baseline relies on provided Heads - 36 in this case - and has no way of inferring new part Heads and their corresponding parts, it suffers from

low recall. PANDA, on the other hand can easily infer new Heads and associated parts. For instance, although *annunciator* was not in the initial Head list, PANDA was able to infer it and its variants such as *annunc*, *annuc*, and *ann*. In addition, PANDA extracted 32 types of annunciators such as *antiskid annunciator*, *door warn annunciator*, and *cabin zone temp annunc*.

In addition, PANDA captured the longest part name possible (a very specific part ) while the baseline broke it down to its constituent chunks, creating parts that did not exist in the data set or were incorrect. Baseline results contained 186 such over-generated parts. In one example, when *access* always preceded *door panel*, PANDA only generated *access door panel*. In contrast, the baseline generated *door panel* (a non-existence part) as well as *access door panel*. Since *door panel* can easily be derived from *access door panel* (a specific part), PANDA still is able to identify generic parts, if they exist, in new incoming records without generating the parts that do not exist in the current data set, which may lead to error. For instance, the constituent part *one valve* that the baseline generates from *generator one valve* is not by itself a valid part.

Lastly, the baseline generated incorrect parts when there were more than one head in a part name. It extracted *temperature control valve*, *temperature control,* and *control valve* from the record containing "temperature control valve" as they were n-grams ending at known heads *control* and *valve*. In fact, out of 979 baseline parts, 466 were common with PANDA and the rest were either over-generated or invalid parts. These facts clearly demonstrate PANDA's superiority over the baseline model in terms of recall, learning ability for heads and parts, and accuracy of extracted parts.

### 4.2 PANDA on Diverse Data Sets

To test the generality of PANDA across different genres of part records, we ran PANDA on MLB, SI and BCS data sets for 5 iterations each. As noted above, MLB and SI are very noisy compared to BCS. Each of these experiments needed less than 2 hours. We report head annotation counts and total extracted parts in each of these data sets in Table 4. The results show that PANDA can process data sets of different genres with minimal annotations and can extract thousands of complex part names from them. As expected, fewer parts were discovered in BCS than in SI and MLB since it consisted of email conversations with boilerplate texts.

| Data Set | Good | Bad | Borderline | Total Ann. | Total Parts |
|---|---|---|---|---|---|
| **MLB** | 463 | 604 | 92 | 1159 | 8721 |
| **SI** | 312 | 379 | 63 | 754 | 9249 |
| **BCS** | 293 | 390 | 64 | 747 | 6554 |

Table 4: Head annotation counts and total parts across data sets of different part genre

### 4.3 Error Analysis

We identified some error types that affected PANDA results. First, there are certain parts that PANDA could not correctly extract due to its assumption that the last word of a part is the head of the part. From the text "Replaced handle of door", it could capture *handle* and *door* separately but not as *door handle* or *handle of door*. Such cases, however, were very rare. Second, POS-tagging errors affected some of PANDA's predictions. It captured *report generator drive* instead of *generator drive* due to *report* being incorrectly tagged as a noun. Third, a few parts were only partially captured due to the maximum part length setting. For instance *variable* was missed in the 6-word part *variable speed constant frequency cool fan.*

## 5 Conclusion and Future Work

We presented PANDA, a novel approach that discovers part names in noisy text. PANDA cleverly exploits the linguistic characteristics of part names in English to automatically generate full part names using basic part names. This automates the training example generation process, the most expensive step for building a supervised machine learning model. Experiments demonstrated that:

- PANDA required minimal human input for training the machine learning model
- PANDA was superior to the existing system in that it was able to infer new heads and parts and dramatically improved recall as compared to the existing system
- PANDA extracted high quality full parts
- PANDA can scale across diverse data sets

With these promising results, PANDA is currently being deployed to extract part names from several data sets for different aircraft models and subsystems. In the future, we plan to focus on the normalization of heads (e.g. *pnl* and *panal* to *panel*) and parts (e.g. *lft valve* and *left vlv* to *left valve*) from PANDA extracted results.

## References


Brill, E. (1992, March). A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing* (pp. 152-155). Association for Computational Linguistics.

Chandramouli, A., Subramanian, G., Bal, D., Ao, S. I., Douglas, C., Grundfest, W. S., & Burgstone, J. (2013). Unsupervised extraction of part names from service logs. In *Proceedings of the World Congress on Engineering and Computer Science* (Vol. 2).

Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A. M., Shaked, T. & Yates, A. (2004, May). Web-scale information extraction in knowitall:(preliminary results). In *Proceedings of the 13th international conference on World Wide Web* (pp. 100-110). ACM.

Freebase. (2018). http://www.freebase.com. Retrieved January 2018.

Freitag, D., & McCallum, A. (1999, July). Information extraction with HMMs and shrinkage. In *Proceedings of the AAAI-99 workshop on machine learning for information extraction* (pp. 31-36).

Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). *Learning to forget: Continual prediction with LSTM*.

Lafferty, J., McCallum, A., & Pereira, F. C. (2001). *Conditional random fields: Probabilistic models for segmenting and labeling sequence data*.

Miller, G. A. (1995). *WordNet: a lexical database for English*. Communications of the ACM, 38, 39-41.

McCallum, A. (2002, August). Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence* (pp. 403-410). Morgan Kaufmann Publishers Inc.

Maedche, A., Neumann, G., & Staab, S. (2003). Bootstrapping an ontology-based information extraction system. In *Intelligent exploration of the web* (pp. 345-359). Physica, Heidelberg.

Paşca, M. (2007, November). Weakly-supervised discovery of named entities using web search queries. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (pp. 683-690). ACM.

Skounakis, M., Craven, M., & Ray, S. (2003, August). *Hierarchical hidden markov models for information extraction*. In IJCAI (pp. 427-433).

Thompson, C. A., Califf, M. E., & Mooney, R. J. (1999, June). Active learning for natural language parsing and information extraction. In *ICML* (pp. 406-414).

Trie. (2018). https://en.wikipedia.org/wiki/Trie. Retrieved January, 2018.

Vilain, M., & Day, D. (2000, September). Phrase parsing with rule sequence processors: an application to the shared CoNLL task. In *Proceedings of the 2nd workshop on Learning language in logic and the 4th conference on Computational natural language learning*-Volume 7 (pp. 160-162). Association for Computational Linguistics.

Williams, J. D., Niraula, N. B., Dasigi, P., Lakshmiratan, A., Suarez, C. G. J., Reddy, M., & Zweig, G. (2015). Rapidly scaling dialog systems with interactive learning. In *Natural Language Dialog Systems and Intelligent Assistants* (pp. 1-13). Springer, Cham.

Unsupervised Extraction of Part names from Service Logs. In Proceedings of the World Congress on Engineering and Computer Science, II, pp. 23-25.