# Leveraging Wikipedia for Ontology Pattern Population from Text

**Michelle Cheatham** and **James Lambert**
Wright State University
3640 Colonel Glenn Hwy.
Dayton, OH 45435

**Charles Vardeman II**
University of Notre Dame
111C Information Technology Center
Notre Dame, IN 46556

**Abstract**

Traditional approaches to populating ontology design patterns from unstructured text often involve using a dictionary, rules, or machine learning approaches that have been established based on a training set of annotated documents. While these approaches are quite effective in many cases, performance can suffer over time as the nature of the text documents changes to reflect advances in the domain of interest. This is particularly true when attempting to populate patterns related to fast-changing domains such as technology, medicine, or law. This paper explores the use of Wikipedia as a source of continually updated background knowledge to facilitate ontology pattern population as the domain changes over time.

## Introduction

Two of the central underpinnings of scientific inquiry are the need to verify results through reproduction of the experiments involved and the importance of "building on the shoulders of giants." In order for these things to be possible, experimental results need to be both discoverable and reproducible. Important steps have been made recently in pursuit of this, including the relaxation of page restrictions on "methodology" sections in many academic journals and the requirements by some funding agencies that investigators make any data they collect publicly available. However, in order for previous work to be truly verifiable and reusable, researchers must be able to not only access the results of those efforts but also to understand the context in which they were created. A key element of this is the need to preserve the underlying computations and analytical process that led to prior results in a generic machine-readable format.

In previous work towards this goal, we developed an ontology design pattern (ODP) to represent the computational environment in which an analysis was performed. This model is briefly described in Section  of this work, and more detail is available in (Cheatham et al. 2017). This paper describes our work on the next step: development of an automated approach to populate the ODP based on data extracted from academic articles. We explore the performance of two common approaches to this task and show that, due to the fast-changing nature of computer technology, they lose their effectiveness over time. We then evaluate the utility of using a continuously manually-curated knowledge base to mitigate this performance degradation. The results illustrate that this method holds some promise.

The remainder of this paper is organized as follows. The section Computational Environment Representation presents the schema of the ontology we seek to populate, while the Dataset section describes the collection of academic articles we use as our training and test sets. The approach and results are presented and analyzed next, and finally some conclusions and ideas for future work in this area are discussed.

## Computational Environment Representation

The Computational Environment ODP was developed over the course of several working sessions by a group of ontological modeling experts, library scientists, and domain scientists from different fields, including computational chemists and high-energy physicists interested in preserving analysis of data collected from the Large Hadron Collider at CERN. Our goal was to arrive at an ontology design pattern that is capable of answering the following competency questions:

- What environment do I need to put in place in order to replicate the work in Paper X?
- There has been an error found in Script Y. Which analyses need to be re-run?
- Based on recent research in Field Z, what tools and resources should new students work to become familiar with?
- Are the results from Study A and Study B comparable from a computational environment perspective?

We focused on creating a model to capture the actual environment present during a computational analysis. Representing all possible environments in which it is feasible for the analysis to be executed is outside of the scope of our current effort. We also do not include the runtime configuration and parameters as part of the environment. The rationale is that to some extent the same environment should be applicable to many computational analyses in the same

field of study, but this would not be true if we included such analysis-specific information as runtime parameters as part of the environment. Data sources were considered outside of the confines of the computational environment for similar reasons. External web services and similar resources were not included because they are not inter-related in the same way as the environmental elements are. For instance, deciding to use a different operating system often necessitates using a different version of drivers, libraries, and software applications, whereas in most cases the entire hardware configuration could be changed with no impact on external services.

Figure 1 shows the schema that we targeted for population in this study. It is a slightly modified version of the ODP presented in (Cheatham et al. 2017) – the overall goal and competencies of the pattern remain the same, but some entities have been omitted, and properties related to the manufacturer, make and model of computers and hardware components have been added, as well as an entity related to programming language, because this information is important for our current application goals.

## Dataset

The dataset consists of 100 academic articles published in 2012 or later (called "current")[1] and 20 published in or before 2002 (called "old"). There are 20 current papers and four old ones from each of five fields of study: biology, chemistry, engineering, mathematics, and physics. Twenty of the current articles were randomly selected to make up the current training set while the remainder form the test set. The documents were collected by searching Google Scholar using the query <field of study> AND algorithm (e.g. biology and algorithm, chemistry and algorithm). Patents and citations were omitted from the search criteria. Any paper with a PDF download link was searched for the terms cpu and processor, in order to quickly determine if the paper had any content related to a computational environment. If the paper contained either term, it was retained in the dataset.

We then manually created a gold standard for the dataset by going through each paper and listing any information relevant to the ODP shown in Figure 1. This process was completed by a single person, but in the few cases in which a question arose (e.g. whether an R4000 is a make or a model), the opinions of others and external knowledge sources such as manufacturer's websites and websites about historical computing technologies were consulted. A typical entry in the gold standard is shown in Listing 1. Note that the entries given do not directly correspond to entities in the ontology. For example, the single tag memory_size_unit: GB corresponds to an instance of the Memory class in the ontology with a hasSize of some Amount that in turn hasUnit GB. Tagging based on each entity within the ontology, including those such as Memory that represent blank nodes, would have made the tagging process more arduous, however, and the information can be readily expanded to match the desired schema using SPARQL construct queries, as described in (Zhou et al. 2018).

Listing 1: Sample entry from the gold standard

```
-bio15
cpu_make: HT Xeon
cpu_frequency_value: 3.2
cpu_frequency_unit: GHz
cpu_num-cores: 16
memory_type: RAM
memory_size_value: 8
memory_size_unit: GB
programming-language: C/C++
os_kernel_name: Linux
os_distribution_name: Red Hat
```

A preliminary analysis of the training and test datasets indicate that approaches trained on the older articles may face difficulties. For example, Figure 2 shows the number of distinct values for each tag in the test set (left), current training set (middle) and old training set (right). From this we can see that the relative number of distinct values for each tag is about the same in the test set and the current training set (i.e. the tags with the largest number of distinct values in the test set are also those with the largest number of distinct values in the training set). This is less true for the old training set – for instance, the older articles never talk about the number of CPU cores (presumably because they were all single core) and the only programming language they mention is Fortran. It therefore may be difficult for models trained on the old training set to correctly extract information relevant to these tags in the test set. New entities becoming relevant over time are sometimes termed "emergent entities" and are particularly challenging for many NER systems (Nakashole, Tylenda, and Weikum 2013).

Figure 3 shows the number of times each tag was used in the test set and both training sets. It is evident that the older articles talk more at the level of computer manufacturer, make and model, while more current articles mention more details such as information about the CPU, the graphics card, and the amount of memory. Looking at both figures (2 and 3), we see that some tags, such as CPU manufacturer, are key for this ontology population task, because there are only a few distinct values that must be recognized, but these few values occur many times within the test set. Models trained on the old training set may be at a particular disadvantage in these cases, if the values for those key tags in the older documents are not reflective of those in the test set.

## Approach and Results

In this work we formulate the problem of populating the computational environment pattern from text solely as a Named Entity Recognition (NER) task. This is in contrast to many other approaches that divide this process into two steps: entity recognition/typing and relation extraction (Petasis et al. 2011). In other words, we are trying to directly arrive at the tags specified in the gold standard as described in the Dataset section, with the intention of later creating

[1]Three articles from the "current" set had to be thrown out at a later stage due to irregularities that caused them to be unparseable by multiple PDF parsers.

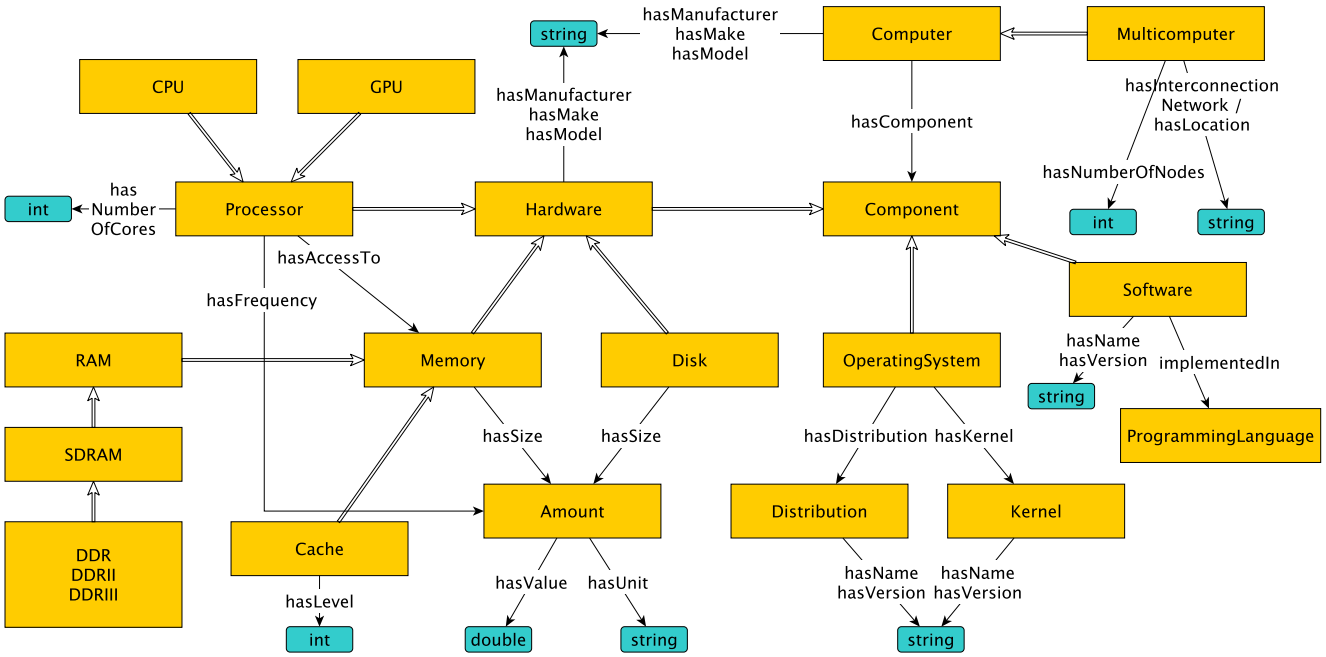


Figure 1: The Computational Environment ODP

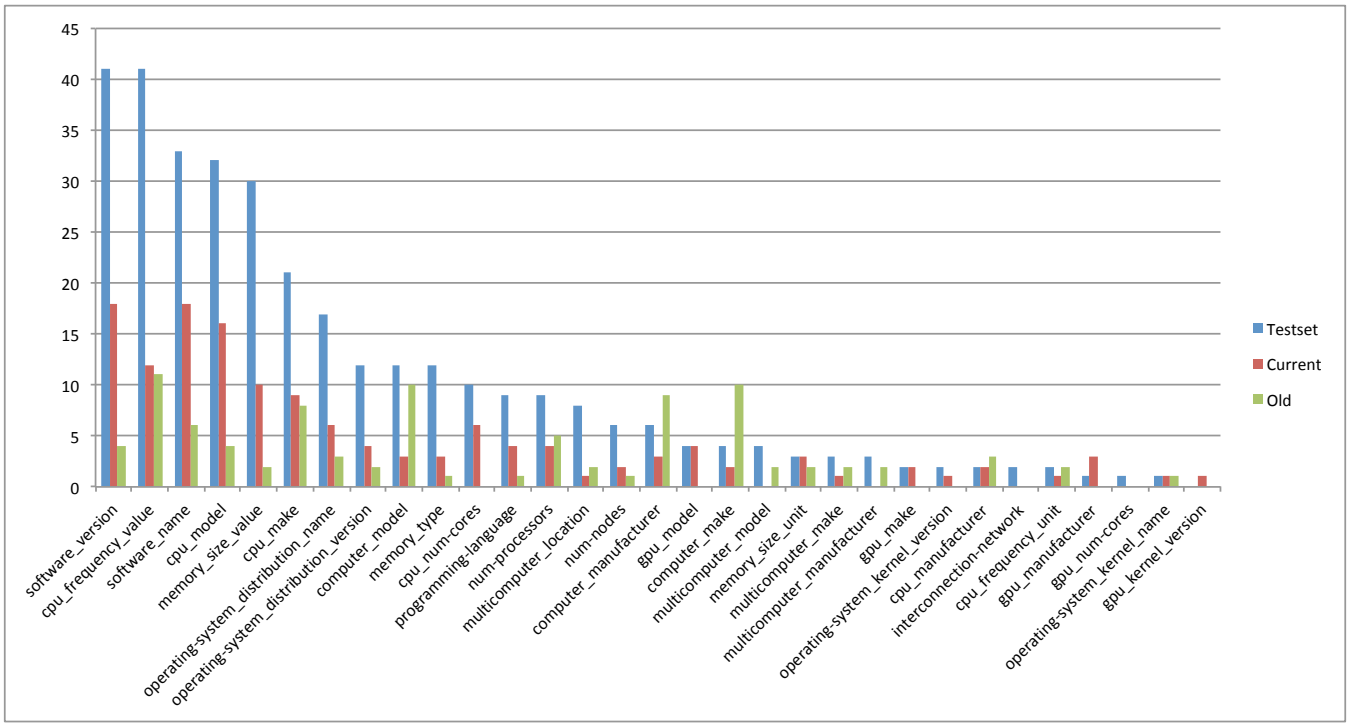


Figure 2: The number of distinct values for each tag in the test set and the current and old training sets.

SPARQL construct queries to expand these tags into the terminology used by the ODP. If more than one computational environment is described in an article, the appropriate relations between instances can be then determined based on proximity in the underlying text.

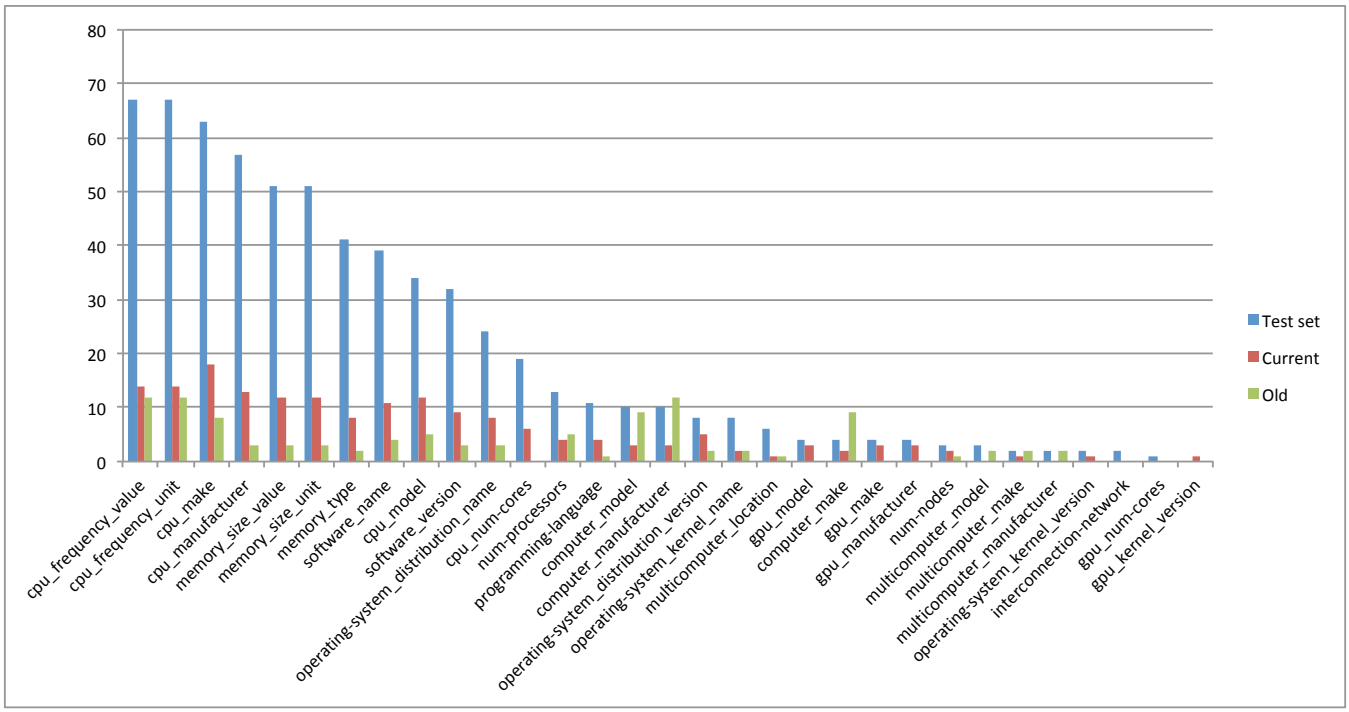


Figure 3: The number of occurrences of each tag in the test set and the current and old training sets.

## Preprocessing

In academic articles, the computational environment tends to be discussed in a small number of isolated points within a document, often a single paragraph, sentence, footnote, or caption. Because some of the techniques we employ in our approach, particularly the use of Wikipedia, are time-intensive, we begin our ontology population task by attempting to identify the key portions of each document (which we generically refer to as the "key paragraph", with the understanding that it may actually be a footnote, caption or other element). Our goal in doing this was to arrive at an ontology-agnostic approach with high recall. The approach we take is quite basic: a Python script is given the ontology (in the form of an OWL file) and the name of the academic article in which to identify the key paragraphs. The script parses the names of all entities from the ontology, splits the entire content academic article (including footnotes, captions, etc.) into paragraphs, and counts the number of times any ontology term appears in each paragraph. Any paragraph with a count within 90 percent of the maximum count for that article is considered a key paragraph. This approach produces a recall of .94 and a precision of .99, for an F-measure of .96. The remainder of the discussion in this section assumes that the key paragraphs have been successfully identified prior to invoking the approach under consideration. The text in each key paragraph is split into sentences, tokenized, lemmatized, and part of speech tagging is performed using the Stanford NLP pipeline (Manning et al. 2014).

## Machine Learning

NER techniques generally fall into two categories: machine learning-based and rules-based (Nadeau and Sekine 2007). Machine learning-based NER systems typically involve training a classifier by manually tagging input documents. The classifier then attempts to learn how to correctly recognize and type entities from new documents based on the training set and a set of features such as a word's part of speech, position in a document, prefix/suffix, frequency, etc. Various approaches have been employed to model the relationship between the features and the entities, including Support Vector Machines (Isozaki and Kazawa 2002), Maximum Entropy (Chieu and Ng 2002), and neural networks (Lample et al. 2016). Because the task of manually generating training data is onerous, there are also semi-supervised and unsupervised machine learning-based NER approaches, but these often only rival, rather than exceed, the performance of supervised systems (Nadeau and Sekine 2007) and so are not considered here.

In this work we applied a Conditional Random Field (CRF) based classifier to the task of tagging information relevant to the computational environment ontology. CRF was chosen due to its long-standing popularity for information extraction from unstructured text (Kristjansson et al. 2004; Bundschus et al. 2008). We again used the Stanford NLP group's implementation (Finkel, Grenager, and Manning 2005). This classifier uses features such as word order, n-grams, part of speech, and word shape to create a probabilistic model to predict the tags in previously unseen documents. We developed models based on both the current and old training sets, an example of which is shown in Listing 2

| Dataset | Model | Precision | Recall | F-measure |
|---|---|---|---|---|
| Training | Current | 0.95 | 0.93 | 0.94 |
| | Old | 0.95 | 0.93 | 0.94 |
| Test | Current | 0.90 | 0.52 | 0.66 |
| | Old | 1.00 | 0.01 | 0.02 |

Table 1: Performance of a machine learning-based approach

| Dataset | Model | Precision | Recall | F-measure |
|---|---|---|---|---|
| Training | Current | 0.96 | 0.93 | 0.95 |
| | Old | 0.99 | 0.94 | 0.97 |
| Test | Current | 0.79 | 0.67 | 0.73 |
| | Old | 0.80 | 0.28 | 0.41 |

Table 2: Performance of a rules-based approach

(the O symbol indicates that no tag is relevant for that token).

Listing 2: Tagged data used to train the CRF classifier

```
running O
on O
a O
PC O
with O
Linux os_kernel_name
CentOS os_distribution_name
5 os_distribution_version
, O
1 memory_size_value
GB memory_size_unit
memory O
and O
2.4 cpu_frequency_value
GHz cpu_frequency_unit
CPU O
```

We then used each model to tag the articles in the test set and assessed the performance in terms of precision, recall and F-measure (Table 1). The F-measure of the approach on the training data is not quite 1.0 because tagging in the Stanford NLP pipeline only happens at the level of tokens, and some of the articles contain malformed tokens, such as IntelCorei7, that contain information about more than one tag. Still, the performance of both models is quite good on the training data. The F-measure using the model trained on current documents drops considerably on the test set, but 0.66 may good enough to be of some use in many applications (Others have found that the performance of NER in technical domains such as biomedicine is in the 58-75 range (Zhang and Ciravegna 2011)). Conversely, the performance using the model trained on older articles is abysmal.

## Rules

Rules-based NER systems allow users to craft rules using a regular expression language that can incorporate text, part of speech, and previously assigned tags. An example rule, which states that any noun phrase that is between an operating system kernel name and the word "version" should be tagged as an operating system distribution name, is shown in Listing 3.

Listing 3: Sample rule

```
{
  ruleType: "tokens",
  pattern: ( [{ner:os_kernel_name}]
             ( [ ({pos:NN} | {pos:NNP}) &
                 !{word:/(?i)version/} ] ) ),
  action: (Annotate($1, ner,
           "os_distribution_name")),
  stage: 2
}
```

Since the results of the machine learning-based approach were mediocre, we also implemented a rules-based approach using the Stanford NLP group's TokenRegex system (Chang and Manning 2014). A strong effort was made to establish a set of rules for each training set that was as general as possible while still maximizing F-measure. Of course, the more general the rules are, the more likely they are to conflict with one another at some point. Developing these rule sets was therefore quite a difficult task, with each one taking about two days to create. These rule sets were then used to tag the articles from the test set. The performance is shown in Table 2.

We again see that the performance of the rule set based on current articles drops on the test set, though it remains significantly higher than that of the machine learning-based approach (.73 versus .66 F-measure). Additionally, the performance of the rule set based on older articles, while only a little more than half that of the current rules, is much more reasonable than that of the CRF classifier using the model trained on old articles (.41 versus .02 F-measure). We tried combining the machine learning- and rules-based approaches, but the combined performance was not any better than that of the rules-based approach alone. We therefore did not consider the machine learning-based approach any further for this ontology population task.

## Enhancing Rules with Background Knowledge

As shown in the previous section, the performance of named entity recognition in this domain degrades considerably over time. In this case, the rules created from articles at least ten years older than those being tagged were 44 percent less effective than rules based on contemporaneous articles (.41 versus .73 F-measure). In looking into the specifics of this issue, one underlying problem is that some tags' rules are based on other tags. For example, a computer's make can often be recognized because it follows a computer's manufacturer, as in Lenovo ThinkPad. If Lenovo is not recognized as a computer manufacturer there is a double hit to performance, because not only is that word not tagged correctly, but neither is ThinkPad. The common computer manufacturers a decade ago (e.g. Silicon Graphics, Compaq, DEC) are not common today, which leads to the observed performance degradation. Even more problematic is that over time completely new technologies become relevant to descriptions of computational environments. A good example

| Dataset | Model | Precision | Recall | F-measure |
|---|---|---|---|---|
| Training | Current | 0.93 | 0.93 | 0.93 |
| | Old | 0.93 | 0.93 | 0.93 |
| Test | Current | 0.71 | 0.70 | 0.70 |
| | Old | 0.61 | 0.41 | 0.49 |

Table 3: Performance of a rules-based approach preceded by Wikipedia-based annotations

is GPUs, which have only become popular for parallel processing relatively recently.

Our goal in this work was to reduce the performance degradation seen as rules age by leveraging a source of background knowledge that is continuously updated. In addition, we sought to develop an approach that does not require the types of time-consuming training typical of machine learning and rules based methods. Ideally, the approach should take the ontology (or desired set of tags) as input and require no additional configuration.

The solution we developed uses Wikipedia as the background knowledge source. We developed a custom NER module that fits into the Stanford NLP pipeline. The module takes the list of desired tags as input. We limit this list to the tags that are not related to numeric values (i.e. we omit tags like num-processors and cpu_num-cores) because querying Wikipedia for a number like "4" or "8" is not going to produce any useful information. We also provide common synonyms for tags (i.e. "processor" for CPU and "company" for manufacturer). When tagging a document, the module queries Wikipedia for each proper noun in the key paragraph(s) and if a page is returned, determines which tag, if any, is most relevant by counting the number of times each tag appears in the first three sentences of the document and weighting tags that appear earlier more than those that appear later. After the Wikipedia annotator finishes, the rules-based annotator is run.

Other researchers have also leveraged Wikipedia for various aspects of the NER and ontology population tasks. Many of these efforts are focused on using Wikipedia to do multilingual NER (Nothman et al. 2013; Kim, Toutanova, and Yu 2012). More related to our current work, Kazama and Torisawa determine a candidate tag for an entity by extracting the first noun after a form of the word "be" within the introductory sentence of its Wikipedia page and use that as one of the features in a CRF classifier (Kazama and Torisawa 2007). Klieger et al. take a similar approach but rather than using a classifier, they leverage WordNet to find synonyms and hypernyms of the type identified from Wikipedia in order to arrive at one of the tags of interest (Kliegr et al. 2008). A more thorough survey of NER approaches that utilize Wikipedia can be found in (Zhang and Ciravegna 2011). The key difference between our work and existing approaches in that the results have been analyzed in a way that enables the ability of Wikipedia to mitigate the atrophy of a rules-based technique to be specifically evaluated.

Our method leads to the results shown in Table 3. While this approach is relatively basic, we see that it improves the performance of the old rule set by 20 percent (0.49 versus 0.41 F-measure). The performance of the current rule set is reduced by 4 percent (0.70 versus 0.73 F-measure), indicating that this approach should not be used unless it is warranted by changes in the domain of interest and the age of the model used for ontology population.

## Conclusions and Future Work

In this work we consider the problem of populating an ontology from a fast-changing domain: computational environments. We show that the performance of two popular approaches degrades significantly over time and propose the use of a continuously updated background knowledge source to mitigate this performance degradation. A basic implementation of this idea improved the performance of a rules-based approach based on older documents by 20 percent. It remains to be determined if this technique can achieve similar results in other fast-changing fields, such as medicine or law. In addition, it is possible that this result can be improved upon by a more advanced use of Wikipedia, such as through neural network based methods like word2vec (Mikolov et al. 2013). We plan to explore this in our future work on this topic.

All of the materials used in this project, including the article set, answer set, ontology, machine learning models, rules, and code, have been published to GitHub (https://github.com/mcheatham/compEnv-extraction).

## Acknowledgments

The first and third authors acknowledge partial support by the National Science Foundation under award PHY-1247316 *DASPOS: Data and Software Preservation for Open Science*. The third author would like to acknowledge partial support from Notre Dame's Center for Research Computing.

## References

Bundschus, M.; Dejori, M.; Stetter, M.; Tresp, V.; and Kriegel, H.-P. 2008. Extraction of semantic biomedical relations from text using conditional random fields. *BMC bioinformatics* 9(1):207.

Chang, A. X., and Manning, C. D. 2014. TokensRegex: Defining cascaded regular expressions over tokens. Technical Report CSTR 2014-02, Department of Computer Science, Stanford University.

Cheatham, M.; Charles Vardeman, I.; Karima, N.; and Hitzler, P. 2017. Computational environment: An odp to support finding and recreating computational analyses. In *8th Workshop on Ontology Design and Patterns –WOP2017*.

Chieu, H. L., and Ng, H. T. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th International Conference on Computational Linguistics*, volume 1, 1–7. Association for Computational Linguistics.

Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proceedings of the 43rd Annual*

*Meeting of the Association for Computational Linguistics*, 363–370. Association for Computational Linguistics.

Isozaki, H., and Kazawa, H. 2002. Efficient support vector classifiers for named entity recognition. In *Proceedings of the 19th International Conference on Computational Linguistics*, volume 1, 1–7. Association for Computational Linguistics.

Kazama, J., and Torisawa, K. 2007. Exploiting Wikipedia as external knowledge for named entity recognition. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Kim, S.; Toutanova, K.; and Yu, H. 2012. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, 694–702. Association for Computational Linguistics.

Kliegr, T.; Chandramouli, K.; Nemrava, J.; Svatek, V.; and Izquierdo, E. 2008. Combining image captions and visual analysis for image concept classification. In *Proceedings of the 9th International Workshop on Multimedia Data Mining: held in conjunction with the ACM SIGKDD 2008*, 8–17. ACM.

Kristjansson, T.; Culotta, A.; Viola, P.; and McCallum, A. 2004. Interactive information extraction with constrained conditional random fields. In *AAAI*, volume 4, 412–418.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Manning, C. D.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S. J.; and McClosky, D. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, 55–60.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, 3111–3119.

Nadeau, D., and Sekine, S. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* 30(1):3–26.

Nakashole, N.; Tylenda, T.; and Weikum, G. 2013. Fine-grained semantic typing of emerging entities. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 1488–1497.

Nothman, J.; Ringland, N.; Radford, W.; Murphy, T.; and Curran, J. R. 2013. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence* 194:151–175.

Petasis, G.; Karkaletsis, V.; Paliouras, G.; Krithara, A.; and Zavitsanos, E. 2011. Ontology population and enrichment: State of the art. In *Knowledge-driven multimedia information extraction and ontology evolution*, 134–166. Springer-Verlag.

Zhang, Z., and Ciravegna, F. 2011. Named entity recognition for ontology population using background knowledge from Wikipedia. In *Ontology Learning and Knowledge Discovery Using the Web: Challenges and Recent Advances*. IGI Global. 79–104.

Zhou, L.; Cheatham, M.; Krisnadhi, A.; and Hitzler, P. 2018. A complex alignment benchmark: Geolink dataset. In *International Semantic Web Conference*, 273–288. Springer.