Creating Training Data for Scientific Named Entity Recognition with Minimal Human Effort

Roselyne B. Tchoua[1], Aswathy Ajith[1], Zhi Hong[1], Logan T. Ward[2], Kyle Chard[2,3], Alexander Belikov[6], Debra J. Audus[4], Shrayesh Patel[5], Juan J. de Pablo[5], and Ian T. Foster[1,2,3]

[1] Department of Computer Science, University of Chicago, Chicago, IL, USA
roselyne@uchicago.edu
[2] Globus, University of Chicago, Chicago, IL, USA
[3] Data Science & Learning Division, Argonne National Lab, Lemont, IL, USA
[4] Materials Science and Engineering Division, National Institute of Standards and Technology, Gaithersburg, MD, USA
[5] Institute for Molecular Engineering, University of Chicago, Chicago, IL, USA
[6] Knowledge Lab, University of Chicago, Chicago, IL, USA

Abstract. Scientific Named Entity Referent Extraction is often more complicated than traditional Named Entity Recognition (NER). For example, in polymer science, chemical structure may be encoded in a variety of nonstandard naming conventions, and authors may refer to polymers with conventional names, commonly used names, labels (in lieu of longer names), synonyms, and acronyms. As a result, accurate scientific NER methods are often based on task-specific rules, which are difficult to develop and maintain, and are not easily generalized to other tasks and fields. Machine learning models require substantial expert-annotated data for training. Here we propose polyNER: a semi-automated system for efficient identification of scientific entities in text. PolyNER applies word embedding models to generate entity-rich corpora for productive expert labeling, and then uses the resulting labeled data to bootstrap a context-based word vector classifier. Evaluation on materials science publications shows that the polyNER approach enables improved precision or recall relative to a state-of-the-art chemical entity extraction system at a dramatically lower cost: it required just two hours of expert time, rather than extensive and expensive rule engineering, to achieve that result. This result highlights the potential for human-computer partnership for constructing domain-specific scientific NER systems.

Keywords: Scientific Named Entities · Word Embedding · Natural Language Processing · Crowdsourcing · Polymers.

1 Introduction

There is a pressing need for automated information extraction and machine learning (ML) tools to extract knowledge from the scientific literature. One task that such tools must perform is the identification of entities within text. Many

rule-based, ML, and hybrid named entity recognition (NER) approaches have been developed for particular entity types (e.g., people and places) [23, 20].

Scientific NER remains challenging due to non-standard encoding and the use of multiple entity *referents* (terms used to refer to an entity). For example, in materials science, polymers are encoded in text using various representations (conventional or commonly-used), acronyms, synonyms, and historical terms. Further challenges arise when trying to distinguish between general and specific references to members of polymer families, or recognizing references to blends of two polymers, etc. Such challenges are not unique to polymer science. However, while NER is a well-studied topic in medicine and biology [5, 16], it has only recently become a focus in materials science [10, 29, 17, 36]. Many approaches applied in other domains rely on large, carefully annotated corpora of training data, a luxury not yet available in domains like polymer science.

Here we introduce polyNER, a hybrid computer-human system for semi-automatically identifying scientific entity referents in text. PolyNER operates in three phases, first applying a fully automated analysis to produce an entity-rich set of candidates for labeling; then engaging experts to approve or reject a modest number of proposed candidates; and finally using the resulting labeled candidates to train a classifier. In both the first and third phases, it uses word embedding models to capture shared contexts in which referents occur. PolyNER thus seeks to substitute the labor-intensive processes of either assembling a large manually labeled corpus or defining complex domain-specific rules with a mix of sophisticated automated analysis and focused expert input.

We evaluate polyNER performance on polymer science publications. We compare the output of its first candidate enrichment phase against expert-labeled data, and find that it retrieves 61.2% of the polymers extracted by experts with a precision (26.0%) far higher than the ratio of target entities vs. non-entities in scientific publications (less than 2% in our experience). We evaluate the performance of polyNER overall by comparing its output polymer referents against both expert-labeled data and a state-of-the art rule-based chemical entity extraction system, ChemDataExtractor (CDE) [36], which we have previously enhanced with dictionary- and rule-based methods for identifying polymers [39]. We find that PolyNER can achieve either 52.7% precision or 90.7% recall, depending on user preference, a 10.5% improvement in precision or 22.4% improvement in recall over the enhanced CDE. These results highlight the potential for creating domain-specific scientific NER systems by combining sophisticated automated analysis with focused expert input.

The rest of this paper is as follows. We review scientific NER systems in Section 2. In Section 3, we motivate the need for identifying polymer names in text. We describe design and implementation in Section 4 and evaluate polyNER in Section 5. We summarize and discuss future work in Section 6.

2 Related Work

Natural Language Processing (NLP) is a way for computers to "read" human language. NLP tasks include automatic summarization, topic modeling, translation, named entity recognition (NER), and relationship extraction. NER systems, which aim to identify and categorize named entities (e.g., a person, organization, location), have been developed using both linguistic grammar-based techniques and ML models. While many ML models have been developed, their performance depends critically on the quantity and quality of training data.

Scientific domains such as molecular biology and medical NLP have long used NER for extracting symptoms, diagnoses, medications, etc. from text [5, 16]. More recently, there has also been much interest in chemical entity and drug recognition [10, 29, 17, 36]. However, even state-of-the-art NER systems do not typically perform well when applied to different domains [13]. Considerable effort is involved in selecting and (often manually) generating quality data for trainable statistical NER systems [14].

Various approaches have been proposed to address the lack of training data for NER and other information extraction tasks. *Distant supervision* maps known entities and relations from a structured knowledge base onto unstructured text [26, 43]. However, many fields, including polymer science, lack such knowledge bases.

Data programming uses *labeling functions* (user-defined programs that provide labels for subsets of data) [27]. Errors due to differences in accuracy and conflicts between labeling functions are addressed by learning and modeling the accuracies of the labeling functions. Under certain conditions, data programming achieves results on par with those of supervised learning methods. But while writing concise scripts to define rules may seem to be a more reasonable task for annotators than exhaustively annotating text, it still requires expert guidance. Moreover, labeling functions typically rely on state-of-the-art entity taggers, such as CoreNLP [19], which recognizes persons, locations, organizations and more, and which itself has been trained using various corpora, including the Conference on Computational Natural Language Learning (CoNLL) dataset [32]. A user-defined function may be defined, for example, as: *if the word "married" appears between two PERSONs (as identified by a state-of-the-art named entity tagger), then extract the pair as potential spouses.* Eventually, we will explore using polyNER and data programming to extract polymer properties. For instance: *if a sentence contains a polymer name and the words "glass transition", then extract number(s) in the sentence as potential glass transition temperature(s) for that polymer.*

Other approaches use unskilled or semi-expert users to crowdsource the labeling task [15, 7, 38]. Nonetheless, domain expertise is often crucial for identifying and extracting complex scientific entities. Hence, we ask: how can we quickly generate annotated data for scientific named entity recognition?

3 Motivation

The complexity of scientific NER is primarily due to the fact that entities, for example biological [12] and chemical [14], can be described in different ways, with vocabularies often specialized to small communities. Such issues arise in the polymer science applications that we focus on here. In principle, International Union of Pure and Applied Chemistry (IUPAC) guidelines define polymer naming conventions [9]. However, such guidelines are not always followed in practice [37]. Polymer names may be reported as source-based names (based on the monomer name), structure-based names (based on the repeat unit), common names (requiring domain-specific knowledge), trade names (based on the manufacturer), and names based on chemical groups within the polymer (requiring context to fully specify the chemistry). Oftentimes, polymers are encoded using acronyms.

These different naming conventions arise in part because a desire for clarity in communications is at odds with the often complicated monomeric structures found in many polymers [1]. For example, sequence-defined polymers, where multiple monomers are chemically bound in a well-defined sequence as in proteins, often defy normal naming practices, as it is not possible to list concisely every monomer and their respective positions [18]. Another class of polymers that often suffer from complicated names are conjugated polymers, which exhibit useful optical and electrical properties. Conjugated polymers are complex due to the co-polymerization of multiple monomers (donor/acceptor units), the type and position of side chains along the polymer backbone, and the coupling between monomer units to control regioregularity [8].

Other challenges arise from the use of labels, structure referents (e.g., "micelles," "nanostructures"), and unusual author-coined acronyms. For example, one author defined the acronym DBGA for N,N-dibenzylglycidylamine and then used the string poly(DBGA) to represent poly(N,N-dibenzylglycidylamine). More naming variations result from typographical variants (e.g., alternative uses of hyphens, brackets, spacing) and alternative component orders.

These issues, which make identifying polymeric names a non-trivial exercise not only for computers but also for experts, arise in many fields with specialized vocabularies. Our long-term goal is to build a hybrid human-computer system in which we leverage both human and machine capabilities for the efficient extraction from text of properties associated with specialized vocabularies. In this work, we focus on the task of identifying polymer names.

4 Design and Implementation

As noted in the introduction, previous approaches to scientific NER have relied on large expert-labeled corpora to train NER tools. Our goal in polyNER is to slash the cost of NER training for new domains by using bootstrap methods to optimize the effectiveness and impact of minimal expert labeling. As shown in Figure 2, rather than having experts review entire papers to identify entity

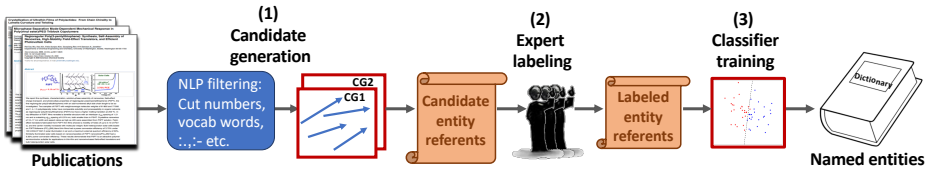

Fig. 1: PolyNER architecture: showing (1) Candidate Generation, which produces candidate named entities from word vectors, (2) Expert Labeling, and (3) Classifier Training, which uses labeled candidates to train supervised ML models for identifying referents.

referents, we use NLP tools to identify a set of promising candidate entity referents (Candidate Generation), then in an Expert Labeling step employ experts to accept or reject those candidates, and finally in a Classifier Training step use the accepted candidates to train an entity classifier

Before turning to the details of the polyNER implementation, we define an NLP filtering process that is used in various places in polyNER to filter out words that are unlikely to be polymer referents. 1) We remove numbers. 2) Hypothesizing that names of scientific entities will not, in general, be English vocabulary words, we remove words found in the SpaCy and NLTK dictionaries of commonly used English words [4, 2]. (We manually remove common polymer names, such as polystyrene and polyethylene, from the dictionaries.) 3) We use SpaCy's part-of-speech tagging functionality to remove non-nouns. 4) We remove unwanted characters (e.g. ':', '.', ',', ':', '-') from the beginning and the end of each candidate, allowing us to recognize, for example, *polyethylene;* (which fails the exact string comparison test against "polyethylene"). 5) We remove plurals (e.g., polyamides, polynorbornenes), as they can represent polymer family names.

4.1 Candidate Generation

This first phase uses word vector representations, vector similarity measures, and minimal domain knowledge to identify a set of high-likelihood ("candidate") entity referents (names, acronyms, synonyms, etc.) in a supplied corpus of full-text documents (in the work presented here, scientific publications).

We first apply the NLP filtering process to reduce false positives. We next face the problem of determining whether a particular string is a polymer referent. String matching only gets us so far: for example, "polyethylene" names a polymer, but "polydispersity" does not. We need also to consider the context in which the string occurs. For example, the polymer name "polystyrene" in a sentence "The melting point of polystyrene is ..." suggests that X may also be a polymer in the sentence "The melting point of X is ...".

NLP researchers have developed a variety of *word embedding* methods for capturing this notion of context. A word embedding method maps each word in a sentence or document to a vector in an n-dimensional real vector space based on the linguistic context in which the word appears. (This mapping may

be based, for example, on co-occurrence frequencies of words.) We can then determine the similarity between two words by computing the distance between their corresponding vectors in the feature space. Such vector representations can be created in many different ways [30, 31]. Recently, the efficient neural network-based Word2Vec has become popular [21, 22].

We consider two measures of context-similarity between word vector representations in this step. CG1 uses the Gensim implementation of the Word2Vec algorithm [28] to generate 100-dimension vectors. CG2 employs an alternative FastText word embedding method that considers sub-word information as well as context [3, 11], allowing it to consider word morphology differences, such as prefixes and suffixes. Sub-word information is especially useful for words for which context information is lacking, as words can still be compared to morphologically-similar existing words. We set the length of the sub-word used for comparison—FastText's *n_gram* parameter—to five characters, based on our intuition that many polymers begin with the prefixes "*poly*" or "*poly(*." FastText produces 120-dimension vectors. Both CG1 and CG2 employ the continuous bag-of-words (CBOW) word embedding, in which a vector representation is generated for each word from an adjustable window of surrounding context words, in any order.

We compute a CG1 (Gensim) vector and a CG2 (FastText) vector for each NLP-filtered word in the input corpus, and also for a small set of representative polymer referents. Here we use polystyrene and its common acronym, PS, based on the assumption that polystyrene, as the most commonly mentioned polymer, provides a large number of example sentences in which polymers are mentioned. We can then determine, for each NLP-filtered word, the extent to which it occurs in a similar context to the representative polymers, by computing the similarities between the word's CG1 and CG2 vectors and those for polystyrene and PS. We discard the lower score for each of CG1 and CG2 to obtain two scores per word.

Having thus obtained scores, we then select as candidates, for each of CG1 and CG2, the N highest-scored words, with N selected based on the time available for experts. We also use a rule-based synonym finder to identify synonyms of generated polymer candidates [33]. For example, if polypropylene has been identified as a candidate, then the expression "*polypropylene (PP)*" leads to PP being added to the candidate list.

4.2 Expert Labeling

The previous step produces a set of candidate polymer referents: NLP-filtered strings that have been determined to occur in similar contexts to our representatives. We next employ an expert polymer scientist to indicate, for each such candidate, whether or not it is in fact a polymer referent. The expert simply approves or rejects each candidate via a simple web interface: a task that is more efficient than reading and annotating words in text. The interface (see Figure 2) provides the expert with example sentences as context for ambiguous candidates, and allows the expert to access the publication(s) in which a particular candidate appears when desired.

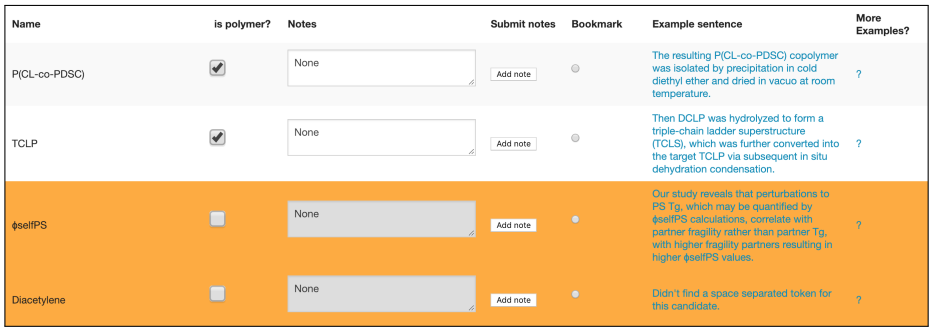

Fig. 2: PolyNER web interface showing annotated candidates. Clicking on "?" delivers up to 25 more example sentences.

4.3 Candidate Discrimination

We next use the expert-labeled data to create an entity classifier. Many classification methods could be applied; we consider three in the work reported here: K Nearest Neighbor (KNN), Support Vector Classifier (SVC), and Random Forest (RF). Previous work has shown that KNNs perform reasonably well in text classification tasks [42]. SVC, an implementation of Support Vector Machines (SVMs), maps data into a feature space in which it can separate the data into two or more sets. RF groups decision trees ("weak learners") to form a strong learner; it produces models that are inspectable, and includes a picture of the most important features. In each case, we use the 100 (Gensim) + 120 (FastText) = 220 dimensions of the two word vectors as input features.

Given limited training and testing data, we evaluate all three classifiers, as implemented within scikit-learn [24], in Section 5.3. We envision that with more annotated data, we will be able to use neural-network-based classifiers.

5 Evaluation

We report on studies in which we evaluate the performance of both the unsupervised Candidate Generation step and various classifiers trained on the labeled data that results from the Candidate Generation and Expert Labeling steps.

5.1 Dataset

We work with two disjoint sets of full-text publications in HTML format from the journal *Macromolecules*: P100 comprising 100 documents with 22 664 sentences and 508 391 (36 293 unique) words or "tokens," and P50 comprising 50 documents with 12 148 sentences and 270 514 (22 571 unique) tokens. For later use in evaluation, we engaged six experts to identify one-word polymer names in P100. They find 467 unique one-word polymer names.

5.2 Evaluation of Candidate Generation Methods

Recall that the polyNER candidate generation module employs two candidate generation methods, CG1 and CG2, plus a rule-based synonym finder. We evaluate the performance of both the complete polyNER candidate generation module and its CG1 and CG2 submodules by comparing the sets of candidates that they each generate from P100, both against the 467 one-word polymer names identified by experts in P100, and against two other polymer name extraction methods, CDE and CDE+, plus a sixth compound method formed by combining polyNER with CDE+. We use exact string-matching between candidates and expert-identified names (all lower cased). Results are in Table 1. For each, we evaluate extraction accuracy in terms of precision, recall, and F_1 score. Recall is the fraction of actual positives that are labeled correctly and precision the fraction of predicted positives that are labeled correctly; F_1, the harmonic average of precision and recall, reaches its best value at 1 and worst at 0.

The first two methods considered, CDE and CDE+, serve as baselines. CDE is a state-of-the-art Python package that extracts chemical named entities and associated properties and relationships from text [36]. As CDE aims to extract all chemical compounds, not just polymers, it serves only as a demonstration of an alternative approach in the absence of a polymer NER system. Its recall is high at 74.5% but its precision is, as expected, low at 8.7%. CDE+ extends CDE with manually defined polymer identification rules [39] to achieve a higher precision of 42.2% but a slightly decreased recall of 68.3%. These results emphasize the difficulty of automatically recognizing complex entities such as polymers.

Rows 3 and 4 show performance for CG1 and CG2 when employed independently. Recall that polyNER performs NLP filtering before applying CG1 and CG2. The filtering step eliminate all but 6878 of the 36 293 unique tokens in P100. Recall also that CG1 and CG2 each assign a score to each of the 6878 remaining words based on their context-based vector similarities to polystyrene and PS, and select the N highest scoring. In this evaluation, we set N=500. CG2, which takes word morphology into account, achieves higher precision and recall than does CG1 (41.8% vs. 15.6% precision and 44.8% recall vs. 16.7% recall for CG2 and CG1, respectively). CG2 retrieves more words starting with "poly" (67% of the 500 candidates vs. only 4% for CG1) while CG1 retrieves more acronyms (38% of the 500 candidates contained more upper than lower case letters, vs. 23% for CG2). CG1 returns more false positives. While character level information is useful for unseen words, or in this case for words lacking context information, we cannot dismiss the use of CG1. Authors often introduce polymer names and subsequently use acronyms more heavily, especially for long names. The facts that CG1 returns more acronyms and that there is likely more context information about acronyms, suggests that the performance of CG1, albeit lower, is solely based on context information.

Row 5 shows results for the complete polyNER candidate generator: that is, the combined CG1 and CG2 candidates plus their rule-based extracted synonyms. This method achieves 61.2% recall and 26.0% precision, producing an entity-rich set of candidates without any domain-specific rules and without

any (tedious, time-consuming, and costly) expert-annotated corpus of polymer names. We are encouraged to observe that polyNER retrieves polymers not extracted by CDE: the combined recall for PolyNER ∪ CDE+ (row 6 in the table) is 81.6%—higher than CDE itself. This result suggests that polyNER's candidate generation module can be used not only to annotate automatically a diverse set of polymers based on context, but also to improve on the results of more sophisticated hybrid rule- and ML-based NER tools.

Figure 3, which shows every word in P100 in FastText vector space, illustrates the challenges and opportunities inherent in differentiating between polymer and non polymer word vectors. The polymer names (in red and green) form two rather diffuse clusters that overlap considerably with non polymers (in blue). Interestingly, the subset of polymer names that are acronyms (the red points) are clearly clustered.

Table 1: Results when polymer candidates are extracted from our test corpus, P100, via different methods. For each, we show true positives, false positives, false negatives, precision, recall, and F-score.

#	**Method**	**Total**	**TP**	**FP**	**FN**	**Precision**	**Recall**	**F_1**
1	CDE	3994	348	3646	119	8.7%	74.5%	15.6%
2	CDE+	755	319	436	148	42.2%	68.3%	52.2%
3	CG1	500	78	422	389	15.6%	16.7%	16.1%
4	CG2	500	209	291	258	41.8%	44.7%	43.2%
5	PolyNER	1099	286	813	181	26.0%	61.2%	36.5%
6	PolyNER ∪ CDE+	1495	381	1114	86	25.4%	81.6%	38.8%

5.3 Evaluation of Classifier Training Methods

We next evaluate how well classifiers trained on expert-labeled output from the Candidate Generation phase perform when applied to full-text documents. Here, we make use of our second dataset, P50, to train and test our classifiers, and P100 to validate the trained classifiers.

Before training our classifiers, we need a set of expert-labeled candidates. Thus we first apply the CG1 and CG2 methods of Section 4.1 to P50, generating a total of 897 unique candidates: 500 for CG2 and 466 from CG1, of which 69 overlapped. (We do not apply the rule-based synonym finder here.) Then we employ an expert to label as polymer or non-polymer each of those 897 candidates, producing a new dataset that we refer to as P50-labeled. Note that this task is quick work for the expert, as only 897 words need to be evaluated: the total time required was two hours. This expert review identifies 260 (29.0%) of the 897 as polymers.

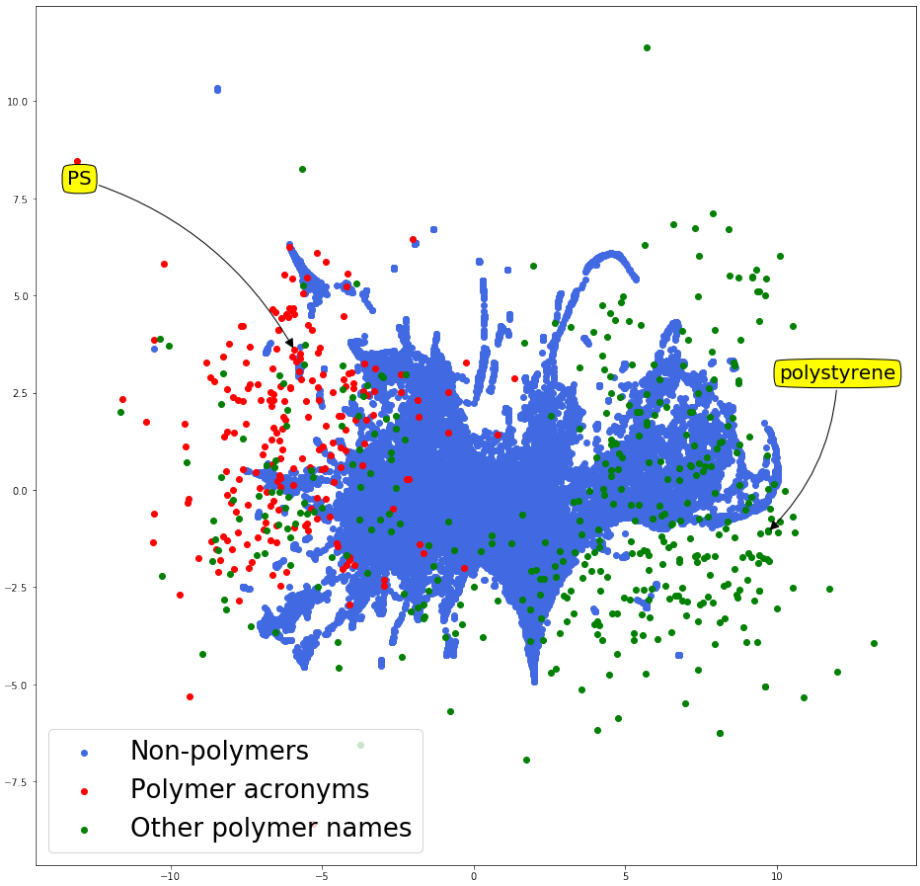

Fig. 3: A two-dimensional representation of all words in P100, generated with the scikit-learn implementation of t-distributed Stochastic Neighbor Embedding (t-SNE) [41]. Of the words identified by experts as polymers, we show acronyms in red and non-acronyms in green; all other words are blue. We label our two representative words. (The t-SNE plot, a dimensionality reduction technique used to graphically simplify large datasets, reduces the 120-dimensional vectors to two-dimensional data points. The axes have no "global" meaning.)

Training and validating the classifiers: We next use the 897 expert-labeled words to train our three classifiers. We use 90% (807) for training and hold out 10% (90) for validation.

The left-hand side of Table 2 shows the performance of the different trained classifiers when applied to the P50 hold-out words. Note that performance here is defined with respect to how well the classifier does at predicting the expert labels assigned to the polyNER-generated candidates—not how well the classifier identifies *all* polymer referents in P50, as we do not have the latter information.

All three classifiers obtain between 66.7% and 100.0% precision and between 28.6% and 57.1% recall. SVC achieves the highest recall and F_1 score. The lower part of the table ("combined classifiers") shows that combined classifiers can improve performance. The 3-of-3 method achieves the highest precision (100.0%) but lowest recall, as one might expect. The $\geq$2 method also achieves 100.0% precision but with a higher recall (42.3% vs 28.6%). The $\geq$1 method has the lowest precision but the highest recall at 57.1%.

Table 2: Results when various classifiers (trained on expert-labeled P50 candidates) are applied to P50 holdouts (left) and P100 (right). The results in the bottom two rows are copied from Table 1 for ease of comparison.

Classifier	**Validation on P50 holdouts**			**Testing on P100**		
	Precision	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
KNN	75.0%	42.8%	54.5%	9.5%	77.8%	16.9%
SVC	66.7%	57.1%	61.5%	16.8%	76.5%	27.5%
RF	100.0%	28.6%	44.4%	51.0%	42.0%	46.1%
Combined						
3-of-3	100.0%	28.6%	44.4%	52.7%	39.1%	44.9%
$\geq$2	100.0%	42.3%	60.0%	22.8%	66.6%	34.0%
$\geq$1	57.1%	57.1%	57.1%	9.1%	90.7%	16.5%
CDE				8.7%	74.5%	15.6%
CDE+				42.2%	68.3%	52.2%

Testing the trained classifiers: We test the trained classifiers by applying each to all 6878 NLP-filtered nouns extracted from P100 and comparing the resulting polymer/non-polymer labels against our ground truth of polymer names extracted from P100 by experts. Results are in the right-hand side of Table 2. RF achieves the highest F_1 score (46.1%) with 51.0% precision and 42.1% recall. While recall is relatively low (fewer entities retrieved), precision is significantly better than that achieved by CDE+. We observe also that combined classifiers can improve precision (52.7%) at the expense of recall, or significantly increase recall (90.1%) at the expense of precision. Users can thus trade off precision and recall, in each case exceeding those achieved by the rule-based CDE+ system.

5.4 Discussion

These results are based on only limited training data: just 897 labeled words, of which 260 are polymers. We view the effectiveness of the classifiers trained with these limited data as demonstrating the feasibility of using small amounts of expert-labeled data to bootstrap context-aware word-vector classifiers. Importantly, this whole process was both inexpensive and generalizable to other domains. Candidate generation was fully automated and involved no domain knowledge besides the two representative words, polystyrene and PS. Labeling required just two hours of an expert's time. Classifier training was again automated and involved no domain knowledge.

6 Conclusion

Despite much progress in NLP, scientific named entity recognition (NER) remains a research challenge. A lack of labeled training data in fields such as polymer science limits the use of machine learning models for this task. PolyNER is a generalizable system that can efficiently retrieve and classify scientific named entities. It uses word representations and minimal domain knowledge (a few representative entities) to produce a small set of candidates for expert labeling; labeled candidates are then used to train named entity classifiers.

PolyNER can achieve either 52.7% precision or 90.7% recall when combining classifiers: a 10.5% improvement in precision or 22.4% in recall over a well-performing hybrid NER model (CDE+) that combines a dictionary, expert created rules, and machine learning algorithms. PolyNER's architecture allows users to tradeoff precision and recall by selecting which classifiers are used for discrimination. One out of every four candidates identified by our current polyNER prototype is in fact a polymer. This enrichment relative to the relative paucity of polymers in publications significantly reduces the effort required by experts. Considering that polyNER relies on simple distance from a known polymer(s), and default word embedding parameters, this result is encouraging.

An important issue to explore in future work is whether classifier performance can be improved by providing additional expert-labeled words. We plan to apply active learning [34] to select good candidates. As we generate more expert-labeled candidates, we will explore the use of neural network word vector classifiers to improve accuracy, and the use of polyNER-labeled data to annotate text for other NER approaches, such as bidirectional long short-term memory models. With a view to exploring generalizability, we are also working to apply polyNER to quite different problems, such as extracting dataset names from social science literature. We may explore more recent word representation models, which are pre-trained on large corpora [25, 6].

Acknowledgments

We thank Mark DiTusa, Tengzhou Ma, and Garrett Grocke for contributing manually extracted polymer names. This work was supported in part by NIST

contract 60NANB15D077, the Center for Hierarchical Materials Design, and DOE contract DE-AC02-06CH11357, and by computer resources provided by Jetstream [40, 35]. Official contribution of the National Institute of Standards and Technology; not subject to copyright in the US.

References

1. Audus, D. J. & de Pablo, J. J.: Polymer informatics: Opportunities and challenges. ACS Macro Letters **6**(10), 1078–1082 (2017)
2. Bird, S. & Loper, E.: NLTK: The natural language toolkit. In: 42nd Annual Meeting of the Association for Computational Linguistics. p. 31 (2004)
3. Bojanowski, P. et al.: Enriching word vectors with subword information. arXiv:1607.04606 (2016)
4. Choi, J. D. et al.: It depends: Dependency parser comparison using a web-based evaluation tool. In: 53rd Annual Meeting of the ACL. vol. 1, pp. 387–396 (2015)
5. Cohen, A. M. & Hersh, W. R.: A survey of current work in biomedical text mining. Briefings in Bioinformatics **6**(1), 57–71 (2005)
6. Devlin, J. et al.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805 (2018)
7. Gao, H. et al.: Harnessing the crowdsourcing power of social media for disaster relief. IEEE Intelligent Systems **26**(3), 10–14 (2011)
8. Himmelberger, S. & Salleo, A.: Engineering semiconducting polymers for efficient charge transport. MRS Communications **5**(3), 383–395 (2015)
9. Hiorns, R. C. et al.: A brief guide to polymer nomenclature. Polymer **54**(1), 3–4 (2013)
10. Jessop, D. M. et al.: OSCAR4: A flexible architecture for chemical text-mining. Journal of Cheminformatics **3**(1), 41 (2011)
11. Joulin, A. et al.: Bag of tricks for efficient text classification. arXiv:1607.01759 (2016)
12. Kim, J.-D. et al.: Introduction to the bio-entity recognition task at JNLPBA. In: Intl. Joint Workshop on NLP in Biomedicine and its Applications. pp. 70–75 (2004)
13. Krallinger, M. et al.: Overview of the chemical compound and drug name recognition (CHEMDNER) task. In: BioCreative Challenge Evaluation Workshop. vol. 2, p. 2 (2013)
14. Krallinger, M. et al.: CHEMDNER: The drugs and chemical names extraction challenge. Journal of Cheminformatics **7**(1), S1 (2015)
15. Krishna, R. et al.: Visual genome: Connecting language and vision using crowdsourced dense image annotations. Intl. J. Computer Vision **123**(1), 32–73 (2017)
16. Leaman, R. & Gonzalez, G.: BANNER: An executable survey of advances in biomedical named entity recognition. In: Pac. Symp. Bio., pp. 652–663 (2008)
17. Leaman, R. et al.: tmChem: A high performance approach for chemical named entity recognition and normalization. Journal of Cheminformatics **7**(1), S3 (2015)
18. Lutz, J.-F.: Aperiodic copolymers. ACS Macro Letters **3**(10), 1020–1023 (2014)
19. Manning, C. D. et al.: The Stanford CoreNLP natural language processing toolkit. In: ACL (System Demonstrations). pp. 55–60 (2014)
20. Marrero, M. et al.: Named entity recognition: fallacies, challenges and opportunities. Computer Standards & Interfaces **35**(5), 482–489 (2013)
21. Mikolov, T. et al.: Efficient estimation of word representations in vector space. arXiv:1301.3781 (2013)

22. Mikolov, T. et al.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Inf. Processing Sys. pp. 3111–3119 (2013)
23. Nadeau, D. & Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investigationes **30**(1), 3–26 (2007)
24. Pedregosa, F. et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
25. Peters, M. E. et al.: Deep contextualized word representations. In: Conf. of the North American Chapter of the Association for Computational Linguistics (2018)
26. Peters, S. E. et al.: A machine reading system for assembling synthetic paleontological databases. PLoS One **9**(12), e113523 (2014)
27. Ratner, A. J. et al.: Data programming: Creating large training sets, quickly. In: Advances in Neural Information Processing Systems. pp. 3567–3575 (2016)
28. Rehurek, R. & Sojka, P.: Software framework for topic modelling with large corpora. In: Workshop on New Challenges for NLP Frameworks (2010)
29. Rocktäschel, T. et al.: ChemSpot: A hybrid system for chemical named entity recognition. Bioinformatics **28**(12), 1633–1640 (2012)
30. Rumelhart, D. E.: Learning internal representations by back-propagating errors. Parallel Distributed Processing **1**, 318–362 (1986)
31. Sabes, P. N. & Jordan, M. I.: Reinforcement learning by probability matching. In: Advances in Neural Information Processing Systems. pp. 1080–1086 (1995)
32. Sang, E. F. T. K. & De Meulder, F.: Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In: 7th Conference on Natural Language Learning. pp. 142–147 (2003)
33. Schwartz, A. S. & Hearst, M. A.: A simple algorithm for identifying abbreviation definitions in biomedical text. In: Pac. Symp. Bio., pp. 451–462 (2002)
34. Settles, B.: Active learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **6**(1), 1–114 (2012)
35. Stewart, C. A. et al.: Jetstream: A self-provisoned, scalable science and engineering cloud environment (2015), `https://doi.org/10.1145/2792745.2792774`
36. Swain, M. C. & Cole, J. M.: ChemDataExtractor: A toolkit for automated extraction of chemical information from the scientific literature. Journal of Chemical Information and Modeling **56**(10), 1894–1904 (2016)
37. Tamames, J. & Valencia, A.: The success (or not) of HUGO nomenclature. Genome Biology **7**(5), 402 (2006)
38. Tchoua, R. B. et al.: A hybrid human-computer approach to the extraction of scientific facts from the literature. Procedia Computer Science **80**, 386–397 (2016)
39. Tchoua, R. B. et al.: Towards a hybrid human-computer scientific information extraction pipeline. In: 13th Intl. Conference on e-Science. pp. 109–118 (2017)
40. Towns, J. et al.: XSEDE: Accelerating scientific discovery. Computing in Science & Engineering **16**(5), 62–74 (2014)
41. van der Maaten, L. J. P. & Hinton, G.: Visualizing data using t-SNE. Journal of Machine Learning Research **9**(Nov), 2579–2605 (2008)
42. Yang, Y. & Liu, X.: A re-examination of text categorization methods. In: 22nd Annual International ACM SIGIR Conference. pp. 42–49. ACM (1999)
43. Zhang, C. et al.: GeoDeepDive: Statistical inference using familiar data-processing languages. In: ACM SIGMOD Conference. pp. 993–996 (2013)