Active Learning Yields Better Training Data for Scientific Named Entity Recognition

Roselyne B. Tchoua*, Zhi Hong*, Logan T. Ward*‡,
Kyle Chard†‡, Debra J. Audus§, Shrayesh N. Patel¶, Juan J. de Pablo¶ and Ian T. Foster*†‡
*Department of Computer Science, University of Chicago, Chicago, IL, USA
†Globus, University of Chicago, Chicago, IL, USA
‡Data Science and Learning Division, Argonne National Laboratory, Argonne, IL, USA
§Materials Science and Engineering Division, National Institute of Standards and Technology, Gaithersburg, MD, USA
¶Institute for Molecular Engineering, University of Chicago, Chicago, IL, USA
Email: roselyne@uchicago.edu

***Abstract*—Despite significant progress in natural language processing, machine learning models require substantial expert-annotated training data to perform well in tasks such as named entity recognition (NER) and entity relations extraction. Furthermore, NER is often more complicated when working with scientific text. For example, in polymer science, chemical structure may be encoded using nonstandard naming conventions, the same concept can be expressed using many different terms (synonymy), and authors may refer to polymers with ad-hoc labels. These challenges, which are not unique to polymer science, make it difficult to generate training data, as specialized skills are needed to label text correctly. We have previously designed 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 classifier. PolyNER facilitates a labeling process that is otherwise tedious and expensive. Here, we use active learning to efficiently obtain more annotations from experts and improve performance. Our approach requires just five hours of expert time to achieve discrimination capacity comparable to that of a state-of-the-art chemical natural language processing toolkit, highlighting the potential for human-computer partnership in domain-specific scientific NER.**

***Index Terms*—Named Entity Recognition, Machine Learning, Word Embedding, Active Learning, Polymers**

I. Introduction

A wealth of valuable research data is published in unstructured form in millions of scientific articles each year. Reading and extracting pertinent information from those articles has become an unmanageable task for scientists and makes it hard to build on existing results. A major obstacle to scientific fact extraction is the difficulty of identifying scientific entities in text. Despite much progress in natural language processing (NLP), scientific named entity recognition (NER) remains a research challenge. The main reason for this gap between NLP advances and scientific extraction needs is the lack of carefully annotated datasets for specific targets. In standard NER, progress is made possible by, for example, the Conference on Computational Natural Language Learning (CoNLL) dataset, which supports much work that advances the state of the art. But NER systems trained on CoNNL data do not perform well for scientific text, due to the distinctive vocabularies used in different scientific disciplines and subdisciplines.

Science-specific training datasets have been established in biology [1] and, more recently, chemistry [2]. However, the expert effort required to design extraction schema, define clear annotation rules, and generate training data is substantial, and cannot feasibly be performed for every field of science. As a consequence, annotated datasets do not exist in most fields, preventing the application of state-of-the-art NER and fact extraction methods.

This problem is apparent in materials science, where materials informatics seeks to combine large datasets and computational models to replace current trial-and-error materials design processes with targeted materials design, thus reducing time-to-market and development costs of new materials [3]. Unfortunately, the lack of annotated training data has hindered progress, preventing large-scale application of NER methods pioneered in, for example, biomedicine. Those methods, which often use hybrid rule-based, machine learning, and statistical techniques to extract entity names and relations from the literature [4, 5], require much training data. While similar efforts have begun in materials science [6–10], the lack of available training data impedes rapid progress. Instead, each new research project targeting a new type of material, property, or relation must first undertake considerable effort to create a large, carefully annotated training set tailored for this new target, a task that often requires considerable in-depth domain knowledge.

The subfield of polymer science puts these problems in particularly stark relief. Polymers have their own unique nomenclature, as we explain below, and thus annotated datasets created for general chemistry are of little value. Scientists and engineers lack access to a freely available large database of polymers and their properties. To address these challenges, we have previously designed polyNER [11, 12], a system for generating training data for scientific NER using semi-supervised and supervised learning. PolyNER uses NLP to produce sets of candidate entities, which experts approve or reject via a Web interface; the resulting labels are used to train context-based word vector classifiers. PolyNER's labels can

also be used to train other machine learning models to leverage other features in addition to context, such as word morphology to recognize target entities. The goal is to substitute the labor-intensive processes of assembling a large manually annotated corpus (and reduce costs) by using small numbers of carefully selected candidates to be labeled via focused expert input. We aim in our work to slash the expert time and effort required to achieve state-of-the-art NER performance in a new field.

In this paper, we seek to improve polyNER's performance by improving the labeling process and the classification of entity word vectors. Specifically, we address two challenges: (1) lack of (expensive) training data in some fields, including our own polymer science application which lacks free access to large polymer databases; and (2) the need for domain expert curators, which impedes the use of crowdsourcing platforms such as Amazon Mechanical Turk [13] or Figure8 (https://www.figure-eight.com/). In the initial labeling phase, we experiment with different *representative* (commonly used) entities to increase the fraction of target entities in the dataset to be labeled and bootstrap the context-based word vector classifiers. In the subsequent labeling and classification steps, we use active learning with maximum entropy uncertainty sampling and two different pools of unlabeled data to train classifiers, and compare their learning rate after five rounds. Using labels generated via active learning, we train word vector classifiers and achieve NER performance comparable to a modified version of a state-of-the art rule-based chemical entity extraction system, ChemDataExtractor (CDE) [9]. We have previously enhanced CDE with dictionary- and rule-based methods for identifying polymers [14]. Our system, however, took five hours of expert time to achieve this result.

The rest of this paper is as follows. In Section II, we motivate the need for identifying polymer names in text. We review semi-supervised methods for NLP systems in Section III. We describe the design and implementation of our active learning-based approach in Section IV. We evaluate our approach in Section V. We summarize and discuss future work in Section VI.

II. Motivation

Our work is generally motivated by the need to extract previously unmined scientific entities; our initial goal is to enable machine-learned extraction of polymer names. The challenges of polymer science NER are similar to those in biomedicine [2, 15]. Entities can be described with multiple referents (synonymy). Conversely, the same word may refer to different concepts depending on context (polysemy). For example, *polystyrene* is often referred to as *PS*, but *polystyrenes* can also be referred to as *GPPS*, *HIPS*, and *EPS*; combined with other monomers yielding *SBR*, *SBS*, and *ABS*; or used to describe polystyrene derivatives such as *PAMS*, *PMS*, and *PSS*. While standards for naming polymers exist (e.g., International Union of Pure and Applied Chemistry (IUPAC [16]) naming conventions), they are not always followed in practice [17]. Instead, 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), generating variability in naming conventions. Typographical variants (e.g., alternative uses of hyphens, brackets, spacing) and alternative component orders cause more variations between polymer names in the literature. The origin of these different naming conventions is linked to the desire for clarity within a journal article, coupled with the often-complicated monomeric structures [18].

In addition to challenges related to the makeup of the scientific entities, the scarcity of entities in scientific literature and the lack of training data also impede the use of machine learning based NER techniques. Considerable time and manual effort are involved in creating and maintaining the balanced CoNLL dataset for standard NER [19]. Example sentences from such corpora include one or more entities per sentence. In our attempt to recognize polymer names in full text documents, we face a very imbalanced dataset where most sentences do not contain a target entity, as there are only a handful of target entities per document. While there has been much interest in machine learned recognition of biochemical entities [7–9, 20], the successes that have been achieved have required much human effort to generate quality training data [2]. Previous work has also found that even state-of-the-art NER systems rarely perform well when applied to different domains [21]. Therefore, the problem of training machine learning models to recognize new scientific entities in a new field, such as polymer science, remains challenging.

III. Related Work

NER and other information extraction tasks rely on a large amount of training data, which are expensive to obtain. Weakly supervised learning methods work with much less training data and aim to address this challenge. They generally fall under two categories: semi-supervised learning and active learning [22]. The key difference between the two is that the former relies on approximately labeled data (as opposed to correctly labeled data for supervised learning) and the latter starts off with unlabeled data. Semi-supervised learning attempts to label data automatically by using prior knowledge and a set of labeled data. For example, it assumes that if x and y are similar, they probably have the same label (first cluster the whole dataset, then label each cluster with labeled data [23]). Active learning assumes there is a source of knowledge, such as a human expert, that can be queried to label a selected batch of unlabeled data.

A. Semi-supervised Approaches

Bootstrapping is a semi-supervised technique, which starts from a small set of seed relation instances and iteratively learns more relation instances and extraction patterns. Snowball [24] which improved the DIPRE system [25], used an intuitive idea to collect new entity relations using a set of seed entity pairs. In the DIPRE system, the intuitive assumption is that, given a few seed entity relations, the text between two known

target entities in close proximity of each other describes and constitutes a *pattern* of the relation between the two. Since that is not the case in practice, the system uses a limited set of regular expressions to limit useful patterns, hence decreasing the number of false positives. A key improvement of Snowball is that its patterns include named entity tags (PERSON, LOCATION, ORGANIZATION, etc.). Given a handful of seed tuples of ORGANIZATION and LOCATION, Snowball attempts to learn the relation *HeadquarteredIn* by assuming that each time the tuples appear in close proximity to each other, the text in between illustrates the desired relation. This text can then be used to discover new tuples, which can in turn be used as seeds for the next discovery round. Of course, organizations may be located but not headquartered in multiple cities; hence it is important to inspect the quality of extraction patterns to reduce noise in the generated output.

Distant supervision maps known entities and relations from a structured knowledge base onto unstructured text [26, 27]. With freely available structured knowledge base such as DBPedia [28] and Freebase [29], it is possible to leverage a large set of known entity pairs to generate training data. Over the past decade, probabilistic approaches have been proposed to allow automatic selection. For example, PaleoDeepDive [26], built upon DeepDive [27], automatically extracts paleontological data from text, tables, and figures in scientific publications. For good performance, such approaches often rely on and extend large databases: for example, PaleoDeepDive uses PaleoDB [30]. The system labels any entity pair that appears in the database as *True*. [Kyle: Following sentence is broken (and missing a bracket] The user defines features (e.g. if a specific keyword appears between two entities, that pair a certain attribute is labeled *True*, but if the entity pairs are too far apart, another attribute is marked *False*, the system then uses statistical inference to determine the probability that each newly discovered pair of interest is *True*.

Data programming, as used in the Snorkel system[Kyle: citation?], has users define *labeling functions* to provide labels for data subsets [31]. 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. 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. In data programming as in bootstrapping and distant supervision it is important to evaluate the quality of functions and extraction patterns to decrease noisy patterns.

B. Active Learning

Active learning [22] assumes that gold standard labels for unlabeled instances can be obtained by querying an oracle (domain expert or source of knowledge). The goal of active learning is to decrease labeling costs by requesting a limited number of labels from the oracle, that have been deemed most valuable by the learner. Uncertainty sampling approaches define "valuable" data by measuring uncertainty in the predictions. For example, in the case of a single learner, querying predictions with maximum entropy in which the learner assigns all classes with equal probability [32] or predictions closest to the decision boundary in the case of support vector machine classifiers [33]. In the case of multiple learners, query-by-committee requests labels for unlabeled instances on which the learners disagree the most [34]. Uncertainty sampling and query-by-committee are representative approaches based on informativeness, where informativeness measures how well an unlabeled instance helps reduce the uncertainty. Another selection criterion addresses representativeness, which measures how well an instance helps represent the structure of input patterns; in this case selection is made by querying data from unlabeled clusters of data [35, 36].

C. CDE and CDE+

We introduce briefly ChemDataExtractor (CDE), a state-of-the-art chemical NLP tool that combines a dictionary, expert-created rules, and machine learning algorithms. CDE was trained on the CHEMDNER corpus: a collection of 10 000 PubMed abstracts with 84 355 chemical entity mentions labeled manually by expert chemistry literature curators, following annotation guidelines specifically defined for this task [2]. In previous work, we modified CDE with manually defined polymer identification rules [14], creating what we term here CDE+. We compare our methods against both CDE and CDE+ in Section V.

D. Placing polyNER in Context

[Kyle: Much of this text is repeated in the following section. I wonder if we should delete here and move anything that is here now, but not in the next section, into the next section.] PolyNER has in common with prior work that it combines semi-supervised and active learning [35, 37]. The initial sampling of unlabeled data is similar to bootstrapping but applied to named entities rather than entity pairs. The initial batch of labels contains strings deemed *similar* to a few seed entities, with similarity determined by using word representations and vector distance measures. As this approach is approximate and is likely to include errors (words that have similar context to entities but are not actual entities), the next phase is expert labeling. Subsequent batches of labels are obtained via active learning and used to train a context-aware word vector classifier in the last phrase. PolyNER can be used in a way that complements other scientific NER approaches. For example, it could be used as a scientific entity tagger (i.e., recognizer) to be used with data programming to extract polymer properties.

IV. Design and Implementation

As previously mentioned, our main goal in designing polyNER is to slash labeling costs by reducing the time and effort spent by experts to generate training data. Rather than labeling entire documents and phrases, annotators label proposed candidate entities to be classified. Earlier results

show that with two hours of labeling we can achieve precision or recall (but not both) on par with state-of-the-art domain specific software, by selecting an ensemble of classifiers for discrimination [12].

Here, we refine polyNER components and incorporate active learning with different sampling strategies in order to further improve performance. PolyNER uses word representations and minimal domain knowledge (a few seed entities) to produce a small set of candidates for expert labeling; labeled candidates are then used to train named entity word vector classifiers. We integrate an active learning loop into polyNER's architecture to incrementally improve classifier performance.

In order to explore whether the use of word vector coordinates as features can accelerate the learning process, we define and compare three alternative sampling strategies: a random strategy that we use as a baseline, and two NLP-based filtering methods. We also apply these methods against two different candidate pools, one set of unlabeled nouns and another set of approximately labeled nouns deemed *similar* to commonly used known entities from our corpus. We describe our sampling strategies and approximate labeling in more details in this section. The general architecture of polyNER is illustrated in Figure 1. We also describe the labeling process, and the training and testing configuration for our word vector classifiers in Section IV-F

A. Computing Word Embeddings

A word embedding method maps each word in a document to a vector in an n-dimensional real vector space that represents 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.

We use Word2Vec, a recent, light-weight and easy-to-use implementation of context-based vector representations [38, 39]. Specifically, we use the Gensim continuous bag-of-words (CBOW) implementation of the Word2Vec algorithm [40] to generate vectors. Based on prior parameter tuning we set the Word2Vec `size` parameter to 100 and the `window_size` to 2; where `size` is the size of the vector, and `window_size` is an adjustable window of surrounding context word used to compute each embedding.

B. NLP-Filtering

The NLP filtering preprocessing step removes strings that are unlikely to be polymer referents. First, we remove unwanted characters (e.g. ':', '.', ',', ':', '-') from the beginning and the end of each candidate, and eliminate numbers (including numbers followed by common units). This step allows us to recognize, for example, "polyethylene;" (with the extraneous ';') as a duplicate of "polyethylene". Hypothesizing that names of scientific entities will not, in general, be English vocabulary words, we also remove words found in the SpaCy dictionaries of commonly used English words [41]. We manually remove common polymer names, such as polystyrene and polyethylene, from the dictionaries. We also use SpaCy's part-of-speech tagging functionality to remove non-nouns. Finally, we remove plurals (e.g., polyamides, polynorbornenes), as they can represent polymer family names. Note that these steps are generalizable and applicable to multiple science fields. We refer to the set of words that results when these filtering operations are applied to our corpus as the *NLP-filtered candidates*. This set is the output of step 1 in Figure 1.

C. Initial Bootstrapping and Labeling

NLP filtering reduces the number of entities to be considered, and increases the target vs. non-target entity ratio. However, there still remain a large pool of potential candidates from which entities are to be selected, of which, in our experience, roughly 5% are polymer names. In order to avoid presenting experts with mostly negative examples, hindering meaningful classification, we boost the number of polymer entities in the first batch of candidates to be annotated by selecting strings with low word vector distance (see Section IV-A) from a set of *seed entities*: words that are observed to occur frequently in a subset of publications, or that are suggested by experts. We discuss this distance metric in more detail below. Based on preliminary experiments, we set the size of each batch of strings to be labeled to 200, or about an hour of expert time. We then train the initial classifier on this bootstrap set, using 80% of the data for training and 20% for testing. We subsequently used three different sampling strategies for following classifications.

D. Sampling Strategies

We implement three sampling strategies, which we refer to as *Random*, *Uncertainty-Based Sampling (UBS)* and *Distance Uncertainty-Based Sampling (Distance UBS)*. We apply each of these strategies to our NLP-filtered candidates to determine which candidates to present to experts for labeling.

1) Random: Here, we randomly select 200 of the NLP-filtered candidates.

2) Uncertainty-Based Sampling (UBS): Our second strategy applies maximum entropy sampling to the NLP-filtered candidates. As previously mentioned, maximum entropy selection is an uncertainty sampling method that identifies data points for which a classifier predicts outcomes that lie near the decision boundary between classes. Thus, when predicting whether or not a word vector represents a polymer, maximum entropy arises when the classifier assigns equal probability to the polymer and not-polymer cases. As we have two classes, this equal probability is 0.5. We use the classifier to obtain a probability p for each NLP-filtered candidate. We select the 200 entries for which p is closest to 0.5 as our sample.

3) Uncertainty-Based Sampling with Distance Ranking (Distance UBS): Our third strategy is identical to UBS, except that it works with just a subset of the NLP-filtered candidates, namely the 10 000 that are closest to a set of seed entities. The intuition here is that a candidate is more likely to be a target referent (a name, acronym, synonym, etc.) if it used in a similar context. For example, the polymer name "polystyrene"

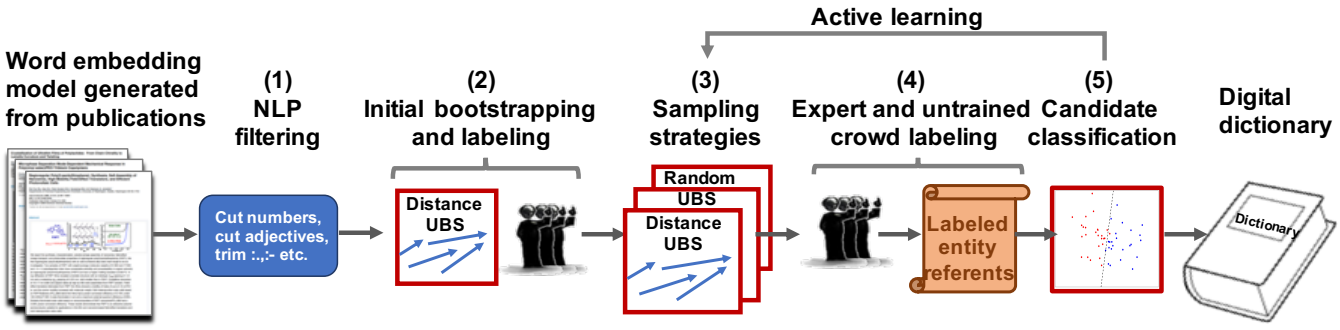

Fig. 1: PolyNER system showing the different phases of polyNER including the NLP-filtering step, the initial bootstrapping and labeling phase as well as the newly integrated active learning loop to classify scientific named entities.

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 ...".

We use the word embeddings introduced in Section IV-A to capture this notion of context, and vector distance between word vectors as a measure of similarity. Whether or not this approach works in practice will depend on whether polymer names are in fact used in consistent contexts as captured by our word embedding vectors.

We can then determine, for each NLP-filtered word, the extent to which it occurs in a similar context to the seed entities, by computing the similarities between the word's vector and those for our seed entities. As we explain in Section V-B, we experiment with one and more seed entities; when dealing with multiple seed entities, we use the lowest distance score for ranking candidates.

E. Bootstrapping

The UBS and Distance UBS sampling methods use a classifier to determine which NLP-filtered entities should be chosen next for expert labeling. This classifier must be trained, and thus we need an initial set of entities to bootstrap this process. We could choose NLP-filtered entities at random for initial labeling, but that choice is unlikely to perform well due to the low proportion of polymers (just 5%) in the NLP-filtered corpus. Instead, therefore, we create an initial bootstrap set comprising the 200 NLP-filtered entities that are closest, in word distance vectors, to a set of seed entities. We then train the initial classifier on this bootstrap set, using 80% of the data for training and 20% for testing.

F. Active Learning Loop

We now discuss our active learning process. As discussed in Section III-B, the basic idea here is that we repeatedly select a set of 200 candidate entities (a "sample") for expert labeling, based on what we have learned from previously labeled entities. We run this process independently with the Random, UBS, and Distance UBS sampling strategies, in order to compare their performance.

1) Use and Evaluation of Classifiers: Not specified in Section IV-D is the nature of the classifier that the UBS and Distance UBS strategies use to estimate the probability of each entity in the NLP-filtered corpus (or, for Distance UBS, the 10 000-entity subset of the NLP-filtered corpus) being a polymer. (The Random strategy does not use a classifier for sampling, as it selects candidates at random.) As we have no prior knowledge of the distribution of target entities in the vector space, we consider seven distinct classifiers in each round of the active learning process: the scikit-Learn [42] implementations of Decision Tree, Gradient Boosting, K-Nearest Neighbor (KNN), Logistic Regression, Linear Support Vector Machine, Naive Bayes, and Random Forest. In each case, we use the word embedding for each string as input features. In each round i, we train these seven classifiers on the sample data gathered in rounds j, $j<i$, and then use the classifier with the highest recall to determine the p scores that UBS and Distance UBS use when assembling their 200-entity sample in that step. (We use recall, or retrieving a maximum of targets, as a measure of performance, because we want to favor extracting a higher number of targets, potentially requiring additional curation, over obtaining fewer correct targets.)

The 200 entities in the new sample are passed to experts for annotation, and the annotated data are added to the set of training data used in the next learning round.

2) Expert (and Non-expert) Labeling: We engage two domain experts to annotate the candidates generated by the *UBS* and *Distance UBS* sampling strategies. Each expert annotates one strategy; we also perform crosschecking for 10% of the first batch of labels, to get a measure of agreement between experts, with results reported in Section V-C. Experts use a web interface (see Figure 2) to approve or reject candidates, a task that is far more efficient than reading and annotating words in text. The interface provides example sentences as context for ambiguous candidates and allows the expert to access the publication(s) in which a particular candidate appears.

We aim to reduce the amount of costly expert time used to obtain labels. Therefore, for our baseline of randomly sampled NLP-filtered nouns, we experiment with a two-phase review process. Tokenization is one of the largest sources of error for scientific entities such as polymers, which contain characters

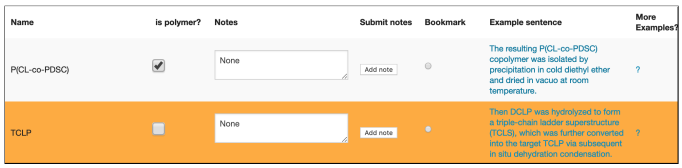

Fig. 2: Web interface for expert review of candidates. The expert indicates whether the name (column 1) is a polymer (tickbox in column 2), providing notes if desired (column 3). Clicking on "?" delivers up to 25 more example sentences.

such as ':', '(', '–', ',' etc. Tokenization can also generate incoherent tokens from text, equations, captions, etc. Such obvious non-candidates can be fairly easily detected by non-experts. For example, an untrained human annotator may be able to recognize that '$d\Sigma/d\Omega)(Q$' is not a polymer name, and thus save time for the experts. Hence, we assigned two graduate student labelers to curate the candidates generated by the random sampling strategy, which are less likely to contain target entities. We asked these untrained labelers to reject obvious non-candidates via the previously mentioned web interface. Our experts then reviewed the remaining candidates, indicating for each whether it is in fact a polymer referent.

V. Evaluation

We first report on a study in which we evaluate the generation of candidate entities using vector distances from representative (frequently used) entities. We then discuss the results of initial classification and subsequent four rounds of active learning using multiple word vector classifiers and our three sampling strategies: random, UBS, and Distance UBS. Finally, we experiment with word representations enhanced with character-level information using FastText [43, 44].

In this work, we evaluate extraction accuracy in terms of precision and recall. *Precision* refers to the fraction of predicted positives that are labeled correctly and *recall* to the fraction of actual positives that are labeled correctly.

A. Dataset

We work with a corpus of 1690 full-text publications in HTML format from *Macromolecules*, a relevant journal in polymer science. These documents comprise 381 947 sentences and 9 229 417 (253 195 unique) words or "tokens," of which 23 205 pass the NLP filter of Section IV-B.

From this corpus, we set aside a test set of 100 documents with 22 664 sentences and 508 391 (36 293 unique) tokens, of which 9656 pass the NLP filter. We engaged six experts to identify all one-word polymer names in this test set, a process that produced 467 unique one-word polymer names. We use these 467 names as a gold standard in subsequent subsections; we automatically label all NLP-filtered strings from the 100-document test set using these manually extracted names.

B. Seed Entities

Recall from Section IV-A that polyNER uses the Word2Vec word embedding tool to compute a word vector for each word. In order to maximize the number of actual entities in the dataset—and the ratio of target to non-target entities—in the initial set of labels, we explore how the choice of seed entities impact the number of target entities retrieved. While we cannot expect meaningful classification using only positive examples, given the imbalance in the whole dataset, we aim to select the Word2Vec parameters that yield the highest ratio of polymers in this initial batch of candidates.

In the experiments that follow, we use the 467 gold standard polymer names identified by experts in our 100-document test set to evaluate performance with different seed entities. Specifically, for each choice of seed entities that we want to evaluate, we determine the 10 000 NLP-filtered words with vectors closest to the seed entity vectors, and report what fraction of the 467 gold standard names are included in that 10 000. We use lower-case exact string matching between the gold standard polymer names and the proposed distance candidate strings to determine if a candidate is a polymer.

In previous work using the same corpus [45, 46], we built a dictionary of polymer names by using a rule-based approach and aggregating synonyms across ChemDataExtractor records. (A record consists of all information found about a chemical entity in a document.) Here we use this dictionary to identify the 10 most frequently occurring polymers in our corpus and their acronyms. We assume that frequent polymers provide a large number of sentences that illustrate context in which polymers are commonly used. Hence, we first test the most frequent, the three most frequent, and the ten most frequent polymers as seed entities. We also experiment with including and excluding their acronyms as additional seed entities. (Note that this modest set of 1, 3 and 10 seed entities could also be suggested by an expert.)

Rows 1–6 of Table I shows the results for these experiments. When using *polystyrene* (the most frequently used name) as a seed entity, the candidates contained 33.6% of the 467 gold standard polymers. We note a 2% increase in the fraction of polymers retrieved when using both *polystyrene* and *PS*, when compared to using *polystyrene* alone. The fraction of polymers increases by 10% when we use three representative entities (the three most frequent polymers in our datasets are *polystyrene*, *poly(methyl methacrylate)*, and *polyethylene*), but by less than 1% when using 10 instead of three entities. These results suggest that there is little value to using more than a few seed entities.

To further explore whether using larger numbers of seed entities may increase the fraction of polymers retrieved, we conducted a second set of experiments. We have built a small database of polymer properties (χDB) in previous work [45, 46]. Our corpus of 1690 publications included 111 out of 175 χDB polymers. We also scraped CrowDB, which lists some polymers and their properties at http://polymerdatabase.com/ for polymer names; 32 out of 295 scraped polymer names were found in our corpus. We measure how many of our gold standard polymers are identified when these 111 and 32 polymer names are used as seed entities, with results shown in rows seven and eight of Table I. These results confirm that using more entities does not increase the yield of polymers.

TABLE I: Fraction of gold standard polymer names in the 10 000 entities that are closest, by word vector distance, to various sets of seed entities.

#	Seed entities	Fraction of polymers extracted
1	Polystyrene	35.6%
2	Polystyrene, with acronym PS	37.7%
3	Three most frequent polymer names	46.9%
4	Three most frequent polymer names, with acronyms	48.0%
5	10 most frequent polymer names	46.5%
6	10 most frequent polymer names, with acronyms	48.4%
7	χDB polymer names	46.7%
8	crowDB polymer names	36.4%

Thus, in all subsequent experiments, we use the three most frequent polymers and their acronyms as seeds.

C. Labeling

We conduct some experiments to estimate labeling time. We ask two polymer scientists to label 200 candidates from a subset of our corpus. One expert reports 30 minutes, the other 45 minutes. We overestimate the time to label 200 candidates at one hour of expert time. Based on user feedback, we also improve the labeling Web interface after the above mentioned test rounds to further facilitate and speed up labeling. For example, we increase the number of example sentences available to provide context, to decrease the number of occurences in which experts have to look up original publications for candidates. We also increase the size of checkmarks to make it easier to reject erroneous candidates. In the initial labeling round, we perform crosschecking for 10% of the batch of labels. We confirm agreement between labels for all but one of 20: an agreement rate of 95%.

D. The Initial Classifier

Recall from Section IV-E that we use an initial set of 200 NLP-filtered entities that are close to seed entities to bootstrap our labeling process. Once those data are labeled by our experts, we use 80% to train a KNN classifier, validating on the remaining 20%. We then test this *initial classifier* against all 9656 NLP-filtered candidates in the 100-document test set, which as noted in Section V-A contain 467 polymers. Results are shown in Figure 3. Its Receiver Operating Characteristic (ROC) curve shows better-than-random behavior, with an area under curve (AUC) of 0.62. AUC can be misleading for imbalanced datasets, as it considers true negatives or correctly predicted non-polymers (inverse ratio). For example, if a dataset contains 90% of class A and 10% of class B, a classifier that predicts every sample as belonging to class A has an accuracy of 90%, but is not useful in practice. Therefore, we also plot the Precision Recall (PR) curve to show the tradeoff between precision and recall. While the AUC for the initial classifier is above random performance (0.5) its PR curve shows poor precision, regardless of recall. In previous work [12], we found that we could achieve better performance with more labels (897), suggesting that a KNN classifier begins to learn with more data. However, 160 labels (80% of 200) is not yet enough.

E. Comparing Sampling Strategies

After the initial round of labeling, we experiment with the three sampling strategies described in Section IV: Random, UBS, and Distance UBS, performing four rounds of active learning for each. Results, in Table II, show a significant amount of fluctuation and no notable improvement in the first two rounds for any strategy: precision remains under the initial precision of 6.5% for all. However, in the third round, we observe increases in precision, an improvement that is sustained in the fourth round for UBS. Figure 4 shows ROC and PR curves for the three strategies after four rounds. The AUC for UBS is 0.74 and that of Distance UBS is 0.70. The PR curves for both are improved (lifting away from the lower left corner of the graph) over the first round, with active learning performing better with UBS than with Distance UBS. When tested against our gold standard of 467 one-word polymer names, the KNN classifier achieves 18.2% precision and 45.6% recall. We notice that even the random strategy PR curve is improved (away from the initial PR curve and close to the Distance UBS curve), indicating that the NLP-filtering alone is enough to enable the learning process after 1000 labels.

We conclude that while the sampling step helps insure that the classes are balanced in the initial batch of labels, restricting ourselves to just distance candidates, as is done by Distance UBS, does not yield better results than using active learning with all NLP-filtered candidates (UBS). Intuitively, basic UBS can find *useful* instances (target and non-targets) to be labeled from the entire word embedding space, while examples from Distance UBS are clustered around the seed entities that may be collocated in that space.

TABLE II: Precision and recall relative to the gold standard for the initial classifier (round 0) and the classifiers trained also with the increased data obtained in each of four learning rounds, 1–4.

Round	Metric	Strategy		
		Random	UBS	Distance UBS
0	Precision	6.5%		
	Recall	19.1%		
1	Precision	3.8%	3.2%	5.3%
	Recall	0.29%	93.6%	56.8%
2	Precision	1.5%	3.8%	5.4%
	Recall	1.5%	46.4%	10.1%
3	Precision	6.0%	**21.2%**	3.9%
	Recall	44.6%	**40.0%**	84.3%
4	Precision	12.3%	**18.2%**	7.2%
	Recall	33.3%	**45.6%**	51.9%

We selected seed entities based on their frequency in our corpus. This observation suggests that we could also study how the choice of seed entities impact of the performance of the classifier during the active learning process. We revisit this concept of *diversity* of labels in the Section V-G. Note that with limited training data and based solely on context, the classifier retrieves 45.6% (more than one third) of the

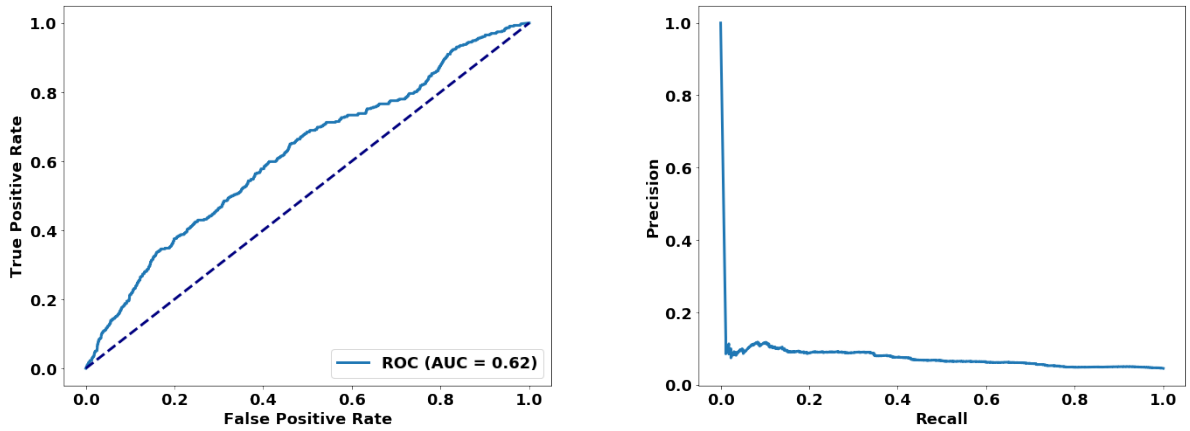

Fig. 3: ROC (left) and PR (right) curves for KNN model for the initial classifier. The PR curve shows that precision is low regardless of recall, indicating that we need more data.

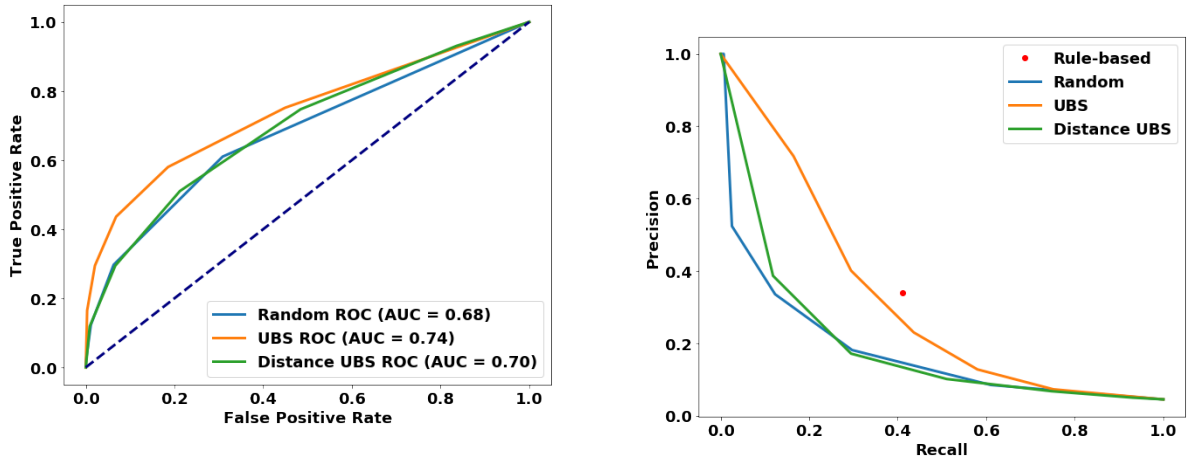

Fig. 4: ROC (left) and PR (right) curves after four rounds. The ROC curves for two active learning strategies, UBS and Distance UBS, show significant improvements, achieving AUCs of 0.74 and 0.70, respectively: significantly better than the 0.62 achieved by the initial classifier in Figure 3. Random achieves an AUC of 0.68. PR curves also show improvements relative to the initial classifier, with all three strategies lifting away from the bottom corner, indicating discriminative capacities. In both types of plots, UBS outperforms Distance UBS and approaches rule-based performance.

gold standard polymers with a precision of 18.2%, after five hours of expert labeling. For comparison, an attempt to extract polymer names using the rule: *if the name contains "poly" extract it as a polymer*, gets recall of 41% and precision of 34% on the same dataset. We conclude that with our relatively small and noisy dataset (entire documents of unstructured and uncurated text), we are able to achieve close to rule-based performance, using active learning and little labeling.

F. Active Learning Labels + Character-Level Embeddings

After just 1000 labels, the context-based classifier using active learning applied to NLP-filtered candidates achieved performance comparable to rule-based performance, but not quite as good as the polymer-enhanced CDE+. With the goal of further improving polyNER performance, we experiment with the use of an alternative word embedding model, FastText, which uses word representations enriched with sub-word (character-level) information. Because FastText considers sub-word information as well as context, it can 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(." Therefore, we generate a FastText word embedding model, and generate character-enhanced vectors for our UBS-labeled candidates.

Next, we train a KNN classifier using vectors for the candidates labeled through active learning from the NLP-filtered candidates (the active learning strategy identified as best-performing in Figure 4). We test the classifier against NLP-filtered nouns from our 100-document test set. KNN classifier performance improves when using these word vectors, achieving 29.7% precision and 81.9% recall, comparable to those achieved by CDE (see Section III-C). CDE's recall is

high at 74.5%, but its precision for polymers is, as expected, low at 8.7%, as it does not incorporate polymer knowledge. In Figure 5, we show the PR curve for the FastText vector classifier and also results for the polymer-enhanced CDE+, which achieve 42.2% precision and 68.3% recall on the same test set. We achieve higher recall than CDE and CDE+ using labels from UBS and FastText vectors and in-between (higher than CDE and lower than CDE+) precision.

G. Discussion and Future Work

We have previously used seed entities to bootstrap classifiers of context-based word vectors. Using an ensemble of classifiers, polyNER allowed users to tradeoff precision and recall. In this attempt to improve performance, while efficiently using experts' time, we used active learning to obtain more labels. However, we do not observe an increase of precision over the 5% fraction of polymers until the third round of active learning with NLP-filtered words (800 labels). This suggests that our label batch size is lower than the minimum number of labels necessary to start the learning process. More detailed study of performance vs. label batch size will help pinpoint the appropriate number of labels and level of bootstrapping required for learning. We can also ask the following questions: Could we achieve CDE performance classifying only context-based (not character enhanced) representation? If so, how many more iterations would it take? If not, how much do we need to increase the size and/or improve our current corpus of 1690 documents? Corpora for NER tasks are curated and typically include sentences with one or more entities. We could investigate other semi-supervised method to eliminate negative sentences from our corpus (e.g. labeling sentences and classification of sentence vectors for example).

PolyNER achieves CDE performance using labels obtained via active learning (UBS sampling strategy) and FastText vectors. We attribute the increased performance to the character embedding enhancement, which not only recognizes "poly" (and yields more names based on this n-gram comparison), but also filters out more anomalous candidates (preceding or following polymer names) generated during tokenization and missed by the filtering steps, such as "A_mB_n" and "*Mw/Mn=1.36*." In other words, the classifiers of character and (context-based) word embedding vectors perform better than classifiers of only context-based word embedding. Given this result, one may wonder whether the active learning process itself could benefit from using this enhanced vector embedding. To determine whether this is the case, we repeated the active learning experiment using the entire corpus of NLP-filtered candidates and classifying FastText (enhanced) vectors instead of Gensim vectors at each round. However, the performance was worse than random.

These results suggest that character-level information enhances classifier performance only once a certain threshold of context information has been captured by the embedding. We explain this observation as follows. In FastText, the portion of the word embedding vector generated by using context varies depending on how much context is available in the entire corpus. For words deemed to have enough context, vectors do not include any character-level information. At the other extreme, for previously unseen words, the embedding is generated based solely on character n-gram information and comparison to other words in the corpus. During the active learning process, candidates to be labeled by experts are selected by using maximum-entropy-based uncertainty sampling: that is, words for which prediction probability is similar for target and non-target. Such candidates are more likely to lack context and thus have vectors that use mostly character-level information. As a result, the expert is often presented with nearly identical candidates (e.g., PS13k/PMMA12k, PS214k/PMMA12k, PS31.6k/PMMA12k), which hinders the learning process as these candidates are located in close vicinity in terms of the full (character and context) word embedding space. In other words, in this full space, while their uncertainty measure is comparable, these examples are not *diverse*, where diversity is a measure of the distance of the examples to each other or previously labeled instances [47]. One solution to explore in future work would be to impose a diversity constraint on the candidates, for example by using batch active learning [48]. This will be part of broader plan to study the effect of quantity (more documents) and quality (select paragraphs or sentences) of data on the word embedding model and the final results.

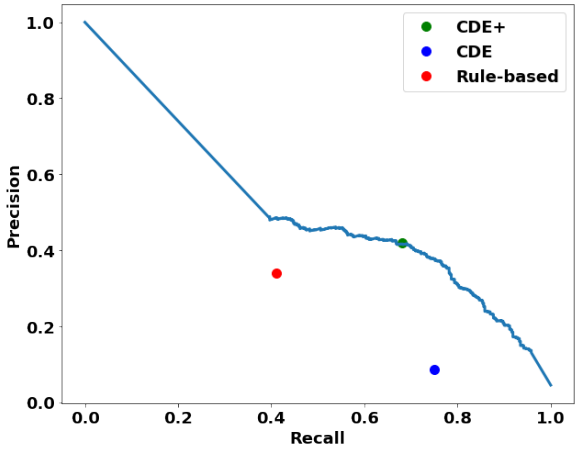

Fig. 5: PR curve for KNN model trained with active learning labels and word representations enriched with character-level information. Results for CDE+ are also shown. Note: PR curves are obtained by varying the threshold of probability that separates classes; straight lines occur when several points have similar probabilities and changing the threshold yields identical precision to recall ratio.

VI. Conclusion

A lack of expert-annotated training data impedes the adoption of machine learning techniques in certain scientific applications. PolyNER overcomes this challenge by using active learning to target expert input so that accurate scientific named entity recognition can be performed at low cost. We show that by using NLP techniques, we can bootstrap a word vector classifier of scientific entities. Using polyNER's labels and a classifier of character-enhanced word embedding vectors,

we achieve performance comparable to a best-of-breed hybrid NER model (CDE+) that required much expert development. In contrast, polyNER was trained on data annotated using just five hours of expert time and a little untrained crowd input. Our work highlights the potential for using minimal labeled data and focused expert input to enable machine learning techniques for previously unmined scientific entities. We are currently exploring using polyNER-labeled data to annotate text for other NER approaches, such as bidirectional long short-term memory models. We also plan to explore the generalizability of our approach further by applying it extracting previously unmined scientific entities in other fields.

ACKNOWLEDGMENTS

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 [49, 50]. Official contribution of the National Institute of Standards and Technology; not subject to copyright in the United States.

REFERENCES

[1] Y. Song *et al.*, "POSBIOTM-NER in the shared task of BioNLP/NLPBA 2004," in *International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, 2004, pp. 100–103.

[2] M. Krallinger *et al.*, "CHEMDNER: The drugs and chemical names extraction challenge," *Journal of Cheminformatics*, vol. 7, no. 1, p. S1, 2015.

[3] J. J. de Pablo *et al.*, "The Materials Genome Initiative, the interplay of experiment, theory and computation," *Current Opinion in Solid State and Materials Science*, vol. 18, no. 2, pp. 99–117, 2014.

[4] R. Leaman and G. Gonzalez, "BANNER: An executable survey of advances in biomedical named entity recognition," in *Biocomputing*, 2008, pp. 652–663.

[5] Z. Zeng *et al.*, "Survey of natural language processing techniques in bioinformatics," *Computational and Mathematical Methods in Medicine*, 2015.

[6] L. Hawizy *et al.*, "ChemicalTagger: A tool for semantic text-mining in chemistry," *Journal of Cheminformatics*, vol. 3, no. 1, p. 17, 2011.

[7] T. Rocktäschel *et al.*, "ChemSpot: A hybrid system for chemical named entity recognition," *Bioinformatics*, vol. 28, no. 12, pp. 1633–1640, 2012.

[8] R. Leaman *et al.*, "tmChem: A high performance approach for chemical named entity recognition and normalization," *Journal of Cheminformatics*, vol. 7, no. 1, p. S3, 2015.

[9] M. C. Swain and J. M. Cole, "ChemDataExtractor: A toolkit for automated extraction of chemical information from the scientific literature," *Journal of Chemical Information and Modeling*, vol. 56, no. 10, pp. 1894–1904, 2016.

[10] S. R. Young *et al.*, "Data mining for better material synthesis: The case of pulsed laser deposition of complex oxides," *Journal of Applied Physics*, vol. 123, no. 11, p. 115303, 2018.

[11] R. Tchoua *et al.*, "Towards hybrid human-machine scientific information extraction," in *2018 New York Scientific Data Summit (NYSDS)*. IEEE, 2018, pp. 1–3.

[12] R. B. Tchoua *et al.*, "Creating training data for scientific named entity recognition with minimal human effort," in *International Conference on Computational Science*, 2019.

[13] M. Buhrmester *et al.*, "Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data?" *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, 2011.

[14] R. B. Tchoua *et al.*, "Towards a hybrid human-computer scientific information extraction pipeline," in *13th International Conference on e-Science*, 2017, pp. 109–118.

[15] J.-D. Kim *et al.*, "Introduction to the bio-entity recognition task at JNLPBA," in *International Joint Workshop on Natural Language Processing in Biomedicine and its Applications*, 2004, pp. 70–75.

[16] R. C. Hiorns *et al.*, "A brief guide to polymer nomenclature," *Polymer*, vol. 54, no. 1, pp. 3–4, 2013.

[17] J. Tamames and A. Valencia, "The success (or not) of HUGO nomenclature," *Genome Biology*, vol. 7, no. 5, p. 402, 2006.

[18] D. J. Audus and J. J. de Pablo, "Polymer informatics: Opportunities and challenges," *ACS Macro Letters*, vol. 6, no. 10, pp. 1078–1082, 2017.

[19] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *7th Conference on Natural Language Learning*, 2003, pp. 142–147.

[20] D. M. Jessop *et al.*, "OSCAR4: A flexible architecture for chemical text-mining," *Journal of Cheminformatics*, vol. 3, no. 1, p. 41, 2011.

[21] M. Krallinger *et al.*, "Overview of the chemical compound and drug name recognition (CHEMDNER) task," in *BioCreative Challenge Evaluation Workshop*, vol. 2, 2013, p. 2.

[22] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National Science Review*, vol. 5, no. 1, pp. 44–53, 2017.

[23] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep. 1530, 2005.

[24] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *5th ACM conference on Digital libraries*, 2000, pp. 85–94.

[25] S. Brin, "Extracting patterns and relations from the world wide web," in *International Workshop on The World Wide Web and Databases*, 1998, pp. 172–183.

[26] S. E. Peters *et al.*, "A machine reading system for assembling synthetic paleontological databases," *PLoS One*, vol. 9, no. 12, p. e113523, 2014.

[27] C. De Sa *et al.*, "DeepDive: Declarative knowledge base construction," *ACM SIGMOD Record*, vol. 45, no. 1, pp. 60–67, 2016.

[28] S. Auer *et al.*, "Dbpedia: A nucleus for a web of open data," in *The Semantic Beb*, 2007, pp. 722–735.

[29] K. Bollacker *et al.*, "Freebase: A collaboratively created graph database for structuring human knowledge," in *SIGMOD International Conference on Management of Data*, 2008, pp. 1247–1250.

[30] "Paleodb," http://paleodb.org, accessed March, 2019.

[31] A. J. Ratner *et al.*, "Data programming: Creating large training sets, quickly," in *Advances in Neural Information Processing Systems*, 2016, pp. 3567–3575.

[32] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *11th International Conference on Machine Learning*, 1994, pp. 148–156.

[33] C. Campbell *et al.*, "Query learning with large margin classifiers," in *17th International Conference on Machine Learning*, 2000, pp. 111–118.

[34] H. S. Seung *et al.*, "Query by committee," in *5th Annual Workshop on Computational Learning Theory*, 1992, pp. 287–294.

[35] H. T. Nguyen and A. Smeulders, "Active learning using pre-clustering," in *21st International Conference on Machine Learning*, 2004, p. 79.

[36] S. Dasgupta and D. Hsu, "Hierarchical sampling for active learning," in *25th International Conference on Machine Learning*, 2008, pp. 208–215.

[37] S. Basu *et al.*, "Active semi-supervision for pairwise constrained clustering," in *International Conference on Data Mining*, 2004, pp. 333–344.

[38] T. Mikolov *et al.*, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[39] ——, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems*, 2013, pp. 3111–3119.

[40] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in *Workshop on New Challenges for NLP Frameworks*, 2010.

[41] J. D. Choi *et al.*, "It depends: Dependency parser comparison using a web-based evaluation tool," in *53rd Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2015, pp. 387–396.

[42] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[43] P. Bojanowski *et al.*, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[44] A. Joulin *et al.*, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[45] R. B. Tchoua *et al.*, "A hybrid human-computer approach to the extraction of scientific facts from the literature," *Procedia Computer Science*, vol. 80, pp. 386–397, 2016.

[46] ——, "Blending education and polymer science: Semiautomated creation of a thermodynamic property database," *Journal of Chemical Education*, vol. 93, no. 9, pp. 1561–1568, 2016.
[47] K. Brinker, "Incorporating diversity in active learning with support vector machines," in *20th International Conference on Machine Learning*, 2003, pp. 59–66.
[48] B. Settles, "Active learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2009.
[49] J. Towns *et al.*, "XSEDE: Accelerating scientific discovery," *Computing in Science & Engineering*, vol. 16, no. 5, pp. 62–74, 2014.
[50] C. A. Stewart *et al.*, "Jetstream: A self-provisoned, scalable science and engineering cloud environment," 2015.