JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

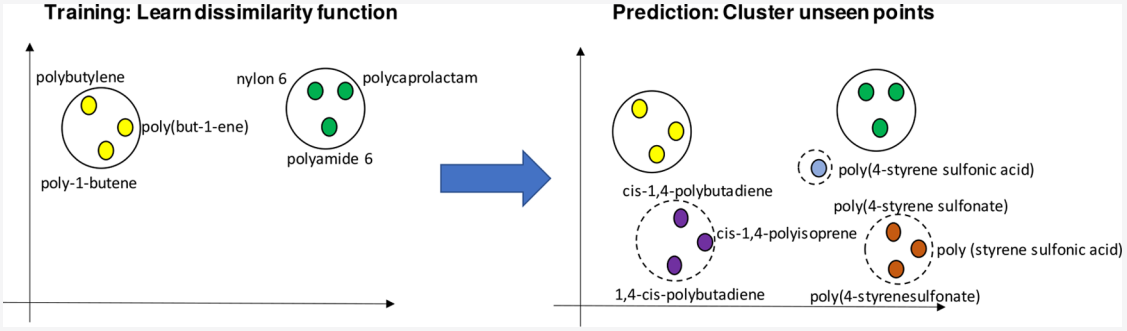
pubs.acs.org/jcim Article

Machine-Guided Polymer Knowledge Extraction Using Natural Language Processing: The Example of Named Entity Normalization

Pranav Shetty and Rampi Ramprasad*

Cite This: J. Chem. Inf. Model. 2021, 61, 5377−5385 Read Online

ACCESS | Metrics & More | Article Recommendations | Supporting Information



ABSTRACT: A rich body of literature has emerged in recent years that discusses the extraction of structured information from materials science text through named entity recognition models. Relatively little work has been done to address the "normalization" of extracted entities, that is, recognizing that two or more seemingly different entities actually refer to the same entity in reality. In this work, we address the normalization of polymer named entities, polymers being a class of materials that often have a variety of common names for the same material in addition to the IUPAC name. We have trained supervised clustering models using Word2Vec and fastText word embeddings reported in previous work so that named entities referring to the same polymer are categorized within the same cluster in the word embedding space. We report the use of parameterized cosine distance functions to cluster and normalize textually derived entities, achieving an F1 score of 0.85. Furthermore, a labeled data set of polymer names was utilized to train our model and to infer the true total number of unique polymers that are actively reported in the literature. For ∼15,500 polymer named entities extracted from our corpus of 0.5 million papers, we detected 6734 unique clusters (i.e., unique polymers), 632 of which were manually curated to train the normalization model. This work will serve as a critical ingredient in a natural language processing-based pipeline for the automatic and efficient extraction of knowledge from the polymer literature.

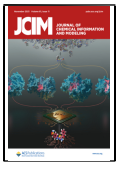
INTRODUCTION

Machine learning and data-driven methods have made significant in-roads into the domain of polymer science. The field of polymer informatics seeks to develop rapid data-driven predictive models of polymer properties and to solve the inverse problem aimed at the design of new polymers with desirable properties.[1−3] This field, just like the parent discipline of materials informatics, suffers from data scarcity.[4] A possible solution to the data-scarcity problem is to use natural language processing (NLP) methods to extract information at scale from the polymer literature. NLP methods have been used in the inorganic materials space for synthesis planning and the extraction of materials insights.[5−7] Additionally, NLP has most recently been used in the polymer space for predicting novel polymers for existing applications.[8]

An important component of NLP is named entity recognition (NER), that is, identifying spans of text belonging to certain predefined categories. The traditional method of training supervised NER models requires manual annotation of large corpora of text. This process can be made much faster by using weak supervision in the form of labeling functions that automatically label input text using heuristic rules defined by domain experts.[9−11] In this work, we release a data set of polymer named entities (PNEs) that can be used as a weak labeling source for building NER models involving polymeric materials. Many of the PNEs in this data set refer to the same chemical entity. We have addressed this problem using named entity normalization (NEN), another critical component in NLP workflows.

Normalization here refers to clustering together all polymer names referring to the same chemical entity and assigning a unique identity to it, a problem that has attracted some recent interest.[12] NEN, also called entity linking, is a well-studied

Received: May 17, 2021
Published: November 9, 2021



https://doi.org/10.1021/acs.jcim.1c00554
J. Chem. Inf. Model. 2021, 61, 5377−5385

problem in NLP. Graph-based approaches that use existing knowledge bases to infer the co-relation between entities and textual approaches that extract features such as word embeddings from large amounts of text that are then used for NEN are the two most common approaches used for this problem.[13,14] Past work in Materials Science has also employed dictionary-based approaches using chemical databases like PubChem and JoChem.[15,16]

Despite having a standard scheme of nomenclature defined by International Union of Pure and Applied Chemistry (IUPAC) rules, that is, the IUPAC source-based name and the IUPAC structure-based names,[17] the vast majority of polymer names found in the literature do not conform to these norms (see Section S1 in the Supporting Information). In order to solve this issue, we normalize polymer names. Normalizing PNEs has several benefits. It enables us to get a clearer overview of the number of polymers that are actively reported upon in the literature. Such a list can also be used to build a search functionality over a knowledge base of polymers that can take arbitrary variations in names as input. Robustly and reliably normalizing PNEs is also the first step in a pipeline that can convert arbitrary polymer names to a structure.

Variations in common names used to refer to polymers are also frequently observed. The "poly(tetrahydrofuran)" polymer is also referred to in the literature as "polyTHF," "poly(tetramethylene glycol)," "polytetramethylene oxide," "poly(tetramethylene oxide)," "PTMO," and "PTMG." Variations in spelling for the same polymer are also commonplace, for example, "poly(vinylidene fluoride)" and "poly(vinilydene fluoride)" and "polytetrafluoroethylene" and "polytetrafloroethylene."[18,19] More examples are shown in Table 1. Many of these are nontrivial variations and cannot be captured using simple heuristic rules. Normalizing polymer entities is thus a challenge and requires machine learning methods in order to perform this task successfully.

Table 1. Examples of Normalized PNEs[a]

polymer name or key	selected name variants of the same polymer
poly(vinylcyclohexane)	poly(cyclohexylethylene), poly(cyclohexyl ethylene), polycyclohexylethylene
poly(p-methylstyrene)	poly(4-methyl styrene), poly(4MS), poly(*p*-methyl styrene), polyparamethylstyrene
polyvinylpyrrolidone	poly(*N*-vinylpyrrolidone), poly(vinyl)pyrrolidone, poly-vinyl pyrrolidone, poly-*N*-vinyl pyrrolidone, polyvidone, poly(vinyl pyrrolidone)
poly(vinyl chloride)	poly-vinyl chloride, poly(vinyl chloride), polyvinyl chloride, poly vinyl chloride

[a]The key used here is the most frequently occurring name for that polymer.

The idea of word vectors from NLP combined with supervised clustering provides a solution to this problem. Word vectors (also referred to as word embeddings) are one of the pillars of modern NLP. One can think of a word vector as representing the meaning of a token in a vector space where a token is a piece of text that is used downstream for processing. Tokens that are similar will have word vectors that are close in the word vector space, that is, have a high dot product. Thus, in a polymer science context, PNEs used for similar applications will have a high dot product, and in particular, PNEs referring to the same polymer will be close to each other in the word vector space. Furthermore, this vector space has an intrinsic structure that allows downstream models such as NER to be trained. Word2Vec and fastText are some of the earliest methods for training word embeddings.[20,21] They rely on contextual information inferred by observing a word in several different contexts over a large corpus of text to train vector representations of words. In order to normalize PNEs to partition our list of polymers, we have trained a model that can learn this pattern of proximity in the word vector space for polymers through the idea of supervised clustering.

Supervised clustering uses a labeled data set of known clusters to learn a dissimilarity function that partitions unseen points into their respective clusters.[22] The first step in supervised clustering is to learn a dissimilarity function using labeled data that is trained to minimize a loss or penalty function over known clusters. The dissimilarity function transforms the original vector space into a space where the known clusters occur close together, and hence, clustering in this space is easier. The next step is to perform clustering to partition the data points in the transformed space.

This is the first effort to systematically and comprehensively address the normalization of chemical named entities (CNEs) in the polymer space. Also reported is the use of parameterized cosine distance functions for supervised clustering, which significantly improves normalization performance. We release our labeled data set of polymer names and clusters with their corresponding word embeddings, our training code, and the list of clusters predicted by the model for unlabeled data. In addition to serving as a weak supervision source for NER, we hope that this data set will serve as a useful benchmark for NEN.

METHODS

Data Set. We collected a corpus of ∼0.5 million polymer papers from various materials science publishers, including Elsevier, Wiley, Royal Society of Chemistry, and Springer Nature. Further details on our corpus of papers can be found elsewhere.[8] We used the ChemDataExtractor[23] tool to extract all CNEs, that is, mentions of all chemical names. Out of these names, we picked a subset that contained the term "poly." This yielded a data set of ∼25,500 PNEs. Due to the diversity in polymer names, this is necessarily a rough heuristic and misses out certain biologically derived polymers such as "dextran" or "cellulose" and includes certain nonpolymer chemical entities such as "polysulfide." This heuristic was, however, observed to work well in practice. Out of 100 randomly sampled PNEs, we found that only one, that is, "polyfunctionalized" was not actually a polymer. From this set, we removed all polymers corresponding to copolymers and blends by using keywords such as "copoly," "-ran-," "-ipn-," and so forth. This processing step was done as the determination of two PNEs referring to the same homopolymer is well defined as opposed to making that determination for different blends or copolymers, which vary on a continuous scale. We also removed all PNEs corresponding to polymer classes such as polyacetals, polyoxides, polyimides, polyolefins, and so forth using a list of polymer classes that we manually constructed and other keywords that are erroneously classified as PNEs such as "polycrystalline," "polyelectrolyte," and so forth. The aforementioned data processing steps left us with 15,500 PNEs. Many of these PNEs refer to the same real-world entity, and hence, the goal of this work is to develop a method to normalize this data set. We manually annotated 2380 PNEs into distinct clusters, which corresponded to 632 unique

https://doi.org/10.1021/acs.jcim.1c00554
J. Chem. Inf. Model. 2021, 61, 5377−5385

polymers. We ensured that the labeled data points were uniformly distributed over the word vector space of all PNEs. We did so by picking points to label from the *t*-SNE plot[8,24] of all unlabeled points, as illustrated in Figure S1. This constituted our training set for supervised clustering of the remainder of the ∼13,000 PNEs. Existing chemical databases containing polymer information such as PoLyInfo,[25] ChemIDPlus,[26] or PubChem[27] contain manually curated variations for many common polymer names but have low coverage over all variations occurring in the literature and hence were not used in this work. The PubChem database, for instance, contains only 251 overlapping entities with our labeled data set corresponding to 131 unique polymers.

Word Vector Models. We have trained two different word vector models on our corpus of papers, namely Word2Vec (Skip-Gram variant of Word2Vec)[20] and fastText.[21] In particular, we generated a 128-dimensional vector representation for all the PNEs in our data set to be normalized (as described in the Data Set Section). The word vectors so generated were used as feature vectors for the downstream machine learning models we trained.

Word2Vec trains a single vector representation for tokens, while fastText sums up the vector representation associated with the subwords in a token. For example, "polyethylene" would have a Word2Vec vector representation trained using only the contextual information in all contexts that it appears in. FastText on the other hand represents a word as a bag of character *n*-grams, including the word itself, where the size of the *n*-gram varies from 3 to 6. The vector representation is obtained by summing up the vectors for all the *n*-grams. For morphologically rich languages like Czech and German, fastText has been known to perform better than Word2Vec baselines on tasks like word analogies.[21] FastText embeddings have also been used in materials science such as in ref 28 where they were used as inputs to a model that predicted precursors to perovskites. Similarly, a normalization task on polymer names, which have rich internal structure reflecting chemical information, is also likely to benefit from using subword information. Details on tokenization and the parameters used during word vector training can be found in ref 8.

Baseline. As a baseline against which to compare our models, we replicated the approach followed in ref 29. This is one of the few papers to attempt normalization over named entities in materials science. This paper performed normalization over seven different classes of materials science entities such as "synthesis methods," "characterization methods," "properties," and so forth. For example, "CVD" and "chemical vapor deposition" are normalized to the same cluster. We adapted the same approach to our problem statement and data set. For every pair of polymers, the corresponding word embeddings were concatenated, which gave us the feature vector X for that pair of polymers. If the pair of polymers belonged to the same cluster, then the pair is assigned a label of one, and if they belonged to different clusters, then a label of zero was assigned. We then trained a binary random forests classifier using the aforementioned feature vector X and label y. In a separate trial, we also augmented the feature vector X with three handcrafted features as described in ref 29. The three features used were (1) the Levenshtein distance[30] between the pair of polymer tokens, (2) a label of one if the pair of polymers has the same stem word and zero otherwise, and (3) the cosine similarity between the two polymer word vectors. Each of the three features used can be defined only over pairs of polymers. This method is referred to as pairwise classification in subsequent discussions.

Observe that the label classes are highly imbalanced with label zero being far more numerous than label one. In order to have a balanced training set and closely follow the procedure outlined in ref 29, we sampled half the points of label one and an equal number of points of label zero. This yielded 16668 points with balanced classes. The remaining points were all part of the test set. The test set remained unbalanced, but this, we must require of our test set as real-world data on which any such model must be tested will be similarly unbalanced.

The SynSetMine framework[31] is used as an additional baseline. This method splits the labeled clusters into synonym sets and instances. Instances could be positive instances, that is, instances that belong to that synonym set but are not included in it, and negative instances that do not belong to that set. A neural network classifier learns whether an instance is positive or negative using the vector representation of the instance and a permutation-invariant vector representation of the set. We used 90% of the labeled clusters as the training set and 10% as the test set in order to be consistent with our later experiments.

Supervised Clustering. In supervised clustering,[32,33] we start with a list of data points, each represented by a D-dimensional feature vector. It is further assumed that all points can be partitioned into clusters with the number of clusters typically not known a priori. Given the labels for a subset of the data with each point assigned to its true cluster, we aim to train a model that can partition the remaining points into their true clusters which are different from the clusters of the training data. This is done by fitting a parameterized distance function with the parameters being tuned based on the training data set. The parameterized distance function acts as a dissimilarity function once a distance threshold is applied to discern points as belonging to the same or different cluster. The parameterized distance function achieves a mapping of the original feature vectors to a different vector space where the data points are "easier" to separate into clusters. We tested two different parameterized distance functions, that is, the Mahalanobis distance[34] and a parameterized version of the cosine distance. If x_i and x_j are two points, then the Mahalanobis distance function is a generalized version of the Euclidean distance and is given by

$$f(x_i, x_j) = \sqrt{(x_i - x_j)^{\mathrm{T}} S^{-1} (x_i - x_j)} \tag{1}$$

where S is the covariance matrix that is learned. The parameterized cosine distance function[35] we used has the form

$$f(x_i, x_j) = 1 - \frac{(Wx_i)^{\mathrm{T}} Wx_j}{\|Wx_i\|_2 \|Wx_j\|_2} \tag{2}$$

where W is the symmetric matrix that maps the points x_i and x_j from the original space to a space where clustering is easily performed and is learned during the training procedure and $\|\cdot\|_2$ is the L_2 norm. We used the EXPα training procedure described in ref 22 to learn this dissimilarity function. This training procedure was adopted because as reported in this paper, other commonly used loss functions such as all-pairs loss and minimum-spanning trees favor some linkage functions over others and can lead to poor clustering quality.

Once the dissimilarity function was learned, we used hierarchical agglomerative clustering (HAC) to perform clustering.

https://doi.org/10.1021/acs.jcim.1c00554
J. Chem. Inf. Model. 2021, 61, 5377−5385

Hierarchical Agglomerative Clustering. HAC[36,37] is a clustering technique in which, starting with every node in its own cluster, the closest clusters are successively merged till only a single node is left. This yields a tree structure of nodes, and the final flat clusters are determined by using a threshold, which itself is learned from the data. We split our labeled data set into 75% training, 15% validation, and 10% test with all points in the same cluster being partitioned into the same set. Word2Vec or fastText embeddings were used, and no handcrafted features were used in this case as the latter are only defined over pairs of polymers. The parameters of the distance function are learned during the training phase, and the threshold for partitioning the points into clusters is found by maximizing the $F1$ score (explained in the Evaluation Metrics section) obtained over the validation set. The threshold here is a distance such that all siblings in a sub-tree at a node having a distance lower than the threshold are partitioned into the same cluster, while all sibling nodes having a distance greater than the threshold are in different clusters (illustrated in Figure 1).

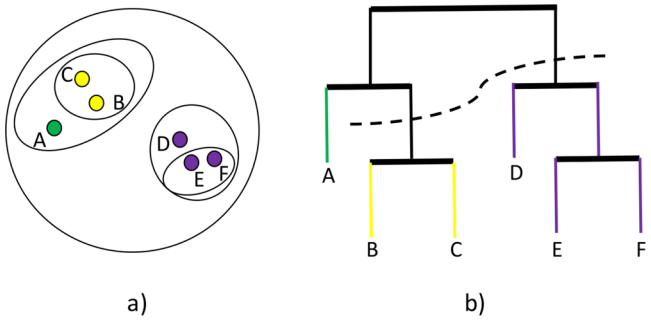


Figure 1. Illustration of HAC. (a) A−E are points to be clustered. The color indicates its true cluster. HAC works by combining closest points successively into larger clusters. (b) Resulting tree diagram. A decision threshold must then be learned to pick out the true clusters (shown by a dashed line). The distance between points is parameterized and learned during the training procedure.

The advantage of hierarchical clustering as opposed to most other clustering algorithms is that the number of clusters does not have to be specified a priori. This is useful in our case where the total number of unique polymers is not known to begin with.

Measuring the distance between clusters during the merging process is an important part of HAC. Let C_u and C_v be the two clusters under consideration. The distance between clusters can be measured in many ways as described below.

1. Single linkage: shortest distance between any pair of points in C_u and C_v
2. Complete linkage: longest distance between any pair of points in C_u and C_v
3. Average linkage: average distance between every pair of points in C_u and C_v
4. ExpLinkage: this is a linkage method first introduced in ref 22 and interpolates smoothly between the above mentioned three linkage methods, which has the below functional form

$$D(C_u, C_v) = \frac{\sum_{x_i \in C_u} \sum_{x_j \in C_v} e^{\alpha f(x_i, x_j)} f(x_i, x_j)}{\sum_{x_i \in C_u} \sum_{x_j \in C_v} e^{\alpha f(x_i, x_j)}} \quad (3)$$

where x_i and x_j are arbitrary points, $f(x_i,x_j)$ is the distance between them, and α is a learnable parameter. As $\alpha \to -\infty$ and $\alpha \to \infty$, $D(C_u,C_v)$ approaches single linkage and complete linkage, respectively, and reduces to average linkage at $\alpha = 0$.

Evaluation Metrics. We evaluated the performance of our models based on the pairwise F1 score[22] computed over the true and predicted clusters for every pair of points. Let W^* be the pairs of points in the evaluation data belonging to the same ground truth cluster and $\hat{W}$ be all pairs of points belonging to the same predicted clusters. From the definitions, it is clear that $W^* \cap$ and $\hat{W}$ correspond to the true positives and $\hat{W} - W^*$ corresponds to false positives, while $W^* - \hat{W}$ corresponds to false negatives. These three quantities can be used to compute the precision, recall, and their harmonic mean to yield the F1 score as shown in the below equations.

$$\text{Precision} = \frac{|W^* \cap \hat{W}|}{|W^* \cap \hat{W}| + |\hat{W} - W^*|}$$

$$\text{Recall} = \frac{|W^* \cap \hat{W}|}{|W^* \cap \hat{W}| + |W^* - \hat{W}|}$$

$$F1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \quad (4)$$

RESULTS AND DISCUSSION

Distribution of Polymer Names. Figure 2a shows the most frequently occurring polymers in our corpus of papers. Figure 2b shows the frequency and the rank of polymers plotted on a log−log scale. Rank 1 polymer here is the most frequently occurring polymer. Observe that the best fit line is nearly a perfect straight line fit, which strongly suggests that

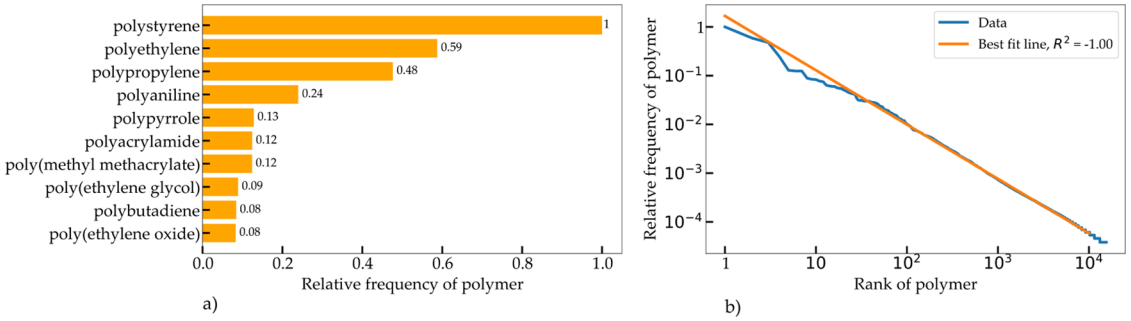


Figure 2. Polymer name statistics: (a) Ten most commonly occurring polymers in our corpus of papers ordered by the relative frequency with which they appear in the literature, relative with respect to the most frequently occurring polymer, i.e., "polystyrene" and (b) plot of frequency of polymers against their rank when ordered by frequency on a log−log scale.

https://doi.org/10.1021/acs.jcim.1c00554
J. Chem. Inf. Model. 2021, 61, 5377−5385

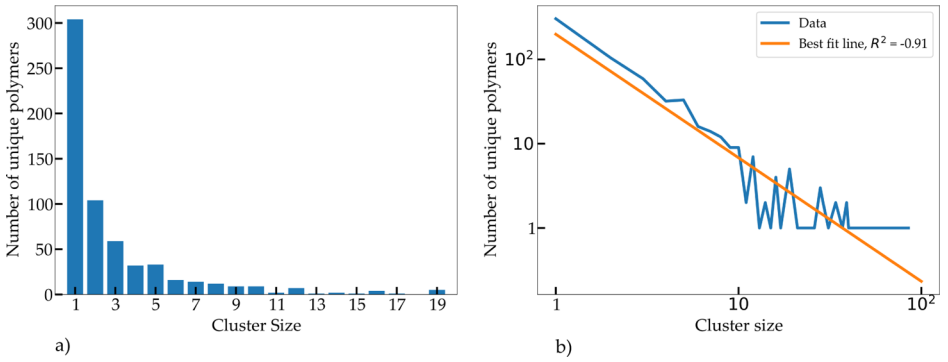


Figure 3. Polymer cluster statistics: (a) plot of the distribution of cluster sizes for the polymers in our data set of 2380 polymer names and 632 clusters and (b) same data as (a) but plotted on a log scale. The data are shown using a continuous line to better illustrate power law behavior.

polymer name frequencies follow a power-law distribution. Thus, the most frequent polymers occur far more frequently than the less frequent ones. This resembles other power-law distributions observed in a written language such as Zipf's law,[38] which is an empirical law that points out that the rank-frequency distribution is an inverse power law for data in many different domains. The power law exponent in Zipf's law applied to the rank-frequency distribution of words in any language is close to one, which is also what we observe over polymer names, as shown in Figure 2.

We can extend the idea that polymer names follow a power-law distribution by taking a closer look at the PNEs. The PNEs shown in Figure 2b are not normalized, that is, multiple PNEs could refer to the same real-world entity. Figure 3a shows the cluster size distribution of polymer names where the cluster size refers to the number of PNEs referring to a unique polymer.

Fitting the cluster size to the number of clusters with a given cluster size on a log scale, as shown in Figure 3b, gave us a straight line with $R^2 = -0.91$ with a power law coefficient of 1.47. This suggests that cluster sizes also follow a power-law distribution. The noise observed at the tail shown in Figure 3b is because large clusters are sparse. If we assume that the cluster sizes are indeed distributed as a power law, we can estimate the number of unique polymers in our data set. To see this, let $\{x_i\}$ be the set of cluster sizes and let $g(.)$ be a function that maps a cluster size to the number of distinct clusters of that size. It is clear that the number of unique polymers is given by $\sum g(x_i)$, and the total number of unnormalized PNEs is given by $\sum x_i g(x_i)$. We know these quantities for our training data set. For the complete data set, we know the total number of polymers. The problem reduces to computing the number of unique polymers given the aforementioned equations. We can show that the ratio of the total polymers to the number of unique polymers, the average cluster size is constant (shown in Section S2 of the Supporting Information), that is, does not depend on the total number of PNEs under consideration. The number of unique polymers in our dataset can thus be estimated to be ∼4100.

Model Performance. The performance of the baseline models for normalizing named entities is shown in Table 2. All the models are observed to have low recall. Reference 29 reported a precision and recall of 0.95 and 0.94, respectively, for a random forest model trained using concatenated Word2Vec vectors plus handcrafted features on 10000 labeled entity pairs. However, as reported in the paper, the test data set is synthetically generated such that the train and test set have balanced classes. Our test set on the other hand is highly imbalanced, with the ratio of the zero-label points to one-label points being nearly 300:1, more closely reflective of real data. The low recall is indicative of a large number of false negatives, that is, polymers that belong to the same cluster but are labeled otherwise. This along with the high precision indicates that points far from the decision boundary are being classified correctly, and for points close to the decision boundary, the model prediction errs on the side of false negatives.

Table 2. Performance Over the Test Set of the Pairwise Classification Model and SynSetMine[a]

model	precision	recall	*F*1
concatenated word vectors	0.92	0.09	0.16
concatenated word vectors + handcrafted features	0.95	0.11	0.19
SynSetMine	0.95	0.30	0.46

[a]All models used fastText features, with word vectors concatenated for pairs of polymers in the case of the pairwise classification approach.

We performed an ablation study on the presence and absence of the handcrafted features described in the "Baseline" subsection. The precision and recall both show a jump in the presence of handcrafted features. Considering that only three additional features are concatenated with a 256-dimensional vector, the handcrafted features indeed capture a lot of information about the underlying data set that helps normalization performance. The performance using Word2Vec features is reported in Table S1.

Modeling normalization as a classification problem poses a significant challenge due to classes being unbalanced in real data. On reformulating as a supervised clustering problem, binary class labels are no longer a part of the problem formulation, and hence, the unbalanced nature of the classes is consequently addressed. Modeling entity normalization as a supervised clustering task is thus observed to give much better performance, as discussed next.

The performance of our supervised clustering model is shown in Figure 4. We report precision, recall, and *F*1 score of the clustering performance on the test set. The error is the standard deviation computed over five iterations, each with a different split of train, test, and validation set.

As shown in Figure 4, fastText clearly outperforms Word2Vec. This indicates that sub-word similarity matching is a good proxy for matching similar polymers. From the same figure, it is clear that the parameterized cosine distance metric outperforms the Mahalanobis distance. The latter is an

https://doi.org/10.1021/acs.jcim.1c00554
J. Chem. Inf. Model. 2021, 61, 5377−5385

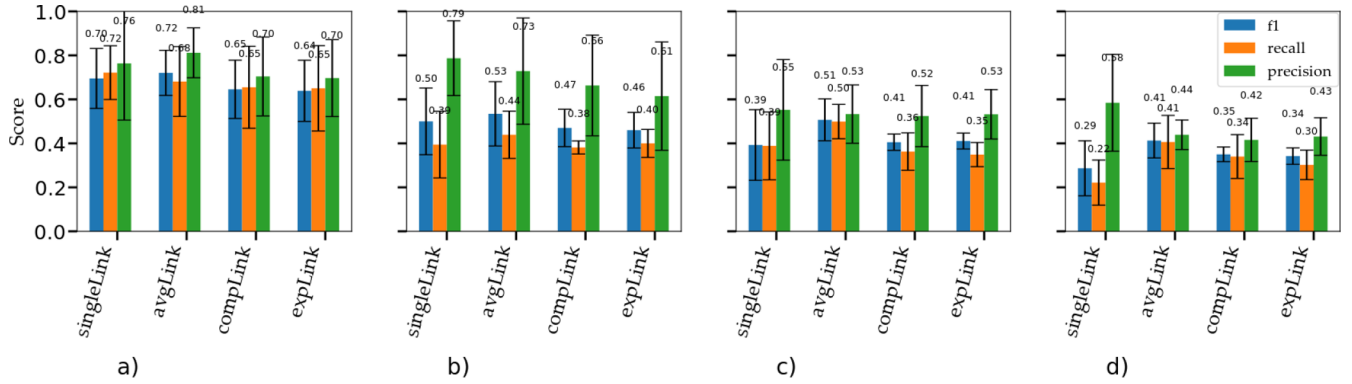


Figure 4. Model performance metrics for the different experiments we considered. (a) FastText word embeddings with the parameterized cosine distance, (b) fastText word embeddings with the Mahalanobis distance, (c) Word2Vec word embeddings with the parameterized cosine distance, and (d) Word2Vec word embeddings with the Mahalanobis distance. The *X*-axis refers to different linkage functions used for hierarchical clustering.

extension of the Euclidean distance. The cosine distance is a natural metric space for word embeddings as it is known that word embeddings similar in meaning have high dot products.[20] Observe that using the parameterized cosine distance increases the recall of the model, clearly visible in the case of fastText, as shown in Figure 4a,b. This indicates that the number of false negatives has gone down, and hence, a better cluster boundary is being learned. This implies that the parameterized cosine distance is a better dissimilarity function for points in the word vector space.

Comparing the linkage functions, we note that there is no clear winner across precision, recall, and *F*1 score. In terms of the *F*1 score, the average linkage appears to be on average better than the other linkages considered across the four models shown in Figure 4. Interestingly, the exponential linkage does not outperform the other linkages, which is in contrast to the result reported in ref 22 for the EXPα training procedure and the fact that the exponential linkage is designed to interpolate between the other three linkages. As the average linkage tends to discover spherical clusters and the exponential linkage can recover clusters of complex shapes, this indicates that the clusters in our data set in the parameterized space are spherical to begin with and the simpler model performs better. Note that the best models shown in Figure 4 have *F*1 scores around ∼0.8. The model's ability to generalize to unseen test data indicates that it does not overfit to the training data.

The pairwise classification approach does not take into account the structure of the cluster and instead looks at only instance level information. SynSetMine, in contrast, utilizes the signal from the cluster structure but still does so using a binary classifier. The supervised clustering approach does not rely on classification but performs clustering utilizing the set structure. The performance trend of the models suggests that utilizing the structure of the sets and using clustering instead of classification is the best approach to follow for entity normalization. The fact that supervised clustering "works" in this case indicates that word embeddings corresponding to material entities can be mapped to a vector space where entities referring to the same material occur "close" to each other. This has been reported previously for noun phrase co-referencing,[22] but this is the first time this idea has been validated for material entities.

We would expect fastText to identify similarities in cases such as "polyethylene" and "poly(ethylene)." However, fastText would have a hard time distinguishing polymers such as "poly(*o*-toluidine)," "poly(*m*-toluidine)," and "poly(*p*-toluidine)," which are spelled similarly and are likely to occur in similar contexts but are chemically different. As Word2Vec generates a vector representation using only contextual cues, we expect that combining the predictions from Word2Vec and fastText would perform better in such cases. To verify this, we ensemble the Word2Vec and fastText models. For a given pair of polymers in the test set, we take a majority vote of all the models being ensembled and using that, we predict whether the two polymers belong to the same cluster or different clusters. We show the results of ensembling averaged over the same train-val-test splits as used before in Figure 5. Only the

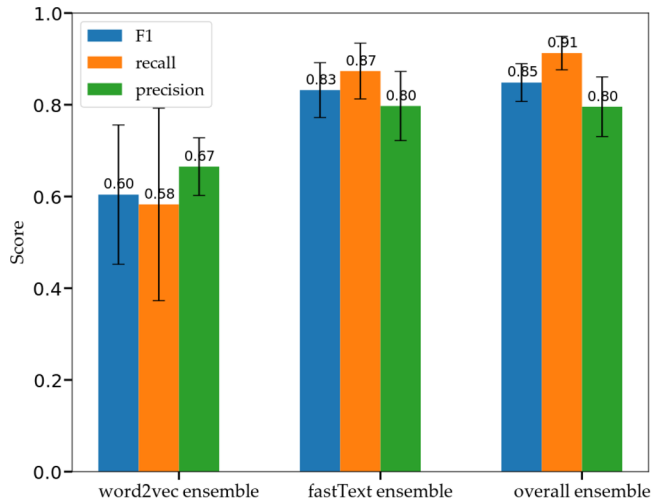


Figure 5. Performance obtained on ensembling the model. The Word2Vec ensemble and the fastText ensemble correspond to an ensemble over the linkage functions used for each of these models. The overall ensemble is computed over Word2Vec and fastText.

model trained using the parameterized cosine distance is considered. The Word2Vec ensemble and the fastText ensemble correspond to an ensemble over all the linkage functions used for each of these models. The "overall ensemble" corresponds to an ensemble over the fastText and Word2Vec models. The precision increases as we go from Word2Vec to fastText, as shown in Figures 4 and 5. The lower precision of Word2Vec indicates that the generated clusters are larger than expected ground truth clusters as several true

https://doi.org/10.1021/acs.jcim.1c00554
J. Chem. Inf. Model. 2021, 61, 5377−5385

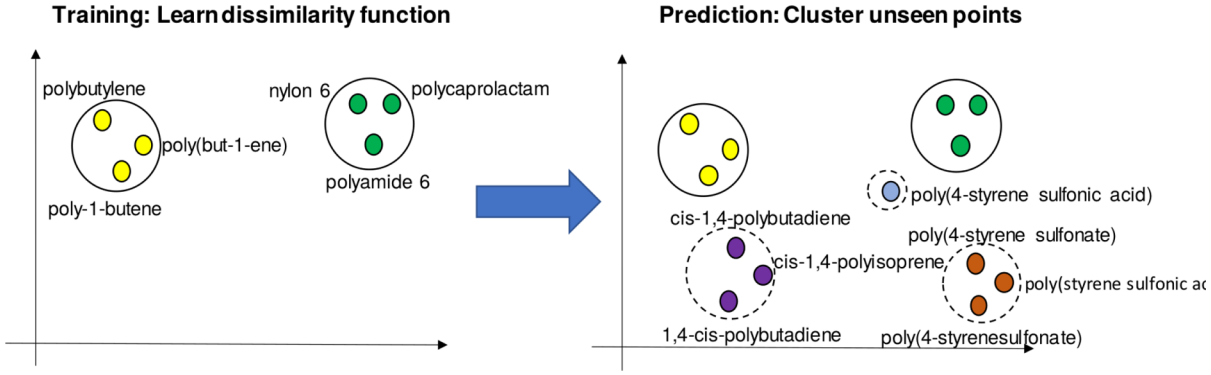


Figure 6. Supervised clustering workflow. On the left, the PNEs in the solid circles are known clusters that are used to learn a parameterized distance function and a decision threshold. The resulting model is used to cluster unseen points (shown as dashed circles on the right). The predicted clusters shown on the right are the outputs of using our trained model for inference on unlabeled points.

clusters are being merged. As Word2Vec only has access to coarser contextual information, chemically similar polymers such as polyalkenes, that is, polyhexene, polydecene, and polybutene are found grouped together. As we move to fastText, which has access to sub-word level information as well as contextual information, the increase in precision indicates that such "superclusters" are being broken up. Observe that the recall of the ensembled Word2Vec model is better than the recall for predictions from the various linkage functions in Figure 4c, which shows that ensembling helps in learning better cluster boundaries. The overall ensemble performed better than the Word2Vec ensemble and the fastText ensemble in terms of recall and $F1$ score.

Extrapolating to Unlabeled Data. The final model described in the preceding sections was applied to the rest of our data set, that is, the unlabeled points, to obtain the predicted clusters. The supervised clustering workflow followed is illustrated in Figure 6. In practice, the threshold for clustering granularity learned from the validation set in the labeled data set is much larger than the true clustering threshold. This is because the labeled data consist of only ∼15% of the total data set of PNEs, and hence, the density of points observed by the model is much lower than the true density of points given that the labeled data were uniformly distributed in the embedding space. In order to correct this, we scaled down the clustering granularity by trial and error till the number of predicted clusters is closer to the number we arrived at through the analysis in the Distribution of Polymer Names subsection. We found empirically that increasing the clustering threshold to adjust the number of predicted clusters close to 4100 as estimated earlier resulted in many polymers that are chemically similar but otherwise distinct getting clustered together, thus resulting in much larger cluster sizes. In order to avoid this, we used a smaller threshold. The threshold is picked such that the maximum size of predicted clusters is the same as the maximum size of curated clusters in the labeled data set. The final number of predicted clusters was 6002 for ∼13,000 PNEs in the unlabeled set. The disparity between the estimated number of clusters and the number of predicted clusters arises as the trained model is unable to learn perfect cluster boundaries for the unlabeled set on account of only a fraction of the data set being labeled. Consequently, we pick a threshold that errs on the side of polymers belonging to the same cluster being separated as opposed to dissimilar polymers being clustered together. Despite the predicted clusters not being perfect, they serve as a starting point for human annotators to further curate this data set, without which this problem would be intractable.

We now analyze some of the predictions of the model shown in Figure 6. "*Cis*-1,4-polybutadiene," "1,4-*cis*-polybutadiene," and "*cis*-1,4-polyisoprene" are clustered together of which the first two are true positives and the third is a false positive. "*Cis*-1,4-polyisoprene" and "*cis*-1,4-polybutadiene" are structurally similar as they are both dienes and both are used in the rubber industry[39,40] and are hence likely to occur in similar contexts. As another example, "poly(4-styrene sulfonate)," "poly(4-styrenesulfonate)," and "poly(styrene sulfonic acid)" occur in the same cluster. Although the sulfonate here is structurally the anionic form of the acid, it can be regarded as a true positive. However, "poly(4-styrene sulfonic acid)," which belongs to this cluster, is misclassified into a separate singleton cluster. This indicates that the partition threshold is not being learned perfectly. Other examples of successfully normalized entities include "sodium poly(acrylate)" and "sodium poly(acrylic acid)," "polyethoxysiloxane" and "polyethoxysiloxanes," and "polyethylene-octene" and "poly(ethylene-octene)." An interesting example of two very dissimilar PNEs that were normalized successfully was "poly(Asp)" and "poly-L-aspartic acid," although "sodium salt of poly(acrylic acid)" is also erroneously predicted to be in this cluster. These examples show that the model learns nontrivial "transformations" that go beyond word stem similarity. Selected examples of predicted clusters are shown in Table 3.

This data set of labeled clusters plus predicted clusters will be used as one component of our polymer informatics ecosystem that will enable the normalization of PNEs when detected in the literature through an NER model. This will allow us to reference multiple mentions of the same polymer in different ways to a unique record. We release this data set along with our code.

SUMMARY AND OUTLOOK

Normalizing polymer names is an essential ingredient to indexing polymers and utilizing literature extracted data for polymer informatics. This is the first work to comprehensively study the normalization of PNEs. We report the use of supervised clustering models trained on Word2Vec and fastText word embeddings for a data set of PNEs. We compare two distance functions, namely the Mahalanobis distance and the parameterized cosine distance. We find that the parameterized cosine distance outperforms the Mahalanobis distance metric. We also find that fastText embeddings

https://doi.org/10.1021/acs.jcim.1c00554
J. Chem. Inf. Model. 2021, 61, 5377−5385

Table 3. Examples of Predicted Clusters[a]

polymer name or key	PNEs in the predicted cluster
poly(styrene sulfonic acid)	poly(4-styrene sulfonate), poly(4-styrenesulfonate), poly(styrene sulfonic acid)
poly(4-styrene sulfonic acid)	poly(4-styrene sulfonic acid)
cis-1,4-polyisoprene	*cis*-1,4-polybutadiene, 1,4-*cis*-polybutadiene, *cis*-1,4-poly(butadiene), *cis*-1,4-polyisoprene, *cis*-1,4-poly(isoprene)
sodium poly(acrylic acid)	sodium poly(acrylate), sodium poly(acrylic acid)
polyethoxysiloxane	polyethoxysiloxane, polyethoxysiloxanes, polyethoxylsiloxane
polyethylene-octene	polyethylene-octene, poly(ethylene-octene)
sodium salt of poly(acrylic acid)	poly(Asp), poly-L-aspartic acid, sodium salt of poly(acrylic acid)

[a]The key used here is the most frequently occurring entity in that cluster.

outperform Word2Vec, which indicates that sub-word information is useful for normalization. Ensembling Word2Vec and fastText produces better results than either model alone and results in the best performing normalization model with an *F*1 score of 0.85. We outperform the recall (and *F*1) of our baselines for this task by a considerable margin. We have created a data set of 15,500 PNEs from a corpus of ∼0.5 million papers and have detected a total of 6634 polymer clusters (632 of which are manually curated).

Our models enable normalization given a predefined list of PNEs. An area for future work would be on-the-fly normalization, that is, given a new piece of text, identify whether a new PNE is identified in the text, not existing in the dictionary, is a new polymer, or is a different way to refer to a polymer already in the dictionary. This would be a key component for enabling semiautonomous data gathering for materials property information. This data would serve to accelerate our past attempts at rational design of application-specific polymers.[41−47]

■ ASSOCIATED CONTENT

Supporting Information

The Supporting Information is available free of charge at https://pubs.acs.org/doi/10.1021/acs.jcim.1c00554.

Additional information on the baseline model, OPSIN, and proof of result used in the Distribution of Polymer Names subsection (PDF)

■ AUTHOR INFORMATION

Corresponding Author

Rampi Ramprasad − *School of Materials Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, United States;* orcid.org/0000-0003-4630-1565; Email: rampi.ramprasad@mse.gatech.edu

Author

Pranav Shetty − *School of Computational Science & Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332, United States;* orcid.org/0000-0003-2015-9556

Complete contact information is available at: https://pubs.acs.org/10.1021/acs.jcim.1c00554

Notes

The authors declare no competing financial interest.

The code and data mentioned in this paper can be found at https://github.com/Ramprasad-Group/polymerNEN.

■ ACKNOWLEDGMENTS

This work was supported by the Office of Naval Research through grants N00014-19-1-2103 and N00014-20-1-2175. The authors also acknowledge useful discussions and feedback from Dr. Lihua Chen, Dr. Christopher Kuenneth, and Dr. Arun Rajan. We also thank the reviewers for their comments.

■ REFERENCES

(1) Batra, R.; Song, L.; Ramprasad, R. Emerging materials intelligence ecosystems propelled by machine learning. *Nat. Rev. Mater.* **2021**, *6*, 655−678.

(2) Doan Tran, H.; Kim, C.; Chen, L.; Chandrasekaran, A.; Batra, R.; Venkatram, S.; Kamal, D.; Lightstone, J. P.; Gurnani, R.; Shetty, P.; Ramprasad, M.; Laws, J.; Shelton, M.; Ramprasad, R. Machine-learning predictions of polymer properties with Polymer Genome. *J. Appl. Phys.* **2020**, *128*, 171104.

(3) Chen, L.; Pilania, G.; Batra, R.; Huan, T. D.; Kim, C.; Kuenneth, C.; Ramprasad, R. Polymer informatics: Current status and critical next steps. *Mater. Sci. Eng., R* **2021**, *144*, 100595.

(4) Meredig, B. Five high-impact research areas in machine learning for materials science. *Chem. Mater.* **2019**, *31*, 9579−9581.

(5) Kim, E.; Huang, K.; Saunders, A.; McCallum, A.; Ceder, G.; Olivetti, E. Materials synthesis insights from scientific literature via text extraction and machine learning. *Chem. Mater.* **2017**, *29*, 9436−9444.

(6) Kononova, O.; Huo, H.; He, T.; Rong, Z.; Botari, T.; Sun, W.; Tshitoyan, V.; Ceder, G. Text-mined dataset of inorganic materials synthesis recipes. *Sci. Data* **2019**, *6*, 203.

(7) Tshitoyan, V.; Dagdelen, J.; Weston, L.; Dunn, A.; Rong, Z.; Kononova, O.; Persson, K. A.; Ceder, G.; Jain, A. Unsupervised word embeddings capture latent knowledge from materials science literature. *Nature* **2019**, *571*, 95−98.

(8) Shetty, P.; Ramprasad, R. Automated knowledge extraction from polymer literature using natural language processing. *iScience* **2021**, *24*, 101922.

(9) Mallory, E. K.; de Rochemonteix, M.; Ratner, A.; Acharya, A.; Re, C.; Bright, R. A.; Altman, R. B. Extracting chemical reactions from text using Snorkel. *BMC Bioinf.* **2020**, *21*, 217.

(10) Li, Y.; Shetty, P.; Liu, L.; Zhang, C.; Song, L BERTifying the Hidden Markov Model for Multi-Source Weakly Supervised Named Entity Recognition. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* **2021**, 6178.

(11) Ratner, A.; Bach, S. H.; Ehrenberg, H.; Fries, J.; Wu, S.; Re Snorkel, C.: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 2017; Vol. *11*, pp 269−282.

(12) Hu, B.; Lin, A.; Brinson, L. C. ChemProps: A RESTful API enabled database for composite polymer name standardization. *J. Cheminf.* **2021**, *13*, 22.

(13) Cho, H.; Choi, W.; Lee, H. A method for named entity normalization in biomedical articles: application to diseases and plants. *BMC Bioinf.* **2017**, *18*, 451.

(14) Kim, D.; Lee, J.; So, C. H.; Jeon, H.; Jeong, M.; Choi, Y.; Yoon, W.; Sung, M.; Kang, J. A neural named entity recognition and multi-type normalization tool for biomedical text mining. *IEEE Access* **2019**, *7*, 73729−73740.

(15) Leaman, R.; Wei, C.-H.; Lu, Z. tmChem: a high performance approach for chemical named entity recognition and normalization. *J. Cheminf.* **2015**, *7*, S3.

(16) Hiszpanski, A. M.; Gallagher, B.; Chellappan, K.; Li, P.; Liu, S.; Kim, H.; Han, J.; Kailkhura, B.; Buttler, D. J.; Han, T. Y.-J. Nanomaterial synthesis insights from machine learning of scientific

articles by extracting, structuring, and visualizing knowledge. *J. Chem. Inf. Model.* **2020**, *60*, 2876−2887.
(17) Hodge, P.; Hellwich, K.-H.; Hiorns, R. C.; Jones, R. G.; Kahovec, J.; Luscombe, C. K.; Purbrick, M. D.; Wilks, E. S. A concise guide to polymer nomenclature for authors of papers and reports in polymer science and technology (IUPAC Technical Report). *Pure Appl. Chem.* **2020**, *92*, 797−813.
(18) Akbari, B.; Lashanizadegan, A.; Darvishi, P.; Pouranfard, A.-R. Preparation of hydrophobic flat sheet membranes from PVDF-HFP copolymer for enhancing the oxygen permeance in nitrogen/oxygen gas mixture. *Chin. J. Chem. Eng.* **2020**, *28*, 1566−1581.
(19) Chen, Y.; Cheng, Y.; Jie, Y.; Cao, X.; Wang, N.; Wang, Z. L. Energy harvesting and wireless power transmission by a hybridized electromagnetic-triboelectric nanogenerator. *Energy Environ. Sci.* **2019**, *12*, 2678−2684.
(20) Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; Dean, J. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, 3111−3119.
(21) Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135−146.
(22) Yadav, N.; Kobren, A.; Monath, N.; McCallum, A. Supervised hierarchical clustering with exponential linkage. *International Conference on Machine Learning*, 2019; pp 6973−6983.
(23) Swain, M. C.; Cole, J. M. ChemDataExtractor: a toolkit for automated extraction of chemical information from the scientific literature. *J. Chem. Inf. Model.* **2016**, *56*, 1894−1904.
(24) van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579−2605.
(25) Otsuka, S.; Kuwajima, I.; Hosoya, J.; Xu, Y.; Yamazaki, M. PoLyInfo: Polymer database for polymeric materials design. 2011. *International Conference on Emerging Intelligent Data and Web Technologies*, 2011; pp 22−29.
(26) Tomasulo, P. ChemIDplus-super source for chemical and drug information. *Med. Ref. Serv. Q.* **2002**, *21*, 53−59.
(27) Kim, S.; Thiessen, P. A.; Bolton, E. E.; Chen, J.; Fu, G.; Gindulyte, A.; Han, L.; He, J.; He, S.; Shoemaker, B. A.; Wang, J.; Yu, B.; Zhang, J.; Bryant, S. H. PubChem substance and compound databases. *Nucleic Acids Res.* **2016**, *44*, D1202−D1213.
(28) Kim, E.; Jensen, Z.; van Grootel, A.; Huang, K.; Staib, M.; Mysore, S.; Chang, H.-S.; Strubell, E.; McCallum, A.; Jegelka, S.; Olivetti, E. Inorganic materials synthesis planning with literature-trained neural networks. *J. Chem. Inf. Model.* **2020**, *60*, 1194−1201.
(29) Weston, L.; Tshitoyan, V.; Dagdelen, J.; Kononova, O.; Trewartha, A.; Persson, K. A.; Ceder, G.; Jain, A. Named entity recognition and normalization applied to large-scale information extraction from the materials science literature. *J. Chem. Inf. Model.* **2019**, *59*, 3692−3702.
(30) Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.* **1966**, *10*, 707−710.
(31) Shen, J.; Lyu, R.; Ren, X.; Vanni, M.; Sadler, B.; Han, J. Mining entity synonyms with efficient neural set generation. *Proc. Conf. AAAI Artif. Intell.* **2019**, *33*, 249−256.
(32) Eick, C. F.; Zeidat, N.; Zhao, Z. Supervised clustering-algorithms and benefits. *Proceedings. 16th IEEE International Conference on Tools with Artificial Intelligence*, 2004; pp 774−776.
(33) Finley, T.; Joachims, T. Supervised clustering with support vector machines. *Proceedings. 22nd International Conference on Machine Learning*, 2005; pp 217−224.
(34) De Maesschalck, R.; Jouan-Rimbaud, D.; Massart, D.L. The mahalanobis distance. *Chemom. Intell. Lab. Syst.* **2000**, *50*, 1−18.
(35) Basu, S.; Bilenko, M.; Mooney, R. J. A probabilistic framework for semi-supervised clustering. *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004; pp 59−68.
(36) Lukasová, A. Hierarchical agglomerative clustering procedure. *Pattern Recogn.* **1979**, *11*, 365−381.
(37) Xu, R.; WunschII, D. Survey of clustering algorithms. *IEEE Trans. Neural Network.* **2005**, *16*, 645−678.
(38) Newman, M. E. J. Power laws, Pareto distributions and Zipf's law. *Contemp. Phys.* **2005**, *46*, 323−351.
(39) Wrana, C.; Schawe, J. E. Isothermal crystallization of cis-1.4-polybutadiene at low temperatures. *Thermochim. Acta* **2020**, *690*, 178669.
(40) Rudin, A.; Choi, P. *The Elements of Polymer Science and Engineering*; Academic press, 2012.
(41) Wu, C.; Chen, L.; Deshmukh, A.; Kamal, D.; Li, Z.; Shetty, P.; Zhou, J.; Sahu, H.; Tran, H.; Sotzing, G.; Ramprasad, R.; Cao, Y. Dielectric Polymers Tolerant to Electric Field and Temperature Extremes: Integration of Phenomenology, Informatics, and Experimental Validation. *ACS Appl. Mater. Interfaces* **2021**, DOI: 10.1021/acsami.1c11885.
(42) Mannodi-Kanakkithodi, A.; Chandrasekaran, A.; Kim, C.; Huan, T. D.; Pilania, G.; Botu, V.; Ramprasad, R. Scoping the polymer genome: A roadmap for rational polymer dielectrics design and beyond. *Mater. Today* **2018**, *21*, 785−796.
(43) Ma, R.; Baldwin, A. F.; Wang, C.; Offenbach, I.; Cakmak, M.; Ramprasad, R.; Sotzing, G. A. Rationally designed polyimides for high-energy density capacitor applications. *ACS Appl. Mater. Interfaces* **2014**, *6*, 10445−10451.
(44) Ma, R.; Sharma, V.; Baldwin, A. F.; Tefferi, M.; Offenbach, I.; Cakmak, M.; Weiss, R.; Cao, Y.; Ramprasad, R.; Sotzing, G. A. Rational design and synthesis of polythioureas as capacitor dielectrics. *J. Mater. Chem. A* **2015**, *3*, 14845−14852.
(45) Huzayyin, A.; Boggs, S.; Ramprasad, R. Density functional analysis of chemical impurities in dielectric polyethylene. *IEEE Trans. Dielectr. Electr. Insul.* **2010**, *17*, 926−930.
(46) Baldwin, A. F.; Huan, T. D.; Ma, R.; Mannodi-Kanakkithodi, A.; Tefferi, M.; Katz, N.; Cao, Y.; Ramprasad, R.; Sotzing, G. A. Rational design of organotin polyesters. *Macromolecules* **2015**, *48*, 2422−2428.
(47) Baldwin, A. F.; Ma, R.; Mannodi-Kanakkithodi, A.; Huan, T. D.; Wang, C.; Tefferi, M.; Marszalek, J. E.; Cakmak, M.; Cao, Y.; Ramprasad, R.; Sotzing, G. Poly(dimethyltin glutarate) as a Prospective Material for High Dielectric Applications. *Adv. Mater.* **2015**, *27*, 346−351.

https://doi.org/10.1021/acs.jcim.1c00554
J. Chem. Inf. Model. 2021, 61, 5377−5385