scientific data

OPEN

ARTICLE

Materials information extraction via automatically generated corpus

Rongen Yan[1], Xue Jiang[2,3], Weiren Wang[2], Depeng Dang[1] ✉ & Yanjing Su[2] ✉

Information Extraction (IE) in Natural Language Processing (NLP) aims to extract structured information from unstructured text to assist a computer in understanding natural language. Machine learning-based IE methods bring more intelligence and possibilities but require an extensive and accurate labeled corpus. In the materials science domain, giving reliable labels is a laborious task that requires the efforts of many professionals. To reduce manual intervention and automatically generate materials corpus during IE, in this work, we propose a semi-supervised IE framework for materials via automatically generated corpus. Taking the superalloy data extraction in our previous work as an example, the proposed framework using Snorkel automatically labels the corpus containing property values. Then Ordered Neurons-Long Short-Term Memory (ON-LSTM) network is adopted to train an information extraction model on the generated corpus. The experimental results show that the F1-score of γ' solvus temperature, density and solidus temperature of superalloys are 83.90%, 94.02%, 89.27%, respectively. Furthermore, we conduct similar experiments on other materials, the experimental results show that the proposed framework is universal in the field of materials.

Introduction

Natural Language Processing (NLP) focuses on a computer understanding text knowledge so that a computer can analyze and process natural language[1]. Information Extraction (IE) in NLP is one of the most prominent text mining technologies and aims to extract structured information from unstructured text[2]. Scientific literature in the field of materials contains a large number of reliable data, which promotes data-driven materials research and development[3–5]. It is time-consuming to rely solely on human manual extraction[6]. So, automatic data extraction of organic and inorganic chemical substances from articles in the fields of chemistry and materials science have made their sense using NLP techniques[7–11].

With the development of machine learning and NLP, IE technology has developed rapidly[6], particularly in biology and medicine. Sunil *et al*. proposed that IE is a process of detecting and classifying semantic relations and used a Convolutional Neural Network (CNN) to obtain semantic features to extract the information in the biomedical domain[12]. Many papers have applied deep learning models for feature optimization; for example, Xinbo *et al*. used Conditional Random Fields (CRFs) to classify the features of the context and used autoencoders and sparsity limitations to solve the problem of word sparsity[13]. Recently, other IE systems have also been investigated in the search for possible information with Long Short-Term Memory (LSTM). Raghavendra *et al.* embedded words into bidirectional LSTM and CRF. They used a recurrent neural network to obtain features and completed clinical concept extraction[14]. Arshad *et al.* presented an LSTM method for understanding language grammar and deducing the relationship between words[15]. However, all of the above neural networks require an extensive and accurate labeled corpus to train the network.

Unfortunately, there are relatively few papers on many materials subjects, such as superalloys, extracting the required information from the paper becomes a tricky job. In our previous work[11], we developed an NLP pipeline to capture both chemical composition and property data from superalloys scientific literature. A rule-based Named Entity Recognition (NER) method and a distance-based heuristic multiple-relation extraction algorithm for the pipeline were proposed to overcome the drawback of limited training corpus labels, and achieve high precision and recall simultaneously. The proposed IE algorithm is a rule-based method, while the machine learning method was abandoned after comparison because the labeled corpus was not enough for training. It is a laborious task that requires the efforts of many professionals if completed by humans alone. Rule-based

[1]School of Artificial Intelligence, Beijing Normal University, Beijing, 100875, China. [2]Beijing Advanced Innovation Center for Materials Genome Engineering, Institute for Advanced Materials and Technology, University of Science and Technology Beijing, Beijing, 100083, China. [3]Collaborative Innovation Center of Steel Technology, University of Science and Technology Beijing, Beijing, 100083, China. ✉e-mail: ddepeng@bnu.edu.cn; yjsu@ustb.edu.cn

strategy is efficient under such conditions but without the ability to learn and update independently. Therefore, automatically generating corpus in the material domain, enabling to reduce manual intervention, is necessary for machine learning-based IE, which will make it a reality for computers to read papers and extract datasets by themselves.

Two problems are inevitable when faced with machine learning problems: data and algorithms. With the improvement of various machine learning frameworks, the application threshold of algorithms is gradually decreasing. However, the acquisition of data is still a labor-intensive and necessary process. At work, we usually face the following problem: the task has a lot of corpus, but none of them have reliable labels. In response to the above problems, the usual methods are unsupervised learning of transferable features, the combination of rule system and model or simple stacking rule system, semi-supervised methods to expand label data, increase manual verification and annotation[16]. But these methods are either too cumbersome to operate, too expensive, or too inflexible. Based on this, Snorkel[16] as a data programming framework that enables fast dataset construction and model training has been proposed by a research team at Stanford University.

In this work, we propose a semi-supervised IE framework for materials domain via automatically generated corpus. Taking the superalloy data extraction in the previous work as an example, the proposed framework using Snorkel[17] automatically labels the corpus containing the name of a superalloy and its corresponding property values. We first put the labeling function written according to the sentence characteristics of scientific literature into the Snorkel function training process and then obtain the accurate training set. Semi-supervised is embodied in human-written labeling functions rather than augmenting the data. Finally, we use the popular Ordered Neurons-LSTM (ON-LSTM)[18] network to train an information extraction model on this automated training corpus and extract property values in the science literature of materials. We obtain about 18% higher results using ON-LSTM than traditional LSTM on the information extraction task. The code is available at https://github.com/MGEdata/auto-generate-corpus. Our contributions are summarized as follows:

- New IE framework is proposed for materials using the semi-supervised method in machine learning to automatically generate corpus. These works are completed based on the previous work[11], and further extract the information in the material field.
- ON-LSTM is used to complete the task of IE. To the best of our knowledge, this is the first time that ON-LSTM and IE are combined to explore the possibility of potential integration.
- Experimental results show that the method proposed in this paper can effectively extract information and be applied to broad materials subjects.

Results

Information extraction workflow.

Our method of extracting material information by automatically generating corpus involves the following steps: NER, generating candidate sets, Snorkel framework and training model, as shown in Fig. 1. In order to explain the algorithm workflow in more detail and more vividly, we take γ' solvus temperature of superalloy as an example. The initial corpus we use is to use the NER method to mark the superalloy name and property value in a sentence. The specific method of NER is detailed in our previous article[11]. However, the initial corpus marks all the superalloy names and property values in a sentence, depending on NER can not accurately find the matching mode of superalloy names and property values if there are multiple superalloy names and property values in a sentence. The next step is to generate candidates. The following is a sample sentence describing the γ' solvus temperature of superalloys:

The γ' solvus temperatures of X_1, X_2 and X_3 are Y_1, Y_2 and Y_3, respectively.

This sentence involves three superalloys and their γ' solvus temperatures. In this sentence, X_i represents the i-th superalloy, and Y_i represents the value of the i-th γ' solvus temperature. In this example, the task that we need to complete is to find their correct pairing: (X_1, Y_1), (X_2, Y_2) and (X_3, Y_3). We define the candidates as an exhaustive combination of the names of superalloys X_1, X_2, X_3 and γ' solvus temperatures Y_1, Y_2, Y_3. Therefore, there are 9 candidates: (X_1, Y_1), (X_1, Y_2), (X_1, Y_3), (X_2, Y_1), (X_2, Y_2), (X_2, Y_3), (X_3, Y_1), (X_3, Y_2), (X_3, Y_3). If there are m names of superalloy and n γ' solvus temperatures in a sentence, $m*n$ candidates will be generated.

In the third step, we write some labeling functions in the Snorkel framework, a semi-supervised method to screen the candidates, and get the correct pairing of the superalloy name and the γ' solvus temperature. So far, we have accurately found the relationship to be extracted and generated the corpus we need. Finally, we use the deep learning model ON-LSTM training model in these corpora, so that the new corpora directly extract the required relationship by using the training model.

Automatic generation of training corpus.

Generation principle. No public corpora of IE can be leveraged due to the few literature in the field of superalloy. Therefore, to train a model in this field, the training corpora problem can be solved through manual search[19]. Snorkel proposes the radical idea that mathematical and systematic structure can be provided for the messy and often entirely manual process of training data creation and management, starting by empowering users to label, build and manage training corpora programmatically.

The third part of Fig. 1 shows the specific process of Snorkel framework. The main advantage of the Snorkel framework is that there is no need to label the dataset manually. When the task changes, the data may need to be relabeled, expanded, or ignored[20]. Users only need to pay attention to the characteristics of each dataset and write labeling functions for the dataset that can automatically determine true and false for candidates. However, Snorkel only proposes a framework for generating the training data and is not designed for a specific field; in previous work[20], used Snorkel in the field of chemistry. In this work, we develop an application of Snorkel that is a weakly supervised learning framework for generating corpora from the scientific literature.

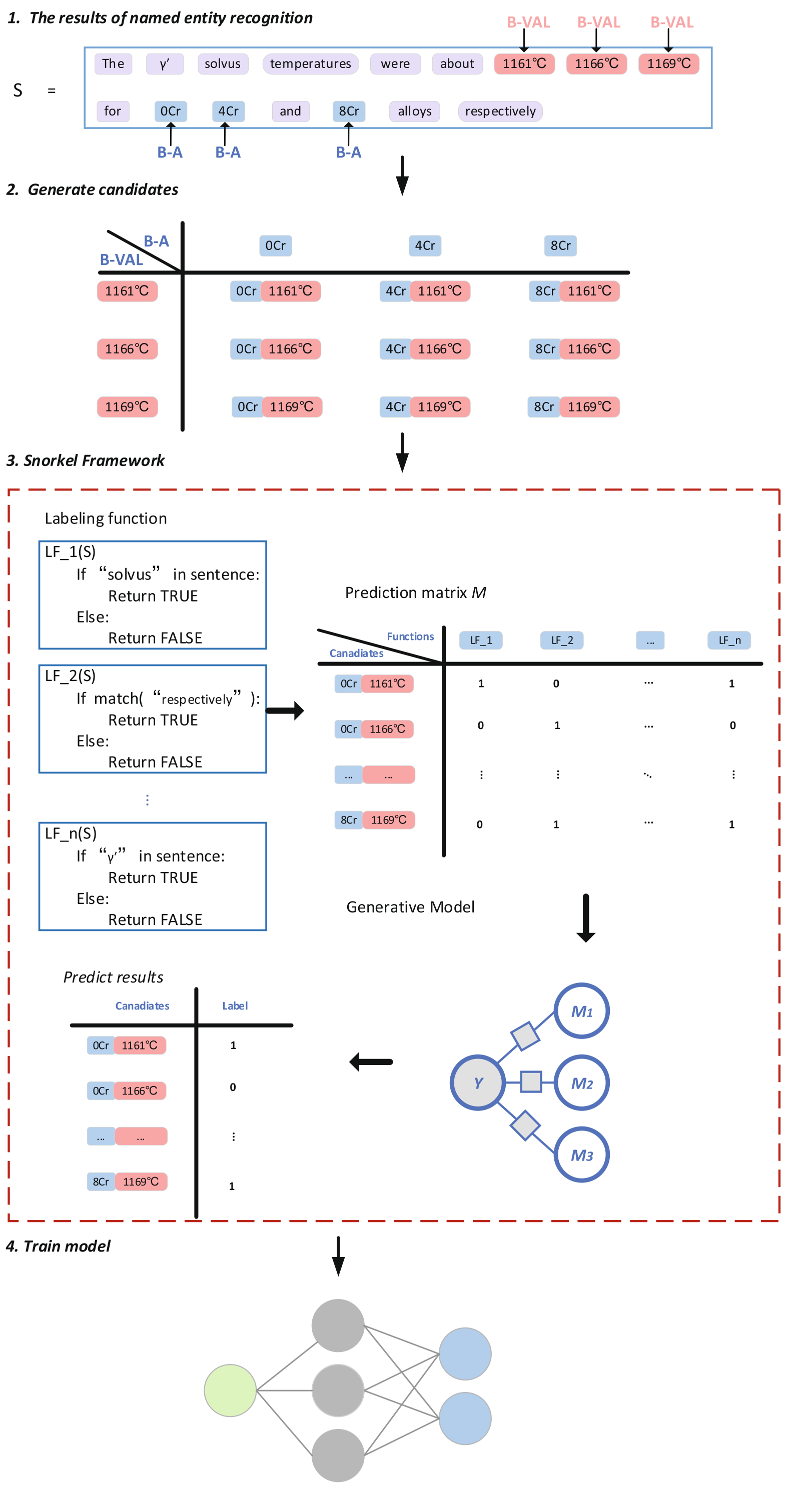
1. The results of named entity recognition
S =
The γ′ solvus temperatures were about 1161℃ 1166℃ 1169℃
for 0Cr 4Cr and 8Cr alloys respectively
B-VAL
B-A
2. Generate candidates
B-A
B-VAL
0Cr
4Cr
8Cr
1161℃
1166℃
1169℃
3. Snorkel Framework
Labeling function
LF_1(S) If "solvus" in sentence: Return TRUE Else: Return FALSE
LF_2(S) If match("respectively"): Return TRUE Else: Return FALSE
LF_n(S) If "γ′" in sentence: Return TRUE Else: Return FALSE
Prediction matrix M
Functions
Canadiates
LF_1
LF_2
LF_n
Generative Model
Y
M1
M2
M3
Predict results
Canadiates
Label
4. Train model

Fig. 1 Process for information extraction. Among them, B-A represents the name of the superalloy, and B-Val represents the property value. *LF_1*, *LF_2*, …, *LF_n* represent the name of labeling functions.

To generate candidates, we use rules to label all relevant words about superalloys and γ' solvus temperature from the scientific literature. We exhaust all combinations of the marked superalloys and γ' solvus temperature to form candidate sets and then judge them through labeling functions. The generative model in Snorkel calculates the accuracy and relevance of the candidate sets based on the consistency and divergence of the written labeling functions. Based on the labeling functions, the generative model does not require actual data and directly judges whether the candidate is right or wrong. Each candidate will be evaluated by all labeling functions to obtain a reasonable result. Candidates are judged correctly, forming the target corpora.

Labeling function	Description
LF_temperature_words	If the sentence contains words related to temperature, such as solvus, γ', we label True.
LF_for	If the sentence contains " 'γ' solvus temperature' for 'superalloy'", we label True.
LF_oneMatch	If there is only one superalloy and one γ' solvus temperature in a sentence, we label True.
LF_temperature_left	If the γ' solvus temperature is included in the parentheses after the superalloy, we label True.
LF_alloy_twoExpress	If a superalloy has two expressions, we label True.
LF_in	If there is a sentence of " 'γ' solvus temperature' in 'superalloy'", we label True.
LF_equal	If there is a keyword "equal" near the superalloy and γ' solvus temperature, we label True.
LF_hasTem	If there is a sentence pattern "has a temperature of", we label True.
LF_between_and	If the γ' solvus temperature is within a certain interval, we label True.
LF_be	If it is clear what the γ' solvus temperature of the superalloy is, we label True.

Table 1. Examples of labeling functions. 'Description' is an explanation of what the label function does.

Data source. For superalloys in materials, we use rule-based methods to classify sentences containing the name of the superalloys and corresponding property values from more than 14,425 full texts of scientific journal articles related to material. Similar to our previous work[11], these articles are accessed through Elsevier Research Products APIs allowing anyone that can obtain an API Key and use the APIs for non-commercial purposes free of charge. The detailed information about Elsevier Research Products APIs can refer https://dev.elsevier.com. After the application is approved, the website will assign an API Key to each user. Through the API Key, we can obtain articles in plain text and XML formats. Once we have the articles, we can perform text mining on the articles. Additionally, we uploaded the dois of 14,425 articles in the supplementary material. The extracted superalloys include two types, Co-based and Ni-based superalloys that account for more than 80% of all superalloys. Sentences containing the property values of superalloys are generally included in the full text, so that we consider the full text of science journal articles. The article about superalloys includes many properties, we focus on three of them: γ' solvus temperature, solidus temperature and density. Among them, 457 sentences related to the γ' solvus temperature. The initial corpus has been published on https://github.com/MGEdata/snorkel. Although only relatively few sentences are obtained, the number of sentences is already quite high for the field of superalloys. In some cases, multiple names and property values are mentioned in one sentence. To accurately match the superalloy and the γ' solvus temperatures, all of the combinations were exhaustively generated to obtain 1,184 pairs. The matched candidate is marked by Snorkel to form corpora. The corpora obtained in this manner reflects the influence of the labeling function on the extraction.

Effect of labeling functions on candidate. Each dataset has unique characteristics, and labeling functions are customized according to the characteristics of the dataset. If users want to use our proposed framework to extract the relationship in their own corpus, they only need to rewrite labeling functions that match the characteristics of the sentences in their corpus. The labeling functions have nothing to do with the source of the corpus, but only the characteristics of the sentence. The scientific literature on superalloys has a more professional vocabulary. We write more than 10 labeling functions according to their semantic characteristics for extracting γ' solvus temperature. Table 1 provides examples of labeling functions. We adjust the writing of the labeling function according to the coverage, overlaps and conflicts of different labeling functions. The list of labeling functions is shown in Table 2. The coverage of labeling functions refers to the proportion of positive and negative samples that are successfully labeled. At the γ' solvus temperature of the extracted superalloy, the comprehensive coverage of the labeling function we write reaches more than 90%. When users use the framework to write labeling functions, try to make the overall coverage of labeling functions as high as possible. To describe overlaps in a finer-grained way, we illustrate using an example. Suppose there are three candidates $c1$, $c2$, $c3$ and two labeling functions $LF1$, $LF2$. If the labeling function judges the candidate as right, it returns 1, if the candidate is judged as false, it returns 0. If the labeling function does not involve the candidate, it abstains and returns -1. The matrix formed by labeling functions $LF1$ and $LF2$ are $[1, -1, 0]$,$[1, -1, -1]$, respectively. Both $LF1$ and $LF2$ judge the first candidate, which is called overlap. Conflict means that two labeling functions involve the same candidate and the judgment results are inconsistent. The more the conflict tends to 0, the more specific the labeling functions are written. We print out the labeling functions through the labeling function analyzer PandasLFApplier on the official website of the Snorkel framework, and find that the conflict is 0. This indicates that there is no conflict between the labeling functions we write. An examination of the table shows that these labeling functions are comprehensive and precise. These functions have achieved good results. For example, LF_in has a 0.46 coverage of candidates.

Evaluation for generated corpus. The generative model judges the true or false of each candidate through given labeling functions, thereby transforming the task of generating the corpora into a classification task. It is well-known that the F1-score is a good measure for classification problems, and some classification problems often use the F1-score as the final evaluation metric. F1-score is the harmonic mean of precision and recall, that is, F1-score $= 2 * \frac{precision * recall}{precision + recall}$. Precision is given by $\frac{TP}{TP + FP}$, and recall is given by $\frac{TP}{TP + FN}$. Here, TP is truly positive, which is judged as a positive sample and in fact is a positive sample. FP is false positive, which is judged to be a positive sample, but in fact is a negative sample. FN is false negative, which is judged to be a negative sample, but in fact is a positive sample. The maximum value of the F1-score is 1, and the minimum value is 0.

Labeling function	Coverage	Overlaps	Conflicts
LF_temperature_words	0.227451	0.078431	0.0
LF_for	0.027451	0.027451	0.0
LF_oneMatch	0.074510	0.068627	0.0
LF_temperature_left	0.066667	0.064706	0.0
LF_alloy_twoExpress	0.007843	0.007843	0.0
LF_in	0.460784	0.176471	0.0
LF_equal	0.003922	0.003942	0.0
LF_hasTem	0.009804	0.009804	0.0
LF_between_and	0.007843	0.007843	0.0
LF_be	0.033333	0.033333	0.0

Table 2. Coverage, overlaps and conflicts of different labeling functions.

In addition to the F1-score, ROC[21] is also an indicator used to measure the imbalance of classification. In particular, ROC-auc is used to evaluate the pros and cons of a binary classifier. ROC-auc is defined as the area under the ROC curve. The ROC curve is generally on a straight line $y = x$, so the value range of all ROC-auc is between 0.5 and 1. In many cases, the ROC curve does not clearly indicate which classifier performs better, and ROC-auc is a numerical value. A larger value corresponds to better classifier effect. For the relationship between the value of ROC-auc and the classifier, we have a rough standard for evaluating the classifier. If ROC-auc is less than 0.5, the model has little discrimination ability. If ROC-auc is greater than 0.5 and less than 0.8, the discrimination ability of the model is acceptable. If the value of ROC-auc is greater than 0.8, the discrimination ability of the model performs better.

We divide the 1184 candidate sets of the γ' solvus temperature into the training set, development set and test set, consisting of 674, 200, and 310 candidate sets, respectively. To verify the effect of using Snorkel to generate the corpora, we invited domain experts to mark the development set and test set manually. Among the 1184 candidate sets, the experts annotate a total of 200 candidate sets as development. Although the manual workload is currently somewhat large, the trained model can generate a larger data set. The manual workload is limited to the early stage and the later use of machine processing will be much faster than manual processing. To date, the training set and test set have not been labeled, and the development set has been manually labeled. We embed the label functions in the Snorkel framework for the development set. The purpose is to extract the correct information from the training set to form the corpora.

The evaluation results of the automatically generated corpus are shown in Fig. 2. The number at the bottom of the figure is the epoch, and the vertical axis represents the specific value. When using the Snorkel framework, we use different epochs. When the model is trained, the effect of the model will become better as the epoch increases, but if we train too many epochs, the model will overfit the training data and the effect will decrease. Ideally, we want to find the inflection point where the model goes from good to bad to decide whether to stop training. After many experiments, we found that the best results are obtained when the epoch is 70. The best ROC-auc was 0.882, and the best F1-score was 0.839. The corresponding inflection point epoch is 70, and more epochs will cause overfitting, resulting in poorer results. These values indicate that the quality of the generated data set is high. Although these values vary slightly with different epoch, it can be seen from the figure that the difference is not significant. This shows that as long as the label function is written accurately, the snorkel learning ability is not highly correlated with the epoch.

Validation of the snorkel model. We obtained corpus using Snorkel. When judging whether candidates are right or wrong, we write the label function at the level of the candidate set. Since different candidates may from the same sentence, when verifying on the test set, the sentences in the test set may have been seen by the model during training. To illustrate the generality of our model, we add un-trained 88 sentences about γ' solvus temperature to generate 298 candidate sets.

We put the generated 298 candidate sets directly into the trained model, and judge each candidate. We invite experts to randomly select 50 pieces of corpora automatically generated by Snorkel for manual inspection. Table 3 is an example of the corpora corrected by experts. The correct pairing is selected from a large number of candidates. The results found that using the method of automatically generating corpus tags. The tag accuracy rate reached more than 80%. The first column labeled 1 is the correct pair, and the one labeled 0 is wrong. The 'name_id' and 'attri_id' respectively represent the position of the superalloy and γ' solvus temperature in a sentence.

Training of relation extraction model. With the large number of labeled corpora produced by the Snorkel, we can use these corpora to train a discriminant model. But we can't help but wonder why we need to train another discriminant model since the Snorkel can accurately determine the type of the sample? This question needs to start with the difference between the generative and the discriminant model. The generative model in Snorkel learns the joint probability distribution $P(X, Y)$ from the data, and then obtains the conditional probability distribution $P(Y|X)$ as the predictive model, the formula for generating the model is expressed as follows.

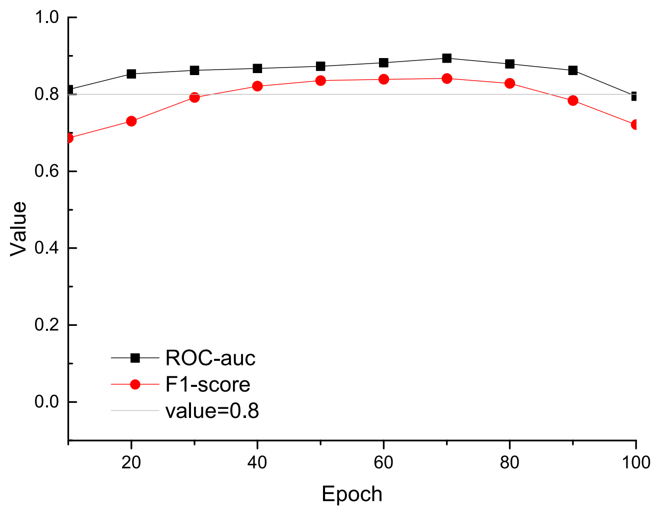

Fig. 2 The performance of F1-score and ROC-auc in the generated dataset. If the value is greater than 0.8, the model is working well.

correct	name_id	attri_id	sentence	tokens
1	(2:2)	(16:17)	In particular Co-30Ni-12Al-41a-12Cr (12Cr)…	[In,particular,Co-30Ni-12Al-41a-12Cr,(, 12Cr,),…]
1	(4:4)	(16:17)	In particular Co-30Ni-12Al-41a-12Cr (12Cr)…	[In,particular,Co-30Ni-12Al-41a-12Cr,(, 12Cr,),…]
1	(12:12)	(15:16)	Moreover, it is worth noting that the solvus temperature value…	[Moreover,it,is,worth,noting,that,the,solvus,temperature,value,…]
0	(12:12)	(26:27)	Moreover, it is worth noting that the solvus temperature value…	[Moreover,it,is,worth,noting,that,the,solvus,temperature,value,…]
0	(12:12)	(36:36)	Moreover, it is worth noting that the solvus temperature value…	[Moreover,it,is,worth,noting,that,the,solvus,temperature,value,…]

Table 3. An example of a manually corrected dataset. 'name_id' and 'attri_id' represent the position of the name and attribute value in the sentence, respectively.

$$P(Y|X) = \frac{P(X,\ Y)}{P(X)} \tag{1}$$

The discriminant model directly learning the conditional probability distribution $P(Y|X)$ from the data is set as a prediction model. Based on the characteriatics of the discriminant and generative models, the corpora produced by the generative model can help the discriminant model improve the coverage of the proposed method. The generative model needs to learn the joint probability distribution $P(X, Y)$, but for those corpora that cannot be covered by all the labeling functions, it is obviously impossible to get $P(X, Y)$. On the contrary, the discriminant model only needs the characteristics of X itself. $P(Y|X)$ can be calculated, so the discriminant model can cover the data points that the generative model cannot cover. In addition, compared to the probability graph model used in generative model training, discriminant models can be trained with more advanced and complex models, such as the ON-LSTM model we use, which can also improve the accuracy of the model.

ON-LSTM integrates the hierarchical structure into the LSTM through specific sorting of neurons, allowing the LSTM to learn the hierarchical structure information automatically. The training method is supervised learning, and the trained model can be used to process a large material corpus. ON-LSTM sorts the neurons inside the LSTM and integrates the hierarchical structure to express richer information[18]. In the original LSTM model, the updates between neurons are not related. For this reason, ON-LSTM adds two gates: the master forget gate $\widetilde{f}_t$ and the master input gate $\widetilde{i}_t$. The structure of ON-LSTM is shown in Fig. 3.

To demonstrate the superiority of the proposed method, our algorithm is compared with several classical algorithms on our proposed dataset. The comparison results are presented in Fig. 4. Among them, Snowball[22] is a general information extraction framework. Modified Snowball[23] is an improvement on the basis of snowball for the material field. Distance-based algorithm is the method proposed in our previous article[11]. LSTM refers to the results obtained after we use Snorke to automatically generate the corpus and then use the LSTM network

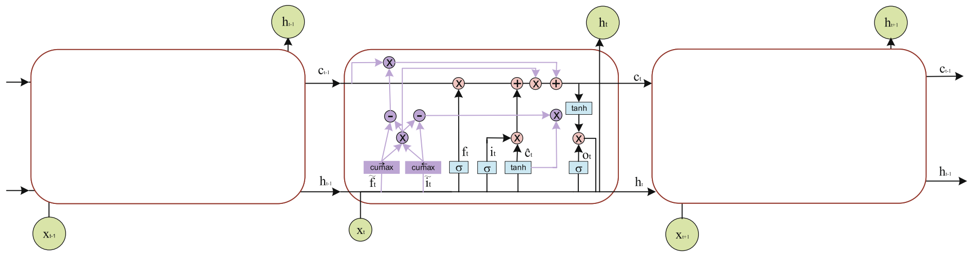

Fig. 3 The internal structure of ON-LSTM, where σ is the activation function *sigmoid*, f_t is forget gate, i_t is input gate and o_t is output gate.

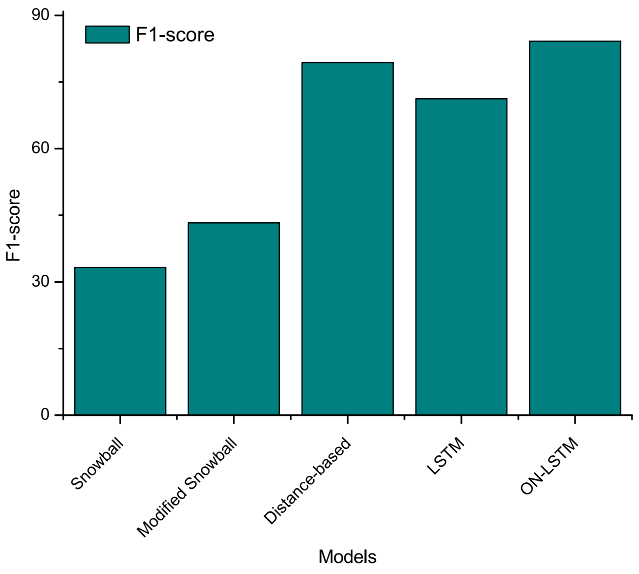

Fig. 4 Comparison results of ON-LSTM and the algorithms proposed in previous articles. ON-LSTM is our proposed method.

training. ON-LSTM is the result of training with ON-LSTM after the production corpus. It is obvious that our proposed method performs much better than the previous classical algorithms. The results show that ON-LSTM performs better than LSTM on the IE task. In other words, ordered neurons can express richer information in sentences and capture semantic information between words.

Generalize validation. The method that we proposed is a general framework for IE without corpora, which is universal in materials. To better illustrate this characteristic, we also extracted other physical properties from the material domain, including density, solidus temperatures of superalloys and hardness information of high entropy alloys. Table 4 shows that F1-score for density, γ' solvus temperature of superalloys and hardness information of high entropy alloys. Experimental results show that our proposed method for relation extraction through automatically generated corpus is versatile and can extract any properties in the material domain.

From Table 4, we can observe that F1-score has good performance in extracting density information from superalloys. We observe the characteristics of sentences containing density and find that these sentences are relatively monotonous compared to other attributes when describing density. This is why the F1-score of the density is relatively high. We summarize several typical sentence patterns as follows, where A represents the attribute and B represents the property value. A_i, B_i represents i-th A or B.

- "More significantly these Co-V-based superalloys have lower density (8.39–8.86 g/cm^3)". When writing a label function, we can describe it in the form of 'A(B)'.
- "The apparent density of the GTD222 and TiC/GTD222 composite powders were 4.56 g/cm^3 and 4.48 g/cm^3 respectively", which can be summarized as the pattern of 'A_1 and A_2 be verb B_1 and B_2'.
- "While the density of Nimonic 90.0 is 8.2 g/cm^3 the layer constituents Ni2Si, Ni5Si2, Cr2B and CrB have density of 7.2 g/cm^3 7.0 g/cm^3 6.6 g/cm^3 and 6.1 g/cm^3 respectively.". Labeling functions can be written as "A_1, A_2, A_3 and A_4 have density of B_1, B_2, B_3, B_4".

Physical properties of materials	number of sentences	Number of candidates	F1-score
density of superalloys	222	846	94.02
solidus temperatures of superalloys	128	348	89.27
hardness of high entropy alloys	155	472	85.81

Table 4. F1-score for density, γ' solvus temperature of superalloys and hardness information of high entropy alloys.

Discussion

Machine learning methods require large amounts of data for model training. Although machine learning methods have been widely used in many fields, they are still novel methods for extracting the required information in the field of materials. The extracted information can help researchers to determine which materials to use under what circumstances.

In this work, we use semi-supervised Snorkel to generate training sets in the field of materials. We take superalloys as an example, and verify the generality of the proposed method in the field of materials through a number of different material types. When generating the training set, since our dataset is highly unbalanced, even a trivial baseline that always outputs negative can obtain high accuracy. Therefore, we evaluated the dataset using the F1-score and ROC-auc rather than accuracy. In addition, we first investigate the potential integration between ON-LSTM and IE. Although we use more advanced methods to train the model, the results are not particularly satisfactory. This may be due to the small number of datasets and the imbalance of positive and negative samples. Although all our processes extract specific information in the field of materials, the proposed method can also be applied to other fields without datasets. Different labeling functions are written according to the requirements, and then the model is trained according to the generated dataset to increase the robustness of extraction. In all cases, the difficulty of writing labeling functions is related to the corpus's difficulty and the information extracted.

Using machine learning methods to extract information in the material field still faces many challenges. On the one hand, machine learning requires a large corpus, while the amount of data in the field of superalloys is small due to the difficulty of acquiring accurate and error-free datasets. In the future, we hope to obtain more articles about materials and obtain more sentences containing the physical properties to obtain larger and higher-quality data sets. On the other hand, we do not use a pre-trained model when extracting information due to the limited number of datasets. The pretraining model obtains models that are not related to specific tasks from large-scale data through self-supervised learning methods that can more effectively express the rich semantic features of words or sentences. In the future, it may be possible to introduce pretraining models such as BERT[24] and XLNet[25,26] in the information extraction stage to fully take advantage of the context information of sentences and accurately use vectors to express the meaning of words.

Methods

In this section, we describe the machine learning methods used in this work, namely the Snorkel method for generating datasets and the ON-LSTM method for training the IE models.

Snorkel. Snorkel is a model that uses weak supervision to generate datasets. It manually labels any outlier data and only requires users to write labeling functions[27]. Snorkel uses data programming[28,29] to obtain its output. The main purpose of Snorkel is to give an $\varphi \in \Phi$ and determine the possible discrete label $\tau \in T$, where Φ represents the candidate set and T represents the set $\{1, 0\}$. To achieve this goal, we need to write some labeling functions λ based on the specific dataset. For users, the written labeling functions are black-box functions, and they do not need to understand the operation of Snorkel on labeling functions. Upon input of candidate set Φ and labeling functions λ, Snorkel outputs labels T to which Φ belongs. Users can write labeling functions in the following ways:

1. Pattern-based: The method formulates some rules by observing the characteristics of sentence patterns. Omar *et al.* proposed the basic principles of observation to help users annotate data sets[30]. Sonal *et al.* used the rules of distribution similarity and word-to-word distance for labeling[31].
2. Distant supervision: Distant supervision refers to an existing knowledge base. Assuming that the knowledge base contains the information to be extracted, it is equivalent to automatically marking a part of the samples; for example, Raphael *et al.* used the information in the knowledge base to extract sentence-level repetitive relationships[32].
3. Weak classifiers: We call a classifier that is slightly better than a random prediction but not very accurate a weak classifier[33]. We can train weak classifiers on other datasets as labeling functions.

If the candidate set contains a data points and the users write b labeling functions, then the matrix $\Gamma \in T^{a*b}$ will be generated. Each labeling function may have coverage, overlaps, and conflicts for the same data point. Snorkel automatically solves the above problems internally and finally forms a single label for each data point. The most important component of Snorkel models, integrating multiple labeling functions, is called a generative model. Snorkel implements this component using the method of data programming. For details, please refer to[27–29].

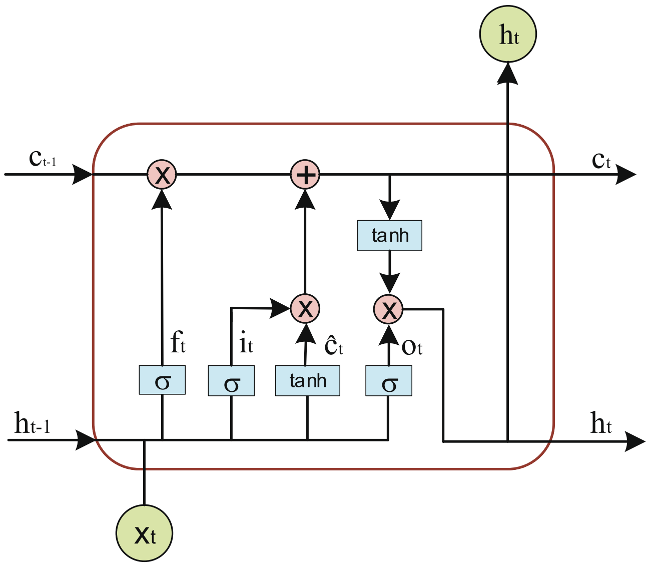

Fig. 5 The internal structure of LSTM. An LSTM cell consists of a memory cell c_t and three gates.

LSTM. After the acquired dataset is embedded by the plug-in that comes with TensorFlow[34], we use the ON-LSTM machine learning algorithm for relation extraction. ON-LSTM is a variant of LSTM. For a clear description of ON-LSTM, we illustrate its process step by step. In this section, we first understand the working principle of LSTM.

LSTM is a special type of recurrent neural network[35] (RNN) that can learn long-term dependencies. LSTM removes or adds information through its memory cell c_t. As shown in Fig. 5, there are three types of gates, namely forget gate f_t, input gate i_t and output gate o_t, in c_t[36]. The first step of LSTM is to decide what information we will discard from the cell state, which is done through the forget gate. The input is the hidden state h_{t-1} of the previous sequence and this sequence of data x_t. The output f_t of the forget gate represents the probability of forgetting the hidden cell state of the previous layer, and is expressed as follows.

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{2}$$

where σ is the activation function *sigmoid* and W_f and b_f are the linear correlation coefficient and bias, respectively. The value of f_t is between 0 and 1; here, 0 means that no information is allowed to pass, and 1 means that any information is allowed to pass.

The input gate determines what new information is stored in the cell state. It consists of two parts: the first part uses the *sigmoid* activation function, and its output is i_t. The second part uses the *tanh* activation function, and its output is $\hat{c}_t$. The results of the two are multiplied to update the cell state. W_i, W_c, b_i, and b_c are linearly related coefficients and biases.

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{3}$$

$$\hat{c}_t = tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{4}$$

Next, we need to update the state of the old cell and update c_{t-1} to c_t. We multiply the old state by f_t and discard the information that will certainly be discarded. For the addition of the product of the input gate i_t and $\hat{c}_t$, the formula is as follows.

$$c_t = f_t * c_{t-1} + i_t * \hat{c}_t \tag{5}$$

Finally, we need to determine the value to output. The formula for the calculation of o_t is as follows. Here, w_0, and b_o indicate the correlation coefficient and bias.

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_0) \tag{6}$$

The update of the hidden state h_t consists of two parts: the first part is o_t, and the second part is composed of c_t and activation functions *tanh*.

$$h_t = o_t * tanh(c_t) \tag{7}$$

ON-LSTM. The new cumax activation function was used according to the previously reported work. The neuron state controls the information to be stored and forgotten. By introducing such a gate mechanism, interdependent update rules between neurons are established so that neurons have an order and a hierarchy of differences.

The object of ON-LSTM thinking is natural language, and nature can usually express some hierarchical structure. In English sentences, letters can be considered the lowest level structure, and words and phrases have a higher level. The higher the level, the coarser the granularity and the greater the span of the sentence. In the ON-LSTM structure, high-level information may retain a considerable distance because the historical information directly copied by the high-level information may cause historical information to be repeated without changing. The low-level information may be updated at each step of input because the low-level information directly duplicates the input. The input is constantly changing, so that the hierarchical structure is embedded through information grading.

The forget gate f_t, input gate i_t, output gate o_t and $\hat{c}_t$ of ON-LSTM given by the same formulas as ct and LSTM, but the update mechanism from $\hat{c}_t$ to c_t is different. The following is the updated formula of the entire ON-LSTM:

$$h_t = o_t * tanh(c_t) \tag{8}$$

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{9}$$

$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{10}$$

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_0) \tag{11}$$

$$\hat{c}_t = tanh(W_c * [h_{t-1}, x_t] + b_c) \tag{12}$$

$$\widetilde{f}_t = cumax(W_{\hat{f}} * [h_{t-1}, x_t] + b_{\hat{f}}) \tag{13}$$

$$\widetilde{i}_t = 1 - cumax(W_{\hat{i}} * [h_{t-1}, x_t] + b_{\hat{i}}) \tag{14}$$

$$w_t = \widetilde{f}_t * \widetilde{i}_t \tag{15}$$

$$c_t = \widetilde{f}_t * c_{t-1} + \widetilde{i}_t * \hat{c}_t \tag{16}$$

$$h_t = o_t * tanh(c_t) \tag{17}$$

The value of the *cumax* activation function decreases monotonically from 1 to 0. Within a certain range, its value tends to 0, indicating that the previous information has been forgotten; if its value tends to 1, the new input content becomes increasingly important. When training the model, we set the dropout as 0.4, the learning rate is 0.1, and the dimension of the word vector is 64.

Data availability

Our initial data and extracted data are available at https://github.com/MGEdata/snorkel.

Code availability

The code is available at https://github.com/MGEdata/auto-generate-corpus. When researchers extract their own corpus, they only need to write labeling functions that meet the characteristics of their own corpus in the framework we write, which is very simple to use.

Received: 9 March 2022; Accepted: 28 June 2022;
Published online: 13 July 2022

References

1. Galassi, A., Lippi, M. & Torroni, P. Attention in natural language processing. *IEEE Transactions on Neural Networks Learn. Syst.* **15**, 3709–3721 (2020).
2. Mooney, R. J. & Bunescu, R. C. Mining knowledge from text using information extraction. *Acm Sigkdd Explor. Newsl.* **7**, 3–10 (2005).
3. Rickman, J. M., Lookman, T. & Kalinin, S. V. Materials informatics: From the atomic-level to the continuum. *Acta Materialia* **168**, 473–510 (2019).
4. Wen, C. *et al.* Machine learning assisted design of high entropy alloys with desired property. *Acta Materialia* **170**, 109–117 (2019).
5. Xue, D. *et al.* Accelerated search for materials with targeted properties by adaptive design. *Nat. communications* **7**, 1–9 (2016).
6. Tshitoyan, V. *et al.* Unsupervised word embeddings capture latent knowledge from materials science literature. *Nat.* **571**, 95–98 (2019).
7. Swain, M. C. & Cole, J. M. Chemdataextractor: a toolkit for automated extraction of chemical information from the scientific literature. *J. chemical information modeling* **56**, 1894–1904 (2016).
8. Krallinger, M., Rabal, O., Lourenco, A., Oyarzabal, J. & Valencia, A. Information retrieval and text mining technologies for chemistry. *Chem. reviews* **117**, 7673–7761 (2017).

9. Kim, E. *et al*. Inorganic materials synthesis planning with literature-trained neural networks. *J. chemical information modeling* **60**, 1194–1201 (2020).
10. Kim, E., Huang, K., Jegelka, S. & Olivetti, E. Virtual screening of inorganic materials synthesis parameters with deep learning. *npj Comput. Mater.* **3**, 1–9 (2017).
11. Wang, W. *et al*. Automated pipeline for superalloy data by text mining. *npj Comput. Mater.* **8**, 1–12 (2022).
12. Sahu, S. K., Anand, A., Oruganty, K. & Gattu, M. Relation extraction from clinical texts using domain invariant convolutional neural network. In *BioNLP@ACL* (2016).
13. Lv, X., Guan, Y., Yang, J. & Wu, J. Clinical relation extraction with deep learning. *Int. J. Hybrid Inf. Technol.* **9**, 237–248 (2016).
14. Chalapathy, R., Borzeshi, E. Z. & Piccardi, M. Bidirectional lstm-crf for clinical concept extraction. *arXiv preprint arXiv:1611.08373* (2016).
15. Javeed, A. An lstm model for extracting hierarchical relations between words for better topic modeling. *J. Physics: Conf. Ser.* **1780**, 012019 (2021).
16. Ratner, A. *et al*. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, vol. 11, 269 (NIH Public Access, 2017).
17. Ratner, A., Bach, S. H., Ehrenberg, H., Fries, J. & Re, C. Snorkel: rapid training data creation with weak supervision. *The VLDB J.* **11**, 269–282 (2017).
18. Shen, Y., Tan, S., Sordoni, A. & Courville, A. C. Ordered neurons: Integrating tree structures into recurrent neural networks. *ArXiv abs/1810.09536* (2019).
19. Gao, T., Han, X., Xie, R., Liu, Z. & Sun, M. Neural snowball for few-shot relation learning. *Proc. AAAI Conf. on Artif. Intell.* **34**, 7772–7779 (2020).
20. Mallory, E. K. *et al*. Extracting chemical reactions from text using snorkel. *BMC Bioinforma*. **21** (2020).
21. Fawcett, T. An introduction to roc analysis. *Pattern recognition letters* **27**, 861–874 (2006).
22. Agichtein, E. & Gravano, L. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries*, 85–94 (2000).
23. Court, C. J. & Cole, J. M. Auto-generated materials database of curie and neel temperatures via semi-supervised relationship extraction. *Sci. data* **5**, 1–12 (2018).
24. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
25. Yang, Z. *et al*. Xlnet: Generalized autoregressive pretraining for language understanding. *Adv. neural information processing systems* **32** (2019).
26. Yan, R., Jiang, X. & Dang, D. Named entity recognition by using xlnet-bilstm-crf. *Neural Process. Lett.* **53**, 1–18 (2021).
27. Ratner, A. *et al*. Snorkel: Rapid training data creation with weak supervision. *The VLDB J.* **29**, 709–730 (2020).
28. Bach, S. H., He, B. D., Ratner, A. J. & Re, C. Learning the structure of generative models without labeled data. *Proc. machine learning research* **70**, 273–82 (2017).
29. Ratner, A., De, S. C., Wu, S., Selsam, D. & Re, C. Data programming: Creating large training sets, quickly. *Adv. neural information processing systems* **29**, 3567 (2016).
30. Zaidan, O. & Eisner, J. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 conference on Empirical methods in natural language processing*, 31–40 (2008).
31. Gupta, S. & Manning, C. D. Improved pattern learning for bootstrapped entity extraction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, 98–108 (2014).
32. Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L. & Weld, D. S. Knowledge-based weak supervision for information extraction of overlapping relations. *In ACL* (2011).
33. Shatalova, O. V., Mednikov, D. A., Protasova, Z. U. & Stadnichenko, N. S. Prediction of the risk of cardiovascular complications with a segmented space of risk factors and synergy channels. *J. Physics: Conf. Ser.* **1679**, 032042 (5pp) (2020).
34. Abadi, M. *et al*. {TensorFlow}: A system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, 265–283 (2016).
35. Zaremba, W., Sutskever, I. & Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* (2014).
36. Shi, X. *et al*. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *arXiv preprint arXiv:1506.04214* (2015).

Acknowledgements
D.D. and Y.S. are the corresponding authors of this paper. This research is supported by The National Key Research and Development Program of China under Grant No.2020YFC1523303;The Key Research and Development Program of Qinghai Province under Grant No.2020-SF-140;the National Natural Science Foundation of China under Grant No. 61672102, No.61073034, No. 61370064 and No. 60940032; the National Social Science Foundation of China under Grant No.BCA150050; the Program for New Century Excellent Talents in the University of Ministry of Education of China under Grant No. NCET-10-0239; the Open Project Sponsor of Beijing Key Laboratory of Intelligent Communication Software and Multimedia under Grant No.ITSM201493;and the Science Foundation of Ministry of Education of China and China Mobile Communicaions Corporation under Grant No. MCM20130371. Special thanks to my boyfriend Ye Tao, who guided me to submit the paper and encouraged me to revise the manuscript over and over again.

Author contributions
The original idea was put forward by R.Y. and D.D. and discussed with X.J. and Y.S. The original data is provided by X.J., W.W. and Y.S. All authors participated in the discussion, analysis, writing and reading of the paper. D.D. and Y.S. managed and guided the project.

Competing interests
The authors declare no competing interests.

Additional information
Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41597-022-01492-2.

Correspondence and requests for materials should be addressed to D.D. or Y.S.