A Corpus and Method for Chinese Named Entity Recognition in Manufacturing

Ruiting Li[1,2], Peiyan Wang[1,2]†
Libang Wang[1,2], Danqingxin Yang[1,2], Dongfeng Cai[1,2]
[1] School of Computer Science, Shenyang Aerospace University, Shenyang, China
[2]Liaoning Professional Technology Innovation Center on Knowledge Engineering
and Human-Computer Interaction, Shenyang, China
liruiting@stu.sau.edu.cn, wangpy@sau.edu.cn
{wanglibang, yangdanqingxin}@stu.sau.edu.cn, caidf@vip.163.com

Abstract

Manufacturing specifications are documents entailing different techniques, processes, and components involved in manufacturing. There is a growing demand for named entity recognition (NER) resources and techniques for manufacturing-specific named entities, with the development of smart manufacturing. In this paper, we introduce a corpus of Chinese manufacturing specifications, named *MS-NERC*, including 4,424 sentences and 16,383 entities. We also propose an entity recognizer named Trainable State Transducer (*TST*), which is initialized with a finite state transducer describing the morphological patterns of entities. It can directly recognize entities based on prior morphological knowledge without training. Experimental results show that *TST* achieves an overall 82.05% F1 score for morphological-specific entities in zero-shot. *TST* can be improved through training, the result of which outperforms neural methods in few-shot and rich-resource. We believe that our corpus and model will be valuable resources for NER research not only in manufacturing but also in other low-resource domains.

Keywords: corpus, named entity recognition, information extraction

1. Introduction

The application of natural language processing (NLP) in manufacturing has propelled the advancement of smart manufacturing, wherein textual data constitutes a pivotal component. Manufacturing specifications are documents entailing different techniques, processes, and components involved in manufacturing. The accurate recognition of named entities in the manufacturing specifications lay the foundation for downstream tasks such as knowledge graph construction (Buchgeher et al., 2021) and relation extraction (Wang et al., 2020). However, unsimilar to domains such as financial (Wu et al., 2020), medical (Tian et al., 2022), and computer code (Tabassum et al., 2020), there is a lack of resources and techniques for recognizing manufacturing-specific named entities (e.g., PART or PART_ID) in manufacturing.

There is no publicly available corpus for Chinese NER in the manufacturing specifications. Corpus annotation demands annotators with a strong background in domain-specific knowledge. The high cost of manual annotation leads to limited corpus. To achieve good performance on NER, conventional neural methods normally rely on a large amount of labeled training data, which is unavailable in manufacturing. Our approach to address the issue is to regard entities' morphological patterns as domain-specific knowledge, which can be used for modeling an entity recognizer.

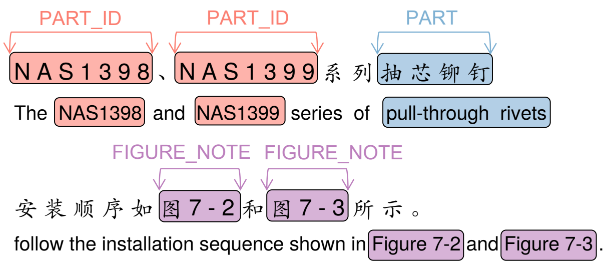
PART_ID
PART_ID
PART
NAS1398、NAS1399系列抽芯铆钉
The NAS1398 and NAS1399 series of pull-through rivets
FIGURE_NOTE
FIGURE_NOTE
安装顺序如图7-2和图7-3所示。
follow the installation sequence shown in Figure 7-2 and Figure 7-3.

Figure 1: Examples of the sentence with named entities and translations in the manufacturing specifications.

In this paper, we introduce *MS-NERC*, a Chinese Manufacturing Specifications Named Entity Recognition Corpus. We identify morphological patterns in manufacturing-specific named entities (PART_ID and FIGURE_NBOTE) as shown in Figure 1. To recognize these entities with very limited training data, we propose *TST*, a Trainable State Transducer for modeling morphological patterns of named entities as a trainable entity recognizer. We test our *TST* model on the *MS-NERC* corpus. Results show the advantage of *TST* in zero-shot, few-shot, and rich-resource. To summarize, our key contributions are the following:

† Corresponding author.

- Annotation guidelines for NER corpus in manufacturing, including annotation instructions and definitions for 16 categories such as PART, PART_ID, FILE, and FIGURE_NOTE. Our guidelines provide a reference for the annotation work in the future.

- A Chinese NER corpus named *MS-NERC* with 4,424 sentences and 16,383 entities. The *MS-NERC* is manually annotated by three experts in manufacturing.

- An entity recognizer named Trainable States Transducer (*TST*) incorporating prior morphological knowledge. The model is initialized with a finite state transducer describing the morphological patterns of entities. It is capable of recognizing related entities based on prior morphological knowledge without training. Besides, *TST* can be improved through supervised learning.

Owing to prior morphological knowledge, *TST* can directly recognize related entities without training. In zero-shot, *TST* attains an F1 score of 82.05%. In both few-shot and rich-resource, *TST* consistently outperforms neural methods as shown in §5.3.

2. Related Work

2.1. NER in Specific Domains

With the application of NER in specific domains, there is a growing interest in both corpus and techniques.

Corpus Annotation The construction of corpus is gradually attracting attention such as news (Mbuvha et al., 2023), social media (Jiang et al., 2022), and computer code (Tabassum et al., 2020). Meanwhile, little attention is paid to manufacturing documents (Chen et al., 2021; Zhang et al., 2019; Jia et al., 2022). For example, Chen et al. (2021) define 7 categories of entities and annotate 1,139 sentences for assembly manufacturing documents. However, there is no corpus for manufacturing specifications. In our study, we annotate manufacturing specifications offering broader coverage than the existing corpus in manufacturing. We define 16 fine-grained categories and annotate a total of 4,424 sentences, significantly surpassing other corpus.

Models Enhanced by Domain Knowledge Adding domain knowledge to neural networks can be effective with insufficient data. One useful way is to apply dictionaries and rules to refine the results obtained from neural networks in chemistry (Ma et al., 2018), military (Feng et al., 2015) and Uyghur language (Zhu, 2019). However, it is invalid when the entities cannot be described by dictionaries and rules. Jia et al. (2022) use domain dictionaries and entity rules for pre-recognizing entities and use pre-recognization features to guide the model training.

2.2. Finite State Transducer in NLP

The majority of previous work combines a finite state transducer (FST) with neural networks. For morphological generation tasks, Rastogi et al. (2016) present a hybrid FST-LSTM architecture. It combines FST and long short-term memory (LSTM), while weights of FST depend on different contexts and LSTM is used to extract the features that determine these weights. Lin et al. (2019) introduce an architecture named NFSTs, which is conditional probability distributions over pairs of strings. In this architecture, FST is used to constrain text generation. Other methods model regular expressions for downstream tasks. FARNN (Jiang et al., 2020) and FSTRNN (Jiang et al., 2021) are designed for intent detection and slot filling. The models above can be converted from sentence-level regular expressions and are interpretable after training. In this way, the advantage of symbolic rules and the neural network can be a combination. Our method is different from these works. In contrast to existing studies, our *TST* model operates at the character level and elucidates the mapping between Chinese characters and entity labels. Especially, we are the first to model FST as a trainable entity recognizer.

3. The *MS-NERC* Corpus

3.1. Text Collection

We collect technical specification documents of an aircraft for annotating. The specifications span four subdomains: assembly manufacturing, composite material processing, mechanical processing, and computer numerical control processing. Each document comprises several chapters, such as material control, equipment control, technical control, procedure control, maintenance control, and quality requirements. From these documents, we extract statements containing manufacturing parameters and manufacturing conditions. We filter out information related to specific products and enterprises due to privacy and legal issues. Ultimately, we obtain a raw dataset of 4,424 sentences.

3.2. Annotation Guidelines

To define the category set in the manufacturing specifications, we refer to the Fundamental Terminology of Mechanical Manufacturing (Hongyu et al., 2008). We define a set consisting of 16 categories after consulting two experts in manufacturing. We present entity labels, entity definitions, and examples as follows.

- **ACCESSORY** marks substances that play an auxiliary role in the production (e.g., 发泡胶 / polystyrene foam).
- **ACCESSORY_ID** marks an identification number assigned to the accessory (e.g., RIX-ZC-106).
- **ATTRIBUTE** marks a physical characteristic in manufacturing (e.g., 升温速率 / heating rate).
- **ATTRIBUTE_VA** marks the specific value of a physical characteristic in manufacturing (e.g., 1.5mm).
- **FIGURE_NOTE** marks an identification number of the referenced figure (e.g., 图6-20 / Figure 6-20).
- **FILE** marks an identification number of cited documents in the manufacturing specifications (e.g., CAR3001).
- **HOLE** marks a hole for part machining, assembly, inspection, and mounting (e.g., 铆钉孔 / rivet hole).
- **MATERIAL** marks a substance used to manufacture parts or components, including metallic and non-metallic materials (e.g., 铝合金 / Aluminum alloy).
- **OPERATION** marks an activity such as assembly, inspection, and handover, following certain procedures and technical requirements (e.g., 胶接 / bonding).
- **PART** marks a basic component unit in manufacturing, including the combination of parts (e.g., 铆钉 / rivet).
- **PART_AR** marks the location of a part for machining or assembly operations (e.g., 蜂窝板表面 / the surface of honeycomb panel).
- **PART_ID** marks an identification number assigned to the part (e.g., NAS1252).
- **PART_NU** marks the number of parts and tools (e.g., 两个 / two).
- **REDUNDANT** marks trimmings and scraps generated in manufacturing (e.g., 毛刺 / burr).
- **TABLE_NOTE** marks an identification number of the referenced table (e.g., 表7-1 / Table 7-1).
- **TOOL** marks a machining tool used in manufacturing (e.g., 刮板 / drawing strickle).

In addition, our annotation adheres to three specific instructions. First, the entities must be specific rather than generalized (e.g., '铆钉' /rivet instead of '零件'/part). Second, the entities should not be accompanied by conjunctions and punctuation marks indicating juxtaposition, except in the case of notes in parentheses (e.g., '1.02mm(0.040in.)'). Lastly, when these specific instructions are met, entities are annotated based on their maximum span without nesting.

3.3. Annotation Process

Our corpus is annotated by three experts in manufacturing. Given the initial inconsistencies in the experts' interpretations of the annotation guidelines, a series of iterative discussions take place during the pre-annotation phase to refine the guidelines. Formal annotation commences once a Cohen's Kappa (Cohen, 1960) score exceeding 0.6 is achieved. During the formal annotation stage, the texts are divided into three groups. Each group is assigned to a different annotator, with a 15% overlap for duplicate assessment. The inter-annotator agreement is subsequently calculated based on this 15% overlap, resulting in a Cohen's Kappa score of 0.68.

3.4. *MS-NERC* Statistics

The corpus comprises a total of 4,424 sentences and 16,383 entities. Table 1 shows the statistics of entity numbers, character-based max lengths, min lengths, and mean lengths. Notably, entity lengths have considerable variation due to the annotation based on maximum span. For example, there are instances of multi-unit attribute values, such as the ATTRIBUTE_VA of 50 max length. Additionally, partially abbreviated entities with lengths ranging from 1 to 3 characters are observed (e.g., '钉' is the abbreviation for '铆钉').

Category	Number	Max	Min	Mean
ACCESSORY	1,861	12	1	3.31
ACCESSORY_ID	876	17	3	7.87
ATTRIBUTE	2,479	19	1	3.54
ATTRIBUTE_VA	1,357	50	1	10.85
FIGURE_NOTE	396	6	3	4.21
FILE	556	12	3	6.82
HOLE	418	12	1	2.14
MATERIAL	481	13	1	3.27
OPERATION	1,933	9	1	2.63
PART	2,407	13	1	3.27
PART_AR	1,396	14	1	3.85
PART_ID	202	15	2	8.75
PART_NU	131	4	1	1.89
REDUNDANT	164	6	1	2.27
TABLE_NOTE	296	6	3	4.05
TOOL	1,430	18	1	3.65

Table 1: Entity statistics in *MS-NERC*.

There are named entities for describing the identification numbers of manufacturing elements (FILE, ACCESSORY_ID, TABLE_NOTE, FIGURE_NOTE, PART_ID) and manufacturing condition parameters (ATTRIBUTE_VA). These named entities conform to morphological patterns, which can be regarded as prior morphological knowledge to inform NER tasks and inspire our model in §4.

Category	Example	REs
ACCESSORY_ID	RQI5275	RQI(\d){4}
ATTRIBUTE_VA	-28.6m	-(\d+).(\d+)m
FIGURE_NOTE	图6-20	图(\d+)-(\d+)
FILE	RQX3001	RQX(\d){4}
PART_ID	NAS1252	NAS(\d+)
TABLE_NOTE	表7-1	表(\d+)-(\d+)

Table 2: Examples of entities and regular expressions.

4. Trainable States Transducer

4.1. Regular Expressions and Finite State Transducer

We employ regular expressions (REs) to summarise the morphological patterns of entities, with examples of entities and their corresponding REs provided in Table 2. However, regular expressions can not define the mapping from characters to entity labels, whereas a FST can.

Sakuma et al. (2012) demonstrate rigorously that a regular expression with capturing groups is equivalent to a FST. We represent each FST textually, utilizing the 'BIOE' tag scheme for the output. In this scheme, each '<:>' separates the character (left) and the output label (right). We employ '$' as a wildcard, and 'OO' denotes a temporary label. The use of '$' and 'OO' serves to provide conditions that the unmatched portion of a sentence can be allowed to match with other FSTs. We show the symbols used in Table 3, which contribute to a concise and precise textual representation of FSTs. Each symbol operates solely on its preceding symbol or sub-expression. To illustrate this process, we depict the regular expression and textual FST of an ATTRIBUTE_VA entity in Tabel 4.

Symbol	Meaning
^	number (0-9)
$*$	zero or more occurrence
|	or operator
?	zero or one occurrence
÷	capital letter (A-Z)
+	one or more occurrence

Table 3: Symbols for FST.

4.2. Parameter Tensors for *TST*

We illustrate the parameter initialization of *TST* in Figure 2. The parameters of *TST* are initialized with a FST, which can be formally defined as a 6-tuple: $\mathcal{F} = \langle \mathcal{S}, \mathcal{I}, \mathcal{O}, \delta, \mathcal{B}, \mathcal{E} \rangle$. $\mathcal{S}$ is a finite set of states, $\mathcal{I}$ is a finite set of input characters, $\mathcal{O}$ is a finite set of output labels,and δ is a function describing all transitions. $\mathcal{B}$ is a finite set of start states and $\mathcal{E}$ is a finite set of final states.

$\mathcal{B}$ and $\mathcal{E}$ in FST can be converted to the start states vector *s* and final states vector *m* by one-hot encoding. Specifically, δ describes the transition in the FST: $\delta(\gamma, x_t) = \upsilon, l_t$. It means that $\mathcal{F}$ accepts a character x_t of input, then transmits from state γ to state υ, and outputs label l_t. To address the high time complexity and space complexity of the 4-order tensor, we decompose it into W_i and W_o, following the work of Jiang et al. (2021). The decomposition can be computed by Eq.(1):

$$\delta[x_t, l_t, \gamma, \upsilon] = W_i[x_t, \gamma, \upsilon] \times W_o[l_t, \gamma, \upsilon] \quad (1)$$

To sum up, *TST* can be formally defined as a 7-tuple: $\mathcal{N} = \langle \mathcal{S}, \mathcal{I}, \mathcal{O}, W_i, W_o, \boldsymbol{s}, \boldsymbol{m} \rangle$.

Algorithm 1 Inference in *TST*

```
Input: $x = (x_1, x_2, ..., x_n)$,
$\mathcal{N} = \langle \mathcal{S}, \mathcal{I}, \mathcal{O}, W_i, W_o, s, m \rangle$.
Output: the label scores $f_t$ of $x_t$.
Step 1: Let $\odot$ denote hadamard product,$\bigotimes$ denote
  outer product.
Let $\alpha_0 = s^\mathsf{T}, \beta_n = m^\mathsf{T}$.
Step 2: calculate forward score $\alpha_t$
for $t = 1 \rightarrow n$ do
    $\alpha_t = W_i[x_t] \cdot \alpha_{t-1}$
end
Step 3: calculate backward score $\beta_t$
for $t = n \rightarrow 1$ do
    $\beta_t = W_i^\mathsf{T}[x_t] \cdot \beta_{t+1}$
end
Step 4: calculate bidirectional score $bi\_scores$
for $t = 1 \rightarrow n$ do
    $bi\_scores = \alpha_t \bigotimes \beta_t$
end
$F_t = \sum_{l_j \in \mathcal{O}} (bi\_scores \odot W_i[x_t]) \odot W_o[l_j]$
$f_t = \sum_{k=1}^{|S|} F_t[:, k]$
return $f_t$
```

4.3. *TST* Inference

Given a sentence $x = (x_1, x_2, ..., x_n)$, the *TST* algorithm aims to find the output labels $l = (l_1, l_2, ..., l_n)$. The score of the label l_t is the sum of the weights of all acceptance paths in *TST* that match x_t and l_t. Casacuberta and de la Higuera (2000) demonstrate that finding the highest scoring output for a given sentence is NP-hard. Our method uses an approximate inference to independently determine the highest-scoring output label at the current position while ignoring the labels at other positions. We show the algorithm of computing label scores simultaneously in Algorithm 1. *TST* modifies its own parameters through supervised learning. However, the entities described by prior morphological knowledge in *TST* may be missing from the training set

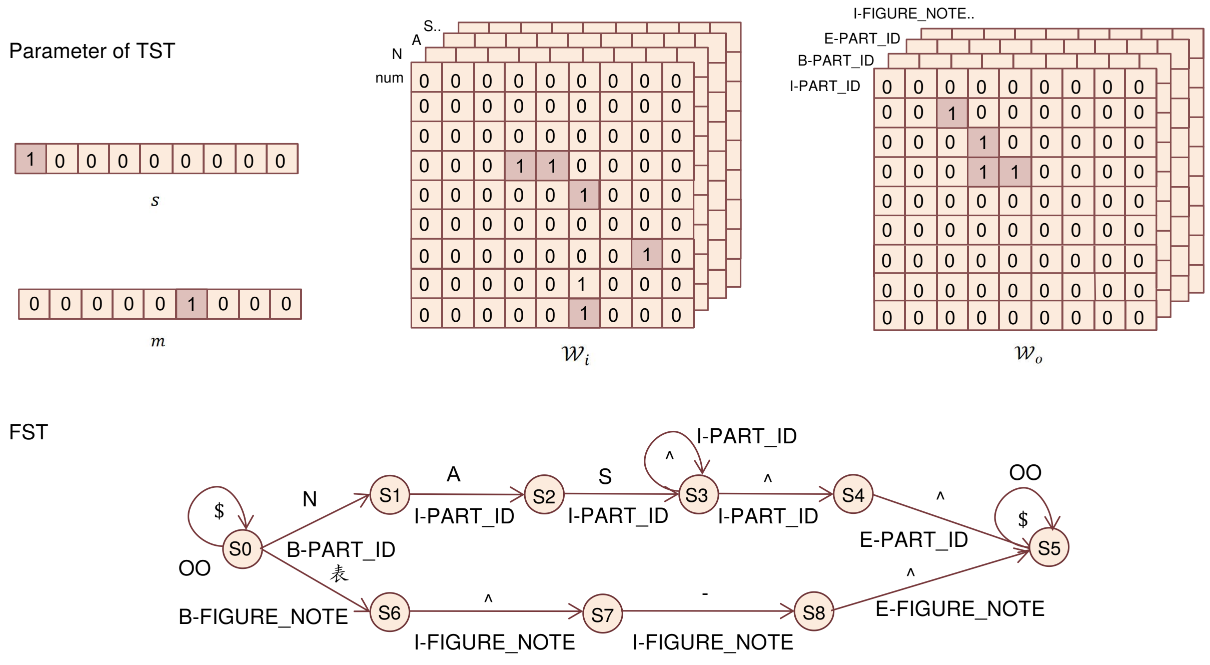

Figure 2: A parameter initialization example of *TST*.

Entitiy	3.2m∼4.9m
RE	(\d+).(\d+)m∼(\d+).(\d+)m
FST	$<:>OO * ∧<:>B-ATTRIBUTE_VA ∧<:>I-ATTRIBUTE_VA * .<:>I-ATTRIBUTE_VA ∧<:>I-ATTRIBUTE_VA + m<:>I-ATTRIBUTE_VA ∼<:>I-ATTRIBUTE_VA ∧<:>I-ATTRIBUTE_VA + .<:>I-ATTRIBUTE_VA ∧<:>I-ATTRIBUTE_VA + m<:>E-ATTRIBUTE_VA $<:>OO *

Table 4: RE and FST of an ATTRIBUTE_VA entity.

in the few-shot scenario. To make the prior morphological knowledge in *TST* stable during training, we use the combined outcome of trained f_t and f_t' without training. Since a subset of the *TST* captured group may also be captured, we use a priority layer to resolve conflicts with the aim of prioritizing 'I-' over 'B-' and 'E-'. The probability vector P_t is calculated to the Eq.(2), where W_p is the weight and b_p is the bias in the priority layer.

$$P_t = priority\left(W_p\left(f_t + f_t'\right) + b_p\right) \tag{2}$$

Before decoding from P_t, OO is still retained in the pending output label, which corresponds to a probability p_k' of undeterminedness. In Eq.(3), if the probability of all labels is less than p_k', we assign the corresponding label as O. Otherwise, we output the label with the highest probability.

$$\begin{gathered} P_t^{'} = \left(\max\left(p_1', p_k'\right), p_2', p_3', ..., p_{k-1}'\right) \\ l_t = \underset{1\leq j\leq k-1}{argmax}\, P_t^{'}\left(j\right) \end{gathered} \tag{3}$$

We use Eq.(4) as a loss function to measure the discrepancy between the prediction and actual labels, where $y^{(t)}$ is one-hot embedding of the target label at time t, while $p_j^{'(t)}$ is the probability of label j for x_t.

$$loss = \sum_{t=1}^{n}\sum_{j=1}^{|\mathcal{O}|} y^{(t)} log\left(p_j^{'(t)}\right) \tag{4}$$

5. Experiments

5.1. Dataset and Regular Expressions

We train and evaluate *TST* on the MS-NERC annotated in §3. We first adopt a standard setting by randomly splitting 70% sentences as the training set, 10% as the development set, and 20% as the testing set. To demonstrate that the prior morphological knowledge equips the initial *TST* with a higher start point in few-shot, we further sample the training data. At the entity level, n-shot means there are n entities of each entity category. At the sentence level, n%-sen means there are n% sentences extracted from the training set. Table 5 shows the statistics of the training sets.

Training Set	Sentence	Entity
20-shot	92	320
50-shot	246	800
3%-sen	92	322
6%-sen	186	667
10%-sen	309	1,129
30%-sen	928	3,342
100%-sen	3,094	11,195

Table 5: Statistics of the training sets.

We ask annotators to write regular expressions and evaluate the efficiency with 100%-sen training set in Figure 3. We only retain the regular expressions describing more than three entities, resulting in a total of 22 regular expressions.

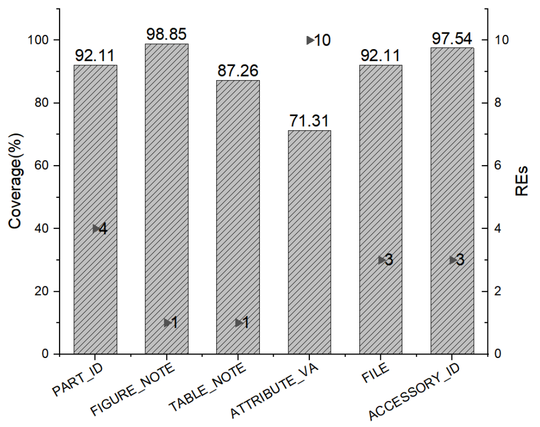

Figure 3: The coverage rate and statistics of regular expressions with 100%-sen training set.

5.2. Baslines

We train *TST* with supervised training sets separated from MS-NERC and compare *TST* with the following baselines. We employ the same decoder for all models.

- Prompt-Slot-Tagging (Hou et al., 2022) reversely predict slot values based on provided slot types. This approach incorporates training by considering the relationships between different slot types.
- Template-based BART (Cui et al., 2021) is a template-based method for exploiting the few-shot learning potential of generative pre-trained language models to sequence labeling.
- NNShot (Yang and Katiyar, 2020) is a method based on nearest neighbor learning and structured inference. This approach uses a supervised NER model trained on the source domain, as a feature extractor.
- PER (Jia et al., 2022) is a specialized NER method in manufacturing. This method utilizes rules and dictionaries for pre-recognition and employs the pre-recognition results to guide the training of the neural network.
- BILSTM (Siami-Namini et al., 2019) is one of the prominent deep learning models employed for addressing sequence-related tasks.
- TENER (Yan et al., 2019) utilizes the Transformer architecture to model information for NER tasks.

Our baselines are specifically designed for different data conditions: three (Template-based BART, Prompt Slot Tagging and NNShot) for few-shot and two (BILSTM and TENER) for rich-resources. PER is a specialized NER method in manufacturing, so we conducted comparisons in both few-shot and rich-resources.

5.3. Experimental Results

Zero-shot We compare the performance of the initial *TST* and REs in Table 6. Table 6 presents F1 and micro-average F1 scores calculated for six entity categories, assessing the recognizing ability of morphological-specific entities without training. Our method outperforms the regular expressions by +3.29% in micro-average F1. This can be attributed to the priority layer of *TST*, which effectively resolves conflicts in output labels and selects the optimal choice.

Type	*TST*	REs
ACCESSORY_ID	96.81	97.08
ATTRIBUTE_VA	63.41	56.63
FIGURE_NOTE	96.97	98.20
FILE	92.98	88.98
PART_ID	58.06	75.51
TABLE_NOTE	97.67	97.67
micro-average	82.05	78.76

Table 6: F1 scores (%) for the initial *TST* and REs in zero-shot.

Rich-resource We show the results of 100%-sen training set in Table 7. The micro-average F1 scores listed are calculated based on all 16 entity categories.*TST* are obviously competitive with

	P	R	F1
TST	**65.96**	61.27	**63.53**
PER	58.07	**67.2**	62.3
BILSTM	55.28	64.67	59.61
TENER	53.37	63.98	58.19

Table 7: Micro-average F1 scores (%) for *TST* and baselines in rich-resource.

	20-shot	50-shot	3%-sen	6%-sen
init-*TST*		29.51		
TST	26.7	**31.24**	**25.77**	**36.7**
PER	0.61	4.8	0.87	6.81
Template-based BART	12.52	15.68	13.87	17.54
Prompt Slot Tagging	16.38	17.08	17.44	23.2
NNShot	**27.2**	30.5	25.25	29.37

Table 8: Micro-average F1 scores (%) for *TST* and baselines in few-shot.

the baselines, which shows the equivalent learning ability. *TST* outperforms PER by +1.23% in F1, demonstrating its effectiveness with full training data. *TST* achieves a higher precision and a lower recall, adopting a conservative approach in comparison to neural networks. This characteristic renders *TST* well-suited for NER tasks in manufacturing that prioritize high precision and low error rates.

Few-shot Table 8 presents the micro-average F1 scores in few-shot, with init-*TST* indicating the initial TST's F1 score across all 16 entity categories in zero-shot. The results demonstrate that our model maintains a lead over the baselines in few-shot and outperforms the init-*TST* after training. Our model's strength lies in its utilization of prior morphological knowledge, distinguishing it from the neural baselines. PER performs poorly in few-shot because this method relies on domain knowledge to guide the neural network training. When there is insufficient training data, domain knowledge can not be effectively leveraged. *TST* underperforms the init-*TST* and NNshot in 20-shot. This discrepancy may be due to the differences between the development and testing sets. Through training, we select and evaluate the best performance model based on the development set. The 6%-sen has fewer entities than the 50-shot but gets a better result. This is because that the 6%-sen training set and the testing set are extracted based on the proportions of the sentences and entities. In 6%-sen, some entities have counts exceeding 50, and these entities also constitute a significant portion of the testing set. *TST* demonstrates more comprehensive learning for such entities, contributing to more accurate recognition during testing and yielding a higher micro-average F1 score.

6. Conclusion

In this paper, we investigate the task of named entity recognition in manufacturing. We define 16 categories and develop a Chinese NER corpus named *MS-NERC* with 4,424 sentences and 16,383 entities in manufacturing. We demonstrate that some entities in *MS-NERC* conform to morphological patterns. We propose an entity recognizer named *TST*, which can be initialized with a FST describing the morphological patterns of entities. Our method efficiently exploits prior morphological knowledge, enabling *TST* to directly recognize related entities, free from the constraints of the training set. Experiments show that *TST* demonstrates competitive performance in zero-shot, few-shot, and rich-resource. To the best of our knowledge, we are the first to model prior morphological knowledge of the entities as a trainable entity recognizer.

7. Acknowledgements

This research was supported by Applied Basic Research Program of Liaoning Province (2022JH2/101300248), Research Programs of China National Committee for Terminology in Science and Technology (YB2022015), and National Natural Science Foundation of China (No.U1908216).

8. Bibliographical References

Georg Buchgeher, David Gabauer, Jorge Martínez Gil, and Lisa Ehrlinger. 2021. Knowledge graphs in manufacturing and production: A systematic literature review. *IEEE Access*, 9:55537–55554.

Francisco Casacuberta and Colin de la Higuera. 2000. Computational complexity of problems on probabilistic grammars and transducers. In *Proceedings of the 2000 International Colloquium on Grammatical Inference (ICGI)*, pages 15–24, Lisbon, Portugal.

Zhiyu Chen, Jinsong Bao, Xiaohu Deng, Siyi Ding, and Tianyuan Liu. 2021. Semantic recognition method of assembly process based on lstm. *Computer Integrated Manufacturing Systems*, 27(6):1582.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of* *the Association for Computational Linguistics (ACL/IJCNLP)*, pages 1835–1845, Online.

Yuntian Feng, Hongjun Zhang, and Wenning Hao. 2015. Named entity recognition for military text. *Journal of Computer Science*, 42(7):15–18.

Ding Hongyu, Xi Daoyun, Zhang Xiufen, Han Linlin, Xiao Chengxaing, and Yunfeng Wang. 2008. *Fundamental Terminology of Mechanical Manufacturing Processes*. General Administration of Quality Supervision, Inspection and Quarantine of the People's Republic of China;Standardization Administration of China.

Yutai Hou, Cheng Chen, Xianzhen Luo, Bohan Li, and Wanxiang Che. 2022. Inverse is better! fast and accurate prompt for few-shot slot tagging. In *Findings of the Association for Computational Linguistics (ACL)*, pages 637–647, Dublin, Ireland.

Meng Jia, Peiyan Wang, Guiping Zhang, and Dongfeng Cai. 2022. Named entity recognition for process text. *Journal of Chinese Information Processing*, 36(3):54–63.

Chengyue Jiang, Zijian Jin, and Kewei Tu. 2021. Neuralizing regular expressions for slot filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9481–9498, Online.

Chengyue Jiang, Yinggong Zhao, Shanbo Chu, Libin Shen, and Kewei Tu. 2020. Cold-start and interpretability: Turning regular expressions into trainable recurrent neural networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3193–3207, Online.

Hang Jiang, Yining Hua, Doug Beeferman, and Deb Roy. 2022. Annotating the tweebank corpus on named entity recognition and building NLP models for social media analysis. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC)*, pages 7199–7208, Marseille, France.

Chu Cheng Lin, Hao Zhu, Matthew R. Gormley, and Jason Eisner. 2019. Neural finite-state transducers: Beyond rational relations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 272–283, Minneapolis, MN, USA.

Jianhong Ma, Liqin Wang, and Shuang Yao. 2018. Named entity recognition for chemical resource text. *Journal of Zhengzhou University (Natural Science Edition)*, 50(4):14–20.

Rendani Mbuvha, David Ifeoluwa Adelani, Tendani Mutavhatsindi, Tshimangadzo Rakhuhu, Aluwani Mauda, Tshifhiwa Joshua Maumela, Andisani Masindi, Seani Rananga, Vukosi Marivate, and Tshilidzi Marwala. 2023. Mphayaner: Named entity recognition for tshivenda. *CoRR*, abs/2304.03952.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 conference of the North American chapter of the Association for Computational Linguistics (NAACL)*, pages 623–633, San Diego California, USA.

Yuto Sakuma, Yasuhiko Minamide, and Andrei Voronkov. 2012. Translating regular expression matching into transducers. *Journal of Applied Logic*, 10(1):32–51.

Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. 2019. The performance of LSTM and bilstm in forecasting time series. In *Proceedings of the 2019 IEEE International Conference on Big Data (IEEE BigData)*, pages 3285–3292, Los Angeles, CA, USA.

Jeniya Tabassum, Mounica Maddela, Wei Xu, and Alan Ritter. 2020. Code and named entity recognition in stackoverflow. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, Online.

Yuanhe Tian, Han Qin, Fei Xia, and Yan Song. 2022. Chimst: A chinese medical corpus for word segmentation and medical term recognition. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference (LREC)*, Marseille, France.

Jun Wang, Mingyan Song, Qiang Ye, and Tong Zhang. 2020. Research on the key technologies of relation extraction from quality text for industrial manufacturing. In *Proceedings of the 2020 International Conference on Computer Science and Management Technology (ICCSMT)*, pages 366–369, Shanghai, China.

Haoyu Wu, Qing Lei, Xinyue Zhang, and Zhengqian Luo. 2020. Creating a large-scale financial news corpus for relation extraction. In *Proceedings of the 2020 3rd International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 259–263, Chengdu, China.

Hang Yan, Bocao Deng, Xiaonan Li, and Xipeng Qiu. 2019. TENER: adapting transformer encoder for named entity recognition. *CoRR*, abs/1911.04474.

Yi Yang and Arzoo Katiyar. 2020. Simple and effective few-shot named entity recognition with structured nearest neighbor learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6365–6375, Online.

Nana Zhang, Peiyan Wang, and Guiping Zhang. 2019. Named entity deep learning recognition method for process operation description text. *Computer Applications and Software*, 36(11):188–195.

Shunle Zhu. 2019. Deep learning based uyghur named entities recognition. *Journal of Computer Engineering And Design*, 40(10):2874–2878+2890.