An Approach for Process Model Extraction by Multi-grained Text Classification

Chen Qian[1], Lijie Wen[1](✉), Akhil Kumar[2], Leilei Lin[1], Li Lin[1], Zan Zong[1], Shu'ang Li[1], and Jianmin Wang[1]

[1] School of Software, Tsinghua University, Beijing 100084, China
qc16@mails.tsinghua.edu.cn, wenlj@tsinghua.edu.cn

[2] Smeal College of Business, Penn State University, State College 16802, USA

Abstract. Process model extraction (PME) is a recently emerged interdiscipline between natural language processing (NLP) and business process management (BPM), which aims to extract process models from textual descriptions. Previous process extractors heavily depend on manual features and ignore the potential relations between clues of different text granularities. In this paper, we formalize the PME task into the multi-grained text classification problem, and propose a hierarchical neural network to effectively model and extract multi-grained information without manually-defined procedural features. Under this structure, we accordingly propose the coarse-to-fine (grained) learning mechanism, training multi-grained tasks in coarse-to-fine grained order to share the high-level knowledge for the low-level tasks. To evaluate our approach, we construct two multi-grained datasets from two different domains and conduct extensive experiments from different dimensions. The experimental results demonstrate that our approach outperforms the state-of-the-art methods with statistical significance and further investigations demonstrate its effectiveness.

Keywords: Process model extraction · Multi-grained text classification · Coarse-to-fine learning · Convolutional neural network

1 Introduction

The widespread adoption of conversational agents such as Alexa, Siri and Google Home demonstrates the natural demand for such assistive agents. To go beyond supporting the simplistic queries such as "*what should I do next?*", these agents need domain-specific procedural knowledge [9]. Procedural knowledge, also called "*how-to-do-it*" knowledge, is the knowledge related to the execution of a series of interrelated tasks [26]. A major source of procedural knowledge is contained in natural textual instructions [34], such as cooking recipes that describe cooking procedures and maintenance manuals that describe repair procedures for various devices and gadgets. While it is possible to manually understand, extract and

S. Dustdar et al. (Eds.): CAiSE 2020, LNCS 12127, pp. 268–282, 2020.
https://doi.org/10.1007/978-3-030-49435-3_17

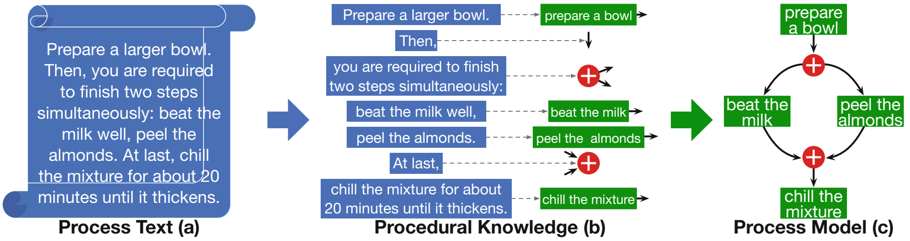

Prepare a larger bowl. Then, you are required to finish two steps simultaneously: beat the milk well, peel the almonds. At last, chill the mixture for about 20 minutes until it thickens.
Process Text (a)
Prepare a larger bowl.
prepare a bowl
Then,
you are required to finish two steps simultaneously:
beat the milk well,
beat the milk
peel the almonds.
peel the almonds
At last,
chill the mixture for about 20 minutes until it thickens.
chill the mixture
Procedural Knowledge (b)
prepare a bowl
beat the milk
peel the almonds
chill the mixture
Process Model (c)

Fig. 1. Illustration of the PME problem.

reuse such knowledge from texts, ultimately that is a very labor-intensive option [20]. To facilitate reuse and repurpose of procedural knowledge, **process model extraction (PME)** is emerging to automatically extract underlying **process models** from **process texts**. As Fig. 1 illustrates, PME extracts and presents the main actions (nodes) and their ordering relations (sequence flows) expressed in the cooking recipe as a process model. This task can liberate humans from the manual efforts of creating and visualizing procedural knowledge by making assistive agents understand process texts intelligently [9].

However, PME is challenging as it requires agents to understand complex descriptions of actions and involves multi-grained information mining. For example, in Fig. 1, to extract the whole process model, the process extractor has to recognize *whether a sentence is describing an action*? (sentence-level information) and *who does what in a sentence describing an action*? (word-level information). Recent research efforts have been made to extract the main procedural knowledge. For example, language-rule based extractors [12] used pre-defined language rules to extract procedural knowledge. Pattern-matching based extractors [26] used NLP tagging or parsing tools to analyze sentences and extract corresponding information from triggered patterns. Knowledge-based extractors employed pre-defined ontology [12] or world model [8] to help extract that information. However, traditional methods suffer from weak generalizability when applied in open-domain or open-topic scenarios since they: 1) require much large-scale and domain-specific procedural features; and 2) ignore the relations between sentence-level and word-level subtasks.

In this paper, we propose a new framework to extract process models from process texts. Specifically, we first formalize the PME task into the multi-grained text classification problem, and then propose a hierarchical neural network to effectively model and extract multi-grained information without manually defined procedural features. Under this hierarchical structure, we accordingly propose the coarse-to-fine (grained) learning mechanism, training multi-grained tasks in coarse-to-fine grained order, to share the high-level knowledge for the low-level tasks. To train and evaluate our model, we construct two multi-grained datasets from two different domains and conduct extensive experiments from different dimensions. Experimental results demonstrate that our approach outperforms state-of-the-art methods with statistical significance.

In summary, this paper makes the following contributions:

- We first formalize the PME task into the multi-grained text classification problem and design a new hierarchical network to model the conditional relation among multi-grained tasks. Supported by automatic feature extraction, it can extract procedural knowledge without employing manual features and defining procedural knowledge.
- We propose the coarse-to-fine learning mechanism that trains multi-grained tasks in coarse-to-fine (grained) order to apply the sentence-level knowledge for the word-level tasks.
- We construct two multi-grained datasets from two different domains to train and evaluate multi-grained text classifiers. The results demonstrate that our approach outperforms the state-of-the-art methods.

2 Related Work

Several **language-rule based** methods have been originally applied to process extraction [18,29,32,34]. Specifically, [34] introduced a generic semantic representation of procedures for analyzing instructions, using Stanford Parser to automatically extract structured procedures from instructions. [32] described an approach for the automatic extraction of workflows from cooking recipes resulting in a formal description of cooking instructions. A chain of standard information extraction pipeline was applied with the help of GATE. They were dedicated to the special characteristics of textual cooking instructions (verb centric, restricted vocabulary of ingredients, relatively independent sentences). Although they are easy to develop and interpret, they require a large number of linguistic rules created by domain experts.

Along this line, **pattern-matching based** methods [7,8,26–28,31] designed various language patterns, considering basic language patterns [27], syntactic tree [7,31] and anaphora resolution [26,28]. For example, [27] presented on the step of anaphora resolution to enrich the process models extracted by introducing a lexical approach and two further approaches based on a set of association rules which were created during a statistical analysis of a corpus of workflows. However, these studies are to some extent limited by domain-specific knowledge bases, making them not applicable for open-domain or open-topic scenarios.

Recently, some **knowledge based** methods [10,12,13] have been applied to this problem and they were shown to perform well. For example, [12] proposed an ontology-based workflow extraction framework that extended classic NLP techniques to extract and disambiguate tasks in texts. Using a model-based representation of workflows and a domain ontology, the extraction process used a context-based approach to recognize workflow components such as data and control elements in a flow. However, they also require a large quantity of cognition-level knowledge, such as a world model [8,10] or an ontology [12,13], which would be time-consuming and labor-intensive to build.

There also exist some **machine-learning based** studies which incorporated traditional machine learning techniques into process extraction [20,23].

[20] leveraged support vector machine to automatically identify whether a task described in a textual process description is manual or automated. [23] used semi-supervised conditional random fields and support vector machine to label process texts and recognize main information.

Other process-related works include process state tracing, action extraction and process search. For example, [3,4,30] proposed neural machine-reading models that constructed dynamic knowledge graphs from procedural text. [2] introduced a network to understand the procedural text through simulation of action dynamics. [24,25] set out to generate the textual descriptions from process models. [15] proposed a probabilistic model to incorporate aspects of procedural semantics and world knowledge. [1] aimed to answer biological questions by predicting a rich process structure and mapping the question to a formal query.

3 Methodology

We first formalize the PME task into the multi-grained text classification problem. Given a process text $\mathcal{T} = \langle S^1, S^2, \cdots, S^n \rangle$ where $S^i = \langle W_1^i, W_2^i, \cdots, W_{|S^i|}^i \rangle$ is a sequence of words, $\forall i = 1, 2, \cdots, n$. For each $S^i \in \mathcal{T}$, the key of PME (α) is to predict a corresponding set of labels that describe the type of the sentence and its corresponding word-level arguments, $\alpha(S^i) = (sType, sSemantic, sArgs)$. $sType$ indicates the type of sentence S^i, which can be *Action* or *Statement*. The *Action* indicates that S^i is an action mention and *Statement* refers to a non-action sentence. If S^i is categorized into *Action* type, then $sSemantic$ is $\varnothing$ (empty marker) and $sArgs = [aRole, aName, aObject]$ denotes the action's executor, action name and direct object respectively. Otherwise, $sArgs$ is $\varnothing$ and $sSemantic$ can be one of $\{\triangleright, \triangleleft, \bullet, \times, +\}$ relation symbols that determine how actions are coordinated. The five relations refer to the beginning of a block of actions (*block begins*), the ending of a block of actions (*block ends*), a successive relation, an optional relation and a concurrent relation, respectively.

Example 1. Consider the two sentences in Fig. 1: S^i = "you are required to finish two steps" and S^j = "chill the mixture for about 20 min until it thickens". S^i means that two following actions should be chosen and done concurrently, thus it is labeled as a *concurrency relation*, $\alpha(S^i) = (\mathit{Statement}, +, \varnothing)$. S^j is an *action mention*, thus it is labeled with its role, action name and object, $\alpha(S^j) = (\mathit{Action}, \varnothing, [\varnothing, \mathit{chill}, \mathit{mixture}])$.

We argue that PME involves three main text classification subtasks:

ST1 Sentence Classification (Sentence-level): identifying whether a sentence is describing an action or a statement.

ST2 Sentence Semantics Recognition (Sentence-level): recognizing the semantics of a *Statement* sentence to control the execution of following actions, i.e., *block begins*, *block ends*, *successive relation*, *optional relation* and *concurrency relation*.

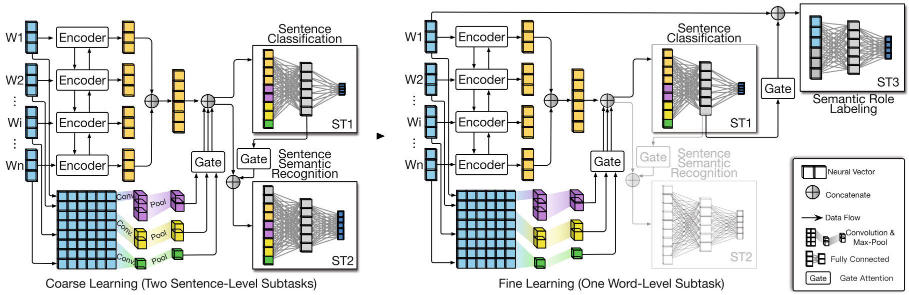


Fig. 2. High-level overview of the proposed method MGTC, which consists of two stages: a) The coarse-grained learning phase combines the bi-directional encoding layer and convolutional neural networks to *pre-train* two sentence-level tasks. b) The fine-grained learning phase utilizes the learned sentence-level knowledge to *fine-tune* the word-level task.

ST3 Semantic Role Labeling (Word-level): assigning semantic roles ($aRole$, $aName$, $aObject$) to words or phrases in an *Action* sentence.

Note that these three tasks are not independent because ST2 and ST3 are conditioned on ST1, i.e., for a single sentence, the result of ST1 determines whether the sentence is passed to ST2 or ST3.

3.1 Overall Framework

To mine sentence-level and word-level information effectively, we propose a deep-learning-based network to effectively avoid manually defining domain-specific procedural features, called Multi-Grained Text Classifier (MGTC). Figure 2 shows the framework of our proposed approach. Specifically, we design a hierarchical structure to model the conditional relations between three subtasks and to effectively extract textual clues from different granularities. Under this framework, we accordingly propose the coarse-to-fine (grained) learning mechanism, training coarse tasks in advance before training fine-grained tasks to share the learned high-level knowledge to the low-level tasks, which conforms with the procedure of human learning more than those methods without the consideration of different granularities. By problem mapping, the output of our model is the sentence-level and word-level labels (see Fig. 1(b) and Example 1), which could be further easily visualized (i.e., from Fig. 1(b) to Fig. 1(c)) as a specific/general process model or a process structure tree [25] via intuitively creating nodes and building sequence flows from the extracted procedural knowledge [8].

3.2 Coarse-Grained Learning of the Sentence-Level Knowledge.

The goal of this phase is to learn sentence-level knowledge in advance. First, it takes a sequence of embedded words $S^i = \langle W_1^i, W_2^i, \cdots \rangle$ as input. Then, the word vectors are bidirectionally encoded to capture inherent clues in a sentence. Meanwhile, a convolutional neural network (CNN) is used to capture local $n$gram features in a sentence. After concatenating $n$gram features and sentence embedding, two sentence-level tasks are trained jointly, using a weight-sharing multitask learning framework, to share learned high-level knowledge between two tasks and improve the generalization ability.

Embedding Layer. We use BERT [5] or Word2Vec to obtain word vectors. BERT is an autoencoding language model, which has been the currently state-of-the-art pre-training approach. Given the input token sequence, a certain portion of tokens are replaced by a special symbol [MASK], and the model is trained to recover the original tokens from the corrupted version. Since density estimation is not part of the objective, BERT is allowed to utilize bidirectional contexts for reconstruction. This can close the aforementioned bidirectional information gap in autoencoding language modeling, leading to improved performance.

Encoding Layer. As a special type of recurrent neural network (RNN), LSTM [14] is particularly suitable for modeling the sequential property of text data. At each step, LSTM combines the current input and knowledge from the previous steps to update the states of the hidden layer. To tackle the gradient vanishing problem of traditional RNNs, LSTM incorporates a gating mechanism to determine when and how the states of hidden layers can be updated. Each LSTM unit contains a memory cell and three gates (i.e., an input gate, a forget gate, and an output gate). The input and output gates control the input activations into the memory cell and the output flow of cell activations into the rest of the network, respectively. The memory cells in LSTM store the sequential states of the network, and each memory cell has a self-loop whose weight is controlled by the forget gate. Let us denote each sentence as (S_i, L_i), where $S_i = [W_i^1, W_i^2, \cdots, W_i^n]$ as a sequence of word vectors representing the plain text and L_i as its label; and $[d^1, d^2, \cdots, d^n]$ denotes the sequence of word vectors of S_i. At step t, LSTM computes unit states of the network as follows:

$$i^{(t)} = \sigma(U_i d^t + W_i h^{(t-1)} + b_i) \quad (1)$$

$$f(t) = \sigma(U_f d^t + W_f h^{(t-1)} + b_f) \quad (2)$$

$$o(t) = \sigma(U_o d^t + W_o h^{(t-1)} + b_o) \quad (3)$$

$$c(t) = f_t \odot c^{(t-1)} + i^{(t)} \odot \tanh(U_c d^t + W_c h^{(t-1)} + b_c) \quad (4)$$

$$h^{(t)} = o^{(t)} \odot \tanh(c^{(t)}) \quad (5)$$

where $i^{(t)}$, $f^{(t)}$, $o^{(t)}$, $c^{(t)}$, and $h^{(t)}$ denote the state of the input gate, forget gate, output gate, memory cell, and hidden layer at step t. W, U, b respectively denote the recurrent weights, input weights, and biases. $\odot$ is the element-wise product. We can extract the latent vector for each step T from LSTM. In order to capture the information from the context both preceding and following a word, we use the bi-directional LSTM (Bi-LSTM) [21]. We concatenate the latent vectors from both directions to construct a bi-directional encoded vector h_i for every single word vector W_i^j, which is:

$$\overrightarrow{h_i} = \overrightarrow{LSTM}(W_i^j), i \in [1, |S_i|] \tag{6}$$

$$\overleftarrow{h_i} = \overleftarrow{LSTM}(W_i^j), i \in [1, |S_i|] \tag{7}$$

$$h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}] \tag{8}$$

Convolution Layer. We also employ multiscale filters to capture local $n$gram information in a sentence. Let x_1^n refer to the concatenation of vectors $x_1, x_2, \cdots, x_n$. The convolution layer involves a set of filters $w \in \mathbb{R}^{h \times k}$, which is solely applied to a window of h to produce a new feature map $v =$:

$$\left[\begin{bmatrix} \sigma(w_i x_1^h + b_1) \\ \sigma(w_i x_2^{h+1} + b_1) \\ \cdots \\ \sigma(w_i x_{n-h+1}^n + b_1) \end{bmatrix}, \begin{bmatrix} \sigma(w_i x_1^h + b_2) \\ \sigma(w_i x_2^{h+1} + b_2) \\ \cdots \\ \sigma(w_i x_{n-h+1}^n + b_2) \end{bmatrix} \cdots \right] \tag{9}$$

where $\sigma(\cdot)$ is the sigmoid function and b_i is a bias term. Meanwhile, we use the max-pooling operation: $\hat{v} = max(v_i)$ to extract the most important features within each feature map.

Multi-task Learning. The goal of this module is to incorporate the multiple features for final sentence-level predictions. Since not all features contribute equally to the final task, we employ the gate attention mechanism to weight each concept information. For a feature representation z, we define the gate-attention as follows: $g = \sigma(Wz + b)$, where σ denotes the sigmoid activation function, which guarantees the values of g in the range $[0, 1]$. W and b are a weight and a bias term which need to be learned. Then, all convoluted representations are injected into the sentence representation by weighted fusing: $z_c = z_T \oplus (g \otimes z)$, where $\otimes$ denotes element-wise multiplication, $\oplus$ denotes concatenation and z_T is the BiLSTM representation. The effect of gate attention is similar to that of feature selection. It is a "soft" feature selection which assigns a larger weight to a vital feature, and a small weight to a trivial feature. Note that we additionally incorporate the hidden feature of ST1 into the input representations of ST2 via a self-attention gate to model the conditional relation between the two sentence-level tasks.

After concatenating all features, it is input into two fully connected multilayer perceptron (MLP) networks to realize feature fusion:

$$o_i = softmax(W_2 \cdot (W_1 \cdot V + b_1) + b_2) \tag{10}$$

where W_1, W_2, b_1 and b_2 are parameters of a network. To obtain the probability distribution on each type $t \in [1, T]$, the *softmax* operation is computed by: $p_k = \frac{\exp(o_t)}{\sum_{i=1}^T \exp(o_i)}$, where T is the class number of a classification task.

Model Training (The Coarse-Grained Learning Phase). We use cross-entropy loss function to train the coarse-grained learning phase, when given a set of training data x_t, y_t, e_t, where x_t is the t-th training example to be predicted, y_t is one-hot representation of the ground-truth type and e_t is the model's output. The goal of training is to minimize the loss function:

$$J(\theta_1, \theta_2) = -\,\lambda_1\big(\sum_{i=1}^{M}\sum_{t_1=1}^{t_1=T_1} y_t^{t_1}\cdot\log(e_t^{t_1})\big) - \lambda_2\big(\sum_{i=1}^{M}\sum_{t_2=T_1+1}^{t_2=T_1+T_2} y_t^{t_2}\cdot\log(e_t^{t_2})\big) \tag{11}$$
$$\lambda_1 + \lambda_2 = 1, \lambda_1, \lambda_2 \geq 0$$

where M is the number of training samples; $T_{1/2}$ is the category number of each subtask; $\lambda_{1/2}$ is a linear balance parameter. The multi-task learning phase can be further decomposed into learning two single tasks successively.

3.3 Fine-Grained Learning of the Word-Level Knowledge

Note that the features of a small-scale classification task may be higher quality than the features of a large-scale classification tasks since the small-scale classification task has relatively sufficient training data [16]. Based on that, compared with sentence-level tasks, we regard the word-level tasks as relatively "large-scale" classification tasks, initializing parameters of the word-level task from the sentence-level tasks and later fine-tuning it. The transfer of parameters can provide a better starting point for the latter than training from scratch using randomly initialized parameters [11].

As the right part of Fig. 2 shows, we extract the last-hidden features z_s in ST1 as learned sentence-level knowledge and concatenate it with word-level embedding z_w via the gate-attention mechanism. The fused representation $[z_s, g(z_w) \odot z_w]$ is fed into a MLP module to perform the word-level prediction. In this phase, we freeze ST2 (the light gray part) since ST2 and ST3 are independent.

Model Training (The Fine-Grained Learning Phase). Similarly, we use cross-entropy loss function to train the fine-grained learning phase, when given a set of training data x_t, y_t, e_t, where x_t is the t-th training example to be predicted, y_t is one-hot representation of the ground-truth type and e_t is the model's output. The goal of training is to minimize the loss function:

$$J(\theta_3) = -\sum_{i=1}^{M}\sum_{t_3=1}^{t_3=T_3} y_t^{t_3}\cdot\log(e_t^{t_3}) \tag{12}$$

where M is the number of training samples; T_3 is the category number of ST3.

4 Experiments

In this section, we first introduce the experimental setup (datasets, baselines and implementation details). The experimental results are then demonstrated to validate the effectiveness of our approach on different datasets.

4.1 Datasets

Since traditional methods mainly use off-the-shelf NLP tools to analyze sentences and extract corresponding information under the pre-defined features or patterns; thus, there was no directly available multi-grained corpus for the PME task. To this end, we constructed two multi-grained PME corpora for the task of extracting process models from texts:

- **Cooking Recipes (COR).** We collected cooking recipes from the world's largest food-focused social network[1]. This corpora has a large and diverse collection of more than 200 cooking recipes and covers every kind of meal type including appetizers, breakfast, desserts, drinks, etc.
- **Maintenance Manuals (MAM).** We collected raw maintenance manuals from a wiki-based site[2] that teaches people how to fix almost all devices. This corpora contains more than 160 maintenance descriptions and covers almost all devices including computer hardware, phones, cars, game consoles, etc.

In the two raw corpora, we first split all documents into sentences and manually assigned labels to them. The sentence-level tags denote whether a sentence is describing cooking ingredients or maintenance tools, performing main actions, or executing conditions. Furthermore, we split all sentences into words and manually assign labels to them to denote the executor, action name and ingredients/tools of actions, i.e., the semantic roles. The statistics of those datasets is given in Table 1.

Table 1. Statistics of the two multi-grained datasets.

Domain	COR	MAM
	Recipe	Maintenance
# Labeled sentences	2,636	2,172
# Labeled words	14,260	20,612
# Sentence-level categories	5	5
# Word-level categories	4	4

4.2 Baselines

We chose several representative baselines:

- A pattern-matching based method (PBSW) [27], which uses NLP tagging tools to extract important linguistic patterns and adopts a set of heuristic anaphora resolution mechanisms to extract corresponding procedural knowledge.

[1] https://www.recipe.com.

[2] https://www.ifixit.com.

- A language-rule based method (ARWE) [28], which introduces a lexical approach and two further approaches based on a set of association rules created during a statistical analysis of a corpus of workflows.
- A traditional-learning based method (RPASVM) [20], which leverages SVM to automatically identify whether a task described in a textual description is manual, an interaction of a human with an information system or automated.

4.3 Implementation Details

We used BERT [5] with text representations of size 100 (also can use Word2Vec [22] in our architecture). In training, we used a Rectified Linear Unit (ReLU) as an activation function [19] and the Adam optimizer [17]. The network was run for 1,000 iterations with a mini-batch size of 32 at the coarse and the fine training. The learning rate is 10^{-4}. We implement MGTC using Python 3.7.3[3] and Tensorflow 1.0.1[4]. All of our experiments were run on a single machine equipped with an Intel Core i7 processor, 32 GB of RAM, and an NVIDIA GeForce-GTX-1080-Ti GPU. For comparison, all methods have been evaluated with the same training and test data. We divide datasets into train/test sets using an 8:2 ratio. The statistical significance between all baseline methods and MGTC is tested using a two-tailed paired t-test [6]. Our code and data are available at https://github.com/qianc62/MGTC.

4.4 Overall Performance

We compare all baseline methods, MGTC and its variants in terms of *classification accuracy*. Besides, we extract corresponding process models from the classification results and compare the *behavior similarity* between extracted models and the gold models using existing behavior evaluation methods [33]. The overall results (accuracy of predicted labels) are summarized in Table 2. From the evaluation results, we can make several key observations:

1) Our proposed model MGTC consistently outperforms all methods on all subtasks with statistical significance. On ST1 (single sentence classification), MGTC improves the accuracy by 7.97% and 7.24% on COR and MAM against the strongest competitor, respectively. On ST2 (sentence semantics recognition), MGTC improves the accuracy by 3.92% and 13.94% on COR and MAM, respectively. On ST3 (semantic role labeling), MGTC still outperforms all methods, improving accuracy by 9.75% and 2.18%, respectively. We believe that this is promising as word-level tasks face the problem of sparseness and ambiguity compared with sentence-level tasks, i.e., words have not relatively enough contextual information, which poses a great challenge for ST3. Moreover, in terms of behavior similarity between the extracted models and the gold models, employing the deep-learning-based framework improves

[3] https://www.python.org.
[4] https://www.tensorflow.org.

Table 2. Experimental results (accuracy; %) of all baselines, MGTC and its variants. *DOP*, *PPP*, *TVC*, *OPM*, *ARM* and *FMS* denote mechanisms applied in reference papers. ▲ and △ indicate the best and the second-best performing methods among all the baselines, respectively. The best performance among all methods is highlighted in boldface. The * denotes statistical significance ($p \leq 0.05$) compared to MGTC.

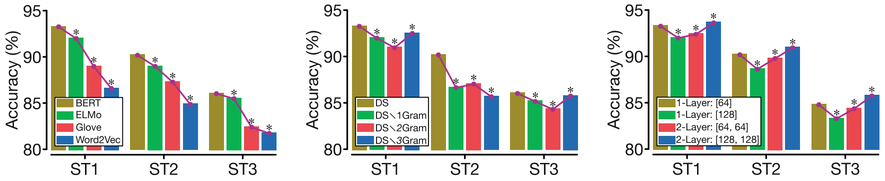
Method	Task & Dataset							
	ST1		ST2		ST3		PME	
	COR	MAM	COR	MAM	COR	MAM	COR	MAM
PBSW+*DOP*	47.63_*	47.58_*	43.66_*	39.49_*	34.38_*	35.60_*	30.35_*	30.27_*
PBSW+*DOP*+*PPP*	51.52_*	52.55_*	45.40_*	45.67_*	48.58_*	47.36_*	38.04_*	37.17_*
PBSW+*DOP*+*PPP*+*TVC*	66.53_*	47.42_*	55.48_*	46.65_*	50.28_*	49.62_*	46.27_*	36.07_*
ARWE+*OPM*	70.66_*	60.46_*	62.43_*	62.45_*	58.46_*	61.62_*	52.46_*	51.09_*
ARWE+*OPM*+*ARM*	76.39_*	69.51_*	56.52_*	$72.53^{\triangle}_*$	$71.38^{\triangle}_*$	$77.62^{\triangle}_*$	$65.21^{\triangle}_*$	59.50_*
ARWE+*OPM*+*ARM*+*FMS*	71.63_*	69.60_*	61.37_*	64.48_*	$72.64^{\blacktriangle}_*$	$78.26^{\blacktriangle}_*$	$69.57^{\blacktriangle}_*$	$66.11^{\blacktriangle}_*$
RPASVM+*1* gram	$82.53^{\triangle}_*$	$81.65^{\triangle}_*$	$84.44^{\triangle}_*$	71.55_*	63.39_*	60.39_*	57.76_*	60.23_*
RPASVM+*1* gram+*2* gram	$85.37^{\blacktriangle}_*$	$84.50^{\blacktriangle}_*$	$87.61^{\blacktriangle}_*$	$72.55^{\blacktriangle}_*$	67.33_*	64.53_*	56.59_*	$61.23^{\triangle}_*$
MGTC	**93.34**	**91.74**	**91.53**	**86.49**	**82.39**	**80.44**	**77.40**	**75.77**
MGTC$\setminus$Gate Mechanism	90.37_*	88.58_*	89.40_*	84.57_*	77.62_*	76.42_*	67.39_*	71.64_*
MGTC$\setminus$Coarse-to-Fine	91.55_*	89.45_*	88.31_*	84.53_*	79.44_*	76.22_*	74.46_*	72.29_*

behavior accuracy by 7.83% and 12.66% respectively, which further verifies that MGTC can extract procedural knowledge without employing manual features and complex procedural knowledge.

2) From ablation studies, we can see that the performance improves after employing the gate-attention mechanism and coarse-to-fine learning. For example, employing coarse-to-fine learning improves the behavior similarity by 2.94% and 3.48% on COR and MAM. That is to say, both the gate-attention mechanism and coarse-to-fine learning can improve accuracy, which shows the effectiveness of the two mechanisms. This is consistent with our intuition that training coarse tasks in advance before fine-grained tasks can learn to better learning procedural knowledge.
3) The experimental results also demonstrate the disadvantages of the manual-feature-based methods. First, since traditional methods employ diverse manual mechanisms or features, they suffer from the problems of poor quality and lack of adaptability. In contrast, deep-learning-based methods learn hidden contextual features automatically and always maintain their superior performance. Second, by considering the long-term language changes, the framework with mixture of multiscale filters can understand texts' semantics better and extract more accurate information. Third, over all subtasks and datasets, we can see that MGTC can maintain more stable results, which suggests that MGTC is more robust on different subtasks and datasets.

4.5 Further Investigation

To further investigate the independent effect of the key parameters or components in our framework, we compare our method with those replacing with



(a) The effect of language models.

(b) The effect of *n*gram filters

(c) The effect of #hidden layers

Fig. 3. Ablation analysis of different components (language models, *n*gram filters, the number of hidden layers) on three subtasks. DS denotes the default settings. $\setminus$ is the removing operation. * denotes statistical significance ($p \leq 0.05$) compared to MGTC.

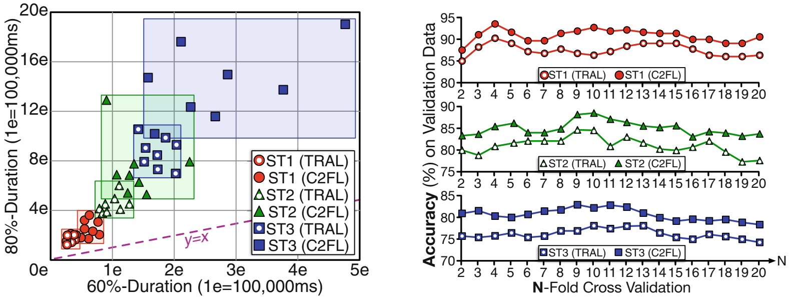
other standard components, using the default settings described in Sect. 4.3. The results are shown in Fig. 3.

Effect of Language Models. Fig. 3(a) shows the performance of those variants employing different language models (BERT, ELMo, GloVe and Word2Vec). It can be observed that, when the default BERT is replaced with ELMo, GloVe or Word2Vec, the performance consistently decreases. Particularly, on ST3, using Word2Vec decreases the accuracy about 4.58 points, which shows that pre-training on large-scale corpora can better capture linguistic information.

Effect of Multi-scale *n*gram. Fig. 3(b) shows the performance of the variants after removing *n*gram (n = 1,2,3) filters from MGTC. It can be observed that, when *n*gram filters are removed from the network, it decreases performance of on average by 3.18%/4.02%/1.06% on the three tasks respectively. This demonstrates that *n*gram filters make a contribution to the promising scores. In particular, removing *3* gram filters obviously influences the performance of ST2, suggesting that *3* gram features play an important role in ST2.

Effect of the Number of Hidden Layers. From Fig. 3(c), we can observe that the performance of two-layer MLP indicates that imposing some more non-linearity is useful, and that the one-layer networks seem slightly weak for certain subtasks. However, further increasing the number of layers may degrade the performance because the model becomes too complicated to train. Using a two-layer architecture seems like a good trade-off between speed and accuracy.

We also conducted more experiments to further investigate the performance of traditional learning (TRAL) and our proposed coarse-to-fine learning method (C2FL). From Fig. 4(a) we can observe that ST1 first converges after a 1.8e average duration (1e = 10^5 ms), followed by ST2 (4.1e) and ST3 (8.2e). Moreover, comparing the results of TRAL and C2FL, one can see that C2FL consistently magnifies and scatters the time consumption on three subtasks. The main reason for this is that coarse-to-fine learning architecture would leverage learned features from sentence-level tasks to word-level tasks in a gradual manner; thus, word-level tasks (ST3) would consume more time than sentence-level tasks (ST1 and ST2). From Fig. 4(b), the N-fold cross-validation results show that the generalization performance of C2FL is remarkably higher than TRAL's on all subtasks. This further indicates that employing coarse-to-fine learning is more useful in extracting information with different granularities. Therefore, we can conclude that, although at the expense of longer running time, coarse-to-fine learning tends to converge to a higher performance.



(a) Time consumption.

(b) N-fold (N=2,3,⋯,20) cross validation results.

Fig. 4. Further investigation of the traditional learning (TRAL) and our proposed coarse-to-fine learning (C2FL).

5 Discussion

Note that many publicly-available model-text pairs (e.g., [8]) are not considered as our datasets, because our supervised method inevitably needs **much-larger** scale of **labeled** (both sentence- and word-level) training data than traditional rule-based ones. Both requirements (data scale and label) are not fully satisfied on previous datasets which are almost uniformly small or unlabeled.

Moreover, the datasets used in this paper consist of relatively general process texts. Thus, a classic pattern-matching based method [8] is not considered in this version because it mainly focuses on extracting Business Process Model and Notation (BPMN) that contains many **specific** elements, such as swimlanes and artifacts, whose attributes currently have no corresponding specifically-defined labels to perform procedural knowledge extraction via our "relatively general" model. Adding such features is a possible extension of our work.

It is also worth mentioning that although we aim to extract from general process texts, our model might suffer from low external validity. One main reason is that both recipes and manuals are rather sequential by nature, which means that we can expect a rather small number of concurrent streams of actions and complex choices. However, note that deep-learning-based methods, to some extent, tend to be more generalized than pattern-based ones, due to their ability for "adaptive knowledge learning" (rather than those that work with a fixed human-defined knowledge). This limitation further motivates us to design more general and robust models in future work.

6 Conclusion

In this paper, we formalize the PME task into the multi-grained text classification problem and propose a hierarchical multi-grained network to model and extract multi-grained information without manually defined procedural features. Under this structure, we accordingly propose the coarse-to-fine learning mechanism, training multi-grained tasks in coarse-to-fine grained order, to apply the high-level knowledge for the low-level tasks. The results demonstrate that our approach outperforms the state-of-the-art methods with statistical significance and the further investigations demonstrate its effectiveness. Therefore, we draw two main conclusions as follows: 1) The deep-learning-based process extractor can effectively capture procedural knowledge without the need for defining domain-specific procedural features; and 2) Our proposed hierarchical network and the coarse-to-fine learning mechanism can better learn the word-level clues based on pre-learned sentence-level knowledge.

Acknowledgement. The work was supported by the National Key Research and Development Program of China (No. 2019YFB1704003), the National Nature Science Foundation of China (No. 71690231) and Tsinghua BNRist. Lijie Wen is the corresponding author.

References

1. Berant, J., Srikumar, V., Chen, P.C., et al.: Modeling biological processes for reading comprehension. In: EMNLP (2014)
2. Bosselut, A., Levy, O., Holtzman, A., et al.: Simulating action dynamics with neural process networks. In: ICLR (2018)
3. Dalvi, B.B., Huang, L., Tandon, N., et al.: Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In: NAACL (2018)
4. Das, R., Munkhdalai, T., Yuan, X., et al.: Building dynamic knowledge graphs from text using machine reading comprehension. arXiv:1810.05682 (2018)
5. Devlin, J., Chang, M.W., Lee, K., et al.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL (2019)
6. Dror, R., Baumer, G., Shlomov, S., Reichart, R.: The Hitchhiker's guide to testing statistical significance in natural language processing. In: ACL (2018)
7. Epure, E.V., Martín-Rodilla, P., Hug, C., et al.: Automatic process model discovery from textual methodologies. In: RCIS, pp. 19–30 (2015)
8. Fabian, F., Jan, M., Frank, P.: Process model generation from natural language text. In: CAiSE (2011)
9. Feng, W., Zhuo, H.H., Kambhampati, S.: Extracting action sequences from texts based on deep reinforcement learning. In: IJCAI (2018)
10. Friedrich, F.: Automated generation of business process models from natural language input (2010)
11. Haj-Yahia, Z., Deleris, L.A., Sieg, A.: Towards unsupervised text classification leveraging experts and word embeddings. In: ACL (2019)
12. Halioui, A., Valtchev, P., Diallo, A.B.: Ontology-based workflow extraction from texts using word sense disambiguation. bioRxiv (2016)

13. Halioui, A., Valtchev, P., Diallo, A.B.: Bioinformatic workflow extraction from scientific texts based on word sense disambiguation and relation extraction. IEEE/ACM Trans. Comput. Biol. Bioinform. **15**(6), 1979–1990 (2018)
14. Hochreiter, S., Schmidhuber, J.: Long short-term memory. In: Neural Computation (1997)
15. Kiddon, C., Ponnuraj, G.T., Zettlemoyer, L., et al.: Mise en place: unsupervised interpretation of instructional recipes. In: EMNLP (2015)
16. Kim, K.M., Kim, Y., Lee, J., et al.: From small-scale to large-scale text classification. In: WWW (2019)
17. Kingma, D.P., Ba, J.L.: Adam: a method for stochastic optimization. In: ICLR (2015)
18. Kolb, J., Leopold, H., Mendling, J., et al.: Creating and updating personalized and verbalized business process descriptions. In: The Practice of Enterprise Modeling, pp. 191–205 (2013)
19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NeurIPS, pp. 1097–1105 (2012)
20. Leopold, H., van der Aa, H., Reijers, H.A.: Identifying candidate tasks for robotic process automation in textual process descriptions. In: Enterprise, Business-Process and Information Systems Modeling (2018)
21. Melamud, O., Goldberger, J., Dagan, I.: Learning generic context embedding with bidirectional LSTM. In: CoNLL (2016)
22. Mikolov, T., Sutskever, I., Chen, K., et al.: Distributed representations of words and phrases and their compositionality. In: NeurIPS (2013)
23. Ni, W., Wei, Z., Zeng, Q., Liu, T.: Case information extraction from natural procedure text. CIMS **24**(7), 1680–1689 (2018)
24. Qian, C., Wen, L., Kumar, A.: BePT: a behavior-based process translator for interpreting and understanding process models. In: CIKM, pp. 1873–1882 (2019)
25. Qian, C., Wen, L., Wang, J., Kumar, A., Li, H.: Structural descriptions of process models based on goal-oriented unfolding. In: CAiSE, pp. 397–412 (2017)
26. Schumacher, P., Minor, M.: Extracting control-flow from text. In: IRI, pp. 203–210, August 2014. https://doi.org/10.1109/IRI.2014.7051891
27. Schumacher, P., Minor, M., Schulte-Zurhausen, E.: Extracting and enriching workflows from text. In: IRI, pp. 285–292, August 2013
28. Schumacher, P., Minor, M., Schulte-Zurhausen, E.: On the use of anaphora resolution for workflow extraction. In: Bouabana-Tebibel, T., Rubin, S.H. (eds.) Integration of Reusable Systems. AISC, vol. 263, pp. 151–170. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04717-1_7
29. Schumacher, P., Minor, M., Walter, K.: Extraction of procedural knowledge from the web: a comparison of two workflow extraction approaches. In: WWW (2012)
30. Tandon, N., Mishra, B.D., Grus, J., et al.: Reasoning about actions and state changes by injecting commonsense knowledge. In: EMNLP (2018)
31. Vakulenko, S.: Extraction of process models from business process descriptions (2011)
32. Walter, Kirstin, Minor, M., Bergmann, R.: Workflow extraction from cooking recipes. In: ICCBR (2011)
33. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. In: TSE (2014)
34. Zhang, Z., Webster, P., Uren, V., et al.: Automatically extracting procedural knowledge from instructional texts using natural language processing. In: LREC (2012)