Solving Sequential Text Classification as Board-Game Playing

Chen Qian
Tsinghua University
qc16@mails.tsinghua.edu.cn

Fuli Feng
National University of Singapore
fulifeng93@gmail.com

Lijie Wen*
Tsinghua University
wenlj@tsinghua.edu.cn

Zhenpeng Chen
Peking University
czp@pku.edu.cn

Li Lin
Tsinghua University
veralin1994@gmail.com

Yanan Zheng
Tsinghua University
zhengyanan932@gmail.com

Tat-Seng Chua
National University of Singapore
chuats@comp.nus.edu.sg

Abstract

Sequential Text Classification (STC) aims to classify a sequence of text fragments (e.g., words in a sentence or sentences in a document) into a sequence of labels. In addition to the intra-fragment text contents, considering the inter-fragment context dependencies is also important for STC. Previous sequence labeling approaches largely generate a sequence of labels in left-to-right reading order. However, the need for context information in making decisions varies across different fragments and is not strictly organized in a left-to-right order. Therefore, it is appealing to label the fragments that need less consideration of context information first before labeling the fragments that need more. In this paper, we propose a novel model that labels a sequence of fragments in jumping order. Specifically, we devise a dedicated board-game to develop a correspondence between solving STC and board-game playing. By defining proper game rules and devising a game state evaluator in which context clues are injected, at each round, each player is effectively pushed to find the optimal move without position restrictions via considering the current game state, which corresponds to producing a label for an unlabeled fragment jumpily with the consideration of the contexts clues. The final game-end state is viewed as the optimal label sequence. Extensive results on three representative datasets show that the proposed approach outperforms the state-of-the-art methods with statistical significance.

Introduction

Sequential Text Classification (STC) is a fundamental and critical research problem in natural language processing (NLP) (Lee and Dernoncourt 2016). The goal of STC is to classify a sequence of text fragments into a sequence of labels. STC involves different text granularities (e.g., words in a sentence or sentences in a document) and serves dual purposes: 1) improving the accuracy of single text classification by incorporating context information (Lee and Dernoncourt 2016); and 2) mining informative text clues at different levels of granularity (Qian et al. 2019). STC can benefit a diversity of NLP tasks, such as the part-of-speech tagging (Ratnaparkhi 1996), dialog act recognition (Liu, Han, and others 2017), fine-grained sentiment analysis (Wang et al. 2018) and clause-level aspect classification (Friedrich, Palmer, and Pinkal 2016).

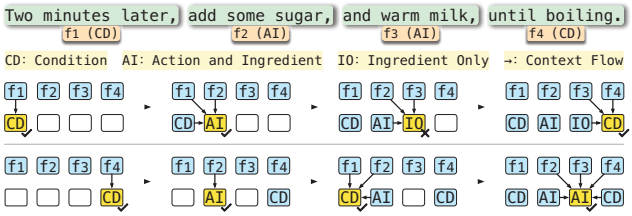

Figure 1: Traditional successive labeling (above) produces a sequence of labels in left-to-right order, while our proposed jump labeling (below) produces a sequence of labels in jumping order.

The key challenge of STC is how to effectively capture and utilize context information because the label of a fragment would be better forecasted if we consider the text information and label information of other fragments (Yang et al. 2016; Lee and Dernoncourt 2016). Recent studies show that the use of both the bidirectional language representation models to encode the contextual text information and sequence labeling models to model the contextual label dependencies is a promising solution (Kumar et al. 2018; Al-Zaidy, Caragea, and Giles 2019). However, in the labeling process, almost all of methods impose the Markov assumption (Rabiner 1989) that the current state is conditionally dependent upon the previous state(s); thus, they label a sequence in the left-to-right reading order (referred to as *successive labeling*). This restricts their ability to capture more beneficial context clues. The upper part of Figure 1 gives an example that illustrates the processes by which a linear-chain conditional random field (LCRF) model classifies a sequence of clause-level fragments. We can see that the LCRF classifies the four fragments in the left-to-right order. In such a case, it might misclassify the third fragment "*and warm milk*" as `IO` (instead of `AI`) without considering the backward clue from the subsequent neighbor "*until boiling*".

Must the label generation process be following the left-

*Lijie Wen is the corresponding author.

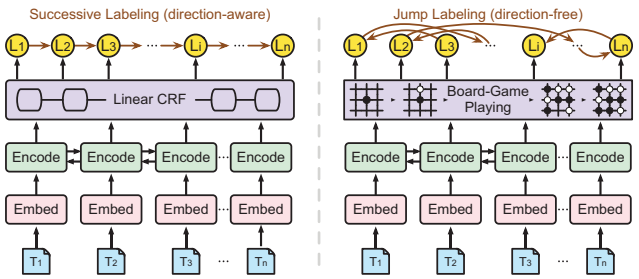

Figure 2: Mainstream STC framework (left) and our proposed model GuGo (right).

to-right reading order? In this paper, we explore a new paradigm that labels the sequential fragments in jumping order (referred to as *jump labeling*). Our intuition is that the need for context information in making decisions varies across different fragments. The order of fragments according to the need for context information needs not be strictly organized in the left-to-right order. Take the bottom part of Figure 1 as an example, the fourth fragment that includes the word "*until*" could be easily classified as `CD` in advance without considering any context information. After the fourth fragment is labeled, the third fragment ("*and warm milk*") could be correctly classified as `AI` by considering the additional context - "*until boiling*" and `CD`. Therefore, a potential solution is to pre-predict the fragments that need less advance consideration of context information to provide more context clues for those fragments that need more. As such, the performance of jump labeling largely relies on the order of labeling sequential fragments (referred to as *labeling order*). Considering that the number of possible labeling orders grows exponentially with the number of fragments, i.e., the state space explosion problem (Groote, Kouters, and Osaiweran 2015), it is challenging to perform jump labeling properly and effectively.

In this paper, we propose a game-based jump labeling model (GuGo) equipped with an efficient jump labeling mechanism to solve the STC task. The key idea is to map the act of classifying fragments to the act of playing a board-game. By defining proper game rules and devising a game state evaluator in which the intra-fragment and inter-fragment context clues are fully injected as the game state evaluation factors, the game will push each player to play the optimal move (i.e., produce a label for a certain fragment) at each round with the best usage of the current game state (i.e., the injected context clues). The final game-end state is viewed as the optimal label sequence of the STC task. Transforming the STC problem into playing a board-game provides our model with three advantages: 1) the way in which players place pieces without position restrictions naturally corresponds to producing labels for unlabeled fragments jumpily in STC; 2) the way in which players evaluate a candidate move from global checkerboard layouts naturally corresponds to bidirectional context incorporation in STC; and 3) utilizing efficient game tree search effectively avoids the state space explosion problem.

Figure 2 shows the architectures of the mainstream STC framework and the proposed model. In our model, an embedding layer is first used to learn a semantic representation for each fragment. These representations are then fed into an encoding layer that implicitly encodes contextual text information. Finally, a game-playing layer performs jump labeling by explicitly taking into consideration the context clues. In summary, we make the following main contributions:

- We propose the idea of jump labeling and devise a new model equipped with the jump labeling mechanism for STC. As compared with successive labeling, jump labeling can choose a better labeling order, providing various degrees of context information for different fragments. To our knowledge, we are the first study that performs the (direction-free) jump labeling paradigm.
- We propose a new operator that performs jump labeling in a board-game-playing manner. By utilizing the efficient game tree search and the proposed speedup strategies, our approach can effectively avoid the state space (i.e., the labeling order) explosion problem.
- The experimental results on three representative datasets show that our proposed approach significantly outperforms the state-of-the-art methods, validating the effectiveness of the proposed method and the jump labeling mechanism. The code is publicly available at `https://github.com/qianc62/GuGo`.

Related Work

STC has been studied extensively within various NLP tasks with different text granularities, including part-of-speech (POS) tagging (Ratnaparkhi 1996), named entity recognition (NER) (Zhou and Su 2002), semantic roles labeling (SRL) (Gildea and Jurafsky 2002) and dialogue act tagging (Ji and Bilmes 2005). Some feature-based methods classify each fragment independently, i.e., each fragment is viewed as an individual text (Blatat, Mrakova, and Popelinsky 2004; Yeung and Lee 2015). However, this strategy relies on a set of handcrafted features and/or does not take into account the inherent dependencies across fragments.

To remedy this, STC solutions were explored in deep learning frameworks to incorporate contextual text information. For example, (Santos and Zadrozny 2014) proposed a deep neural network that learns character-level representation to perform POS tagging. (Kalchbrenner and Blunsom 2013) investigated the possibility of using CNN for dialogue act classification. In addition, (Lee and Dernoncourt 2016) presented a model based on RNN+CNN to incorporate contexts for short text classification.

To further incorporate the contextual label information, linear statistical methods were widely studied, including the hidden Markov models (HMM) (Stolcke et al. 2000; Venkataraman et al. 2003), maximum entropy models (MEM) (Ratnaparkhi 1996) and conditional random fields (CRF) (Kim, Cavedon, and Baldwin 2010; Quarteroni, Ivanov, and Riccardi 2011). More recently, some studies combined deep-learning-based representation models and the linear statistical models to incorporate both the contextual text and label information. For example, (Al-Zaidy, Caragea, and Giles 2019) combined BiLSTM+CRF model

for keyphrase extraction and (Ye and Ling 2018) for sequence labeling. (Ma and Hovy 2016) introduced a network that benefits from both the word-level and character-level representations by using the BiLSTM+CNN+CRF model for POS tagging and NER. (Wu et al. 2019) introduced a CNN+LSTM+CRF architecture to capture local and long-distance contexts.

As combining bidirectional representation models and the linear statistical models together, especially BiLSTM+CRF, can effectively capture context information, the STC task has gradually come to be regarded as the sequence labeling problem. However, almost all of sequence labeling techniques are limited to left-to-right order. In this paper, we eliminate the traditional successive labeling paradigm via jump labeling.

Methodology

We first formulate the STC problem: given a sequence of text fragments $\mathcal{F} = \langle F_1, F_2, \cdots, F_n \rangle$ and a predefined category set $\mathcal{C}$, the goal of STC is to predict a sequence of labels $\mathcal{L} = \langle L_1, L_2, \cdots, L_n \rangle$ such that each $L_i \in \mathcal{C}$ $(1 \leq i \leq n)$ describes the category of F_i. Note that a fragment $F_i \in \mathcal{F}$ can be any text content, e.g., a sentence, a clause or a word.

We propose a game-based jump labeling model (GuGo) to predict sequential fragments in a jumping manner. The high-level overview of GuGo is shown in Figure 2. GuGo consists of three main phases: 1) given a sequence of fragments, a pretrained language model, BERT (Devlin et al. 2019), is first used to learn fragment embeddings by pooling word embeddings; 2) these representations are then fed into a BiLSTM encoding layer (Melamud, Goldberger, and Dagan 2016) to implicitly encode the contextual text information and an MLP to project each fragment embedding to a preliminary probability; 3) the encoded vectors are then passed to the game-playing layer for jump labeling.

Specifically, the game-playing layer: 1) maps each encoded vector (one vector per fragment) into a prior probability over categories via a pretrained multilayer perceptron (MLP) (Trischler et al. 2016); and 2) utilizes these probabilities as factors to evaluate game states and produce the refined predictions ($\hat{\mathcal{L}}$) jumpily. In the following, we will demonstrate how we perform jump labeling to shed light on its rationale. First, we compare the processes of labeling sequential fragments to the processes of playing a board-game (for jump labeling). Then, we design a new board-game (for problem transformation). Finally, we utilize an efficient game search to play the game (to find the optimal label sequence).

Problem Mapping

First, we map the processes of STC to the processes of board-game playing for jump labeling. Typically, there are three main steps in playing the board-game: 1) the game starts when all squares are empty; 2) players place pieces on empty squares iteratively; and 3) the game ends when all squares are occupied. As in the game-playing process, there are three main phases in sequence labeling: 1) the to-be-predicted fragments are initialized with unlabeled states; 2) the labeler fills labels for unlabeled fragments iteratively; and 3) the process terminates when all fragments are labeled.

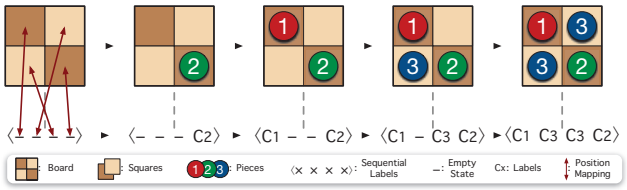

Figure 3: Graphical illustration of the process mapping between board-game playing and sequence labeling.

Based on this observation, we map n fragments of a sequence to n squares of a checkerboard, to establish the one-to-one mapping between a sequence and a checkerboard (see Figure 3). In this way, labeling a sequence can be transformed into playing a board game, and placing pieces without position restrictions naturally corresponds to producing labels jumpily, i.e., jump labeling.

Game Design

Given a sequence of text fragments $\mathcal{F} = \langle F_1, F_2, \cdots, F_n \rangle$ and a category set $\mathcal{C}$, we design a new board-game for problem transformation. The key of such board-game is to design appropriate game rules that make the final game-end state correspond to the optimal solution of STC.

Game Elements. There is a checkerboard with n squares arranged in an $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil$ grid[1], $|\mathcal{C}|$ kinds of pieces and two players[2]. The i^{th} fragment in a sequence is mapped to a square coordinate (j, k) of a checkerboard using $\psi_n(i) = (j, k) = (\lceil \frac{i}{\sqrt{n}} \rceil, i - (\lceil \frac{i}{\sqrt{n}} \rceil - 1)\lceil\sqrt{n}\rceil)$, i.e., map n fragments to n squares of the checkerboard in row-major order. It should be noted that although we employ the row-major order to facilitate the presentation and illustration, any one-to-one mapping (or any board shape) is acceptable.

Game Rules. Given these game elements, we define three main game rules. 1) The game starts when all squares are empty and ends when all are occupied. 2) Each player begins with an unlimited amount of $|\mathcal{C}|$ kinds of pieces. At each round, each player places one piece on an unoccupied square (this action is also referred to as *making a move*), after which the players take alternate turn. 3) After each move, the player can earn a bonus that is dependent on the quality of that move. The quality of a move is defined as how well the current game state is considered, which involves the intra-fragment features and the inter-fragment dependencies (detailed below). The goal of each player is to maximize his/her total bonuses in the whole game. The final game-end state is viewed as the labeling output of the STC task.

[1] $\lceil x \rceil$ rounds x up to an integer, i.e., the ceiling operation. When n is not a square number, i.e., $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil > n$, we can easily remove the last $\lceil\sqrt{n}\rceil \times \lceil\sqrt{n}\rceil - n$ redundant squares.

[2] Actually, the game can be designed with multiple versions to solve jump labeling, including 1 player, 2 players and even n (#piece types) players. Under different versions, the game rules should be correspondingly changed to achieve the desired labeling sequence.

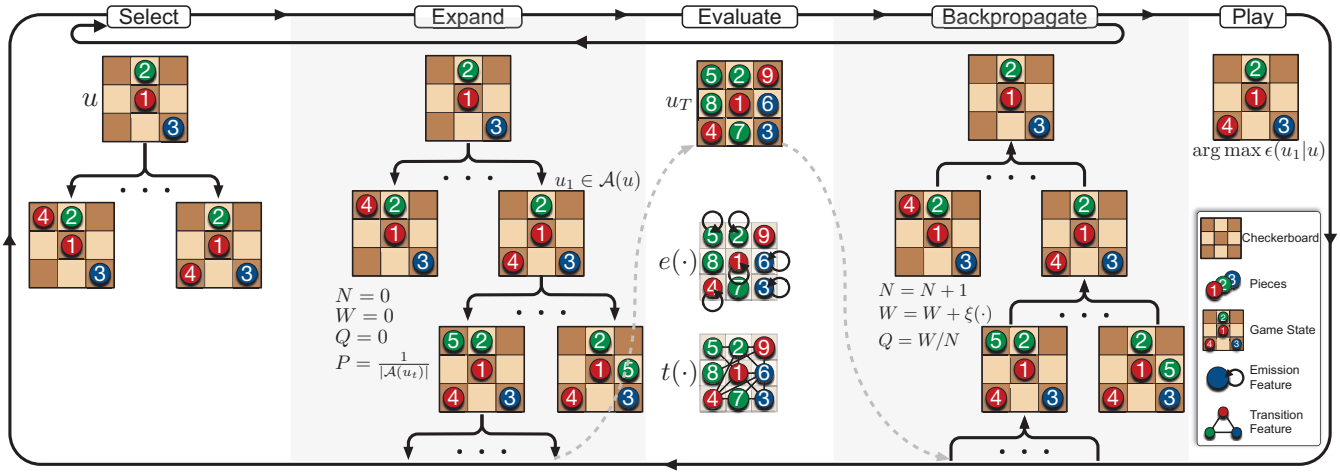

Figure 4: The architecture of searching for the best move. Each simulation traverses the tree by selecting the best potential candidate move. Then, the leaf node is expanded and evaluated before the edge statistics are updated in a backward fashion. Once the search is complete, the best move is selected to play.

The reason why we set two players for the game is to conform with the Minimax principle used in game theory (Straffin 1993) that each player has to logically analyze and make his/her best move at each round, given that another player is logically analyzing the best way to achieve his/her ends. Besides, by designing the game state evaluator in which intra-fragment features and inter-fragment dependencies are fully injected as the evaluation factors, two players will fight for moves with higher chance to earn more bonuses by appropriately utilizing the current game state. For STC, the game rules would push each player to find the optimal label for a certain unlabeled fragment at each round with the best usage of both intra-fragment and inter-fragment information.

Playing the Game

Inspired by the remarkable success of Monte Carlo Tree Search (MCTS) in board-games (Silver et al. 2017), we employ MCTS to find the best move to obtain the most bonus for a player at each round. Technically, each node in the MCTS tree represents a specific game state that consists of all square status (could be empty or a specific type of piece). At each iteration, given the game state u of current round, MCTS searches for the best subsequent move from u. Typically, MCTS constructs a search tree evaluated by random sampling of the game state space (i.e., by random simulating the future game states). Figure 4 shows one iteration of executing the five actions iteratively:

- **Select**. Given a game state u, it selects child nodes that represent states leading to better overall outcome.
- **Expand**. If a selected node u_1 (the subscript denotes the according timestep) is not a terminal node, i.e. there exists at least one unoccupied square in u_1, it runs random simulations from u_1 to a terminal[3] node u_T.
- **Evaluate**. For each expansion, MCTS evaluates the terminal node u_T as the simulation result.
- **Backpropagate**. MCTS uses the simulation result to update statistics in the edges on the path from u_T to u_1.
- **Play**. MCTS estimates the quality of each child node of u using simulated statistics and then selects the best child node to make the move.

Specifically, in the simulation process, to record the statistical information, each node (u) in the tree contains edges (u, v) for all legal moves $v \in \mathcal{A}(u)$. Each edge (u, v) stores a set of statistics $\{N(u, v), W(u, v), Q(u, v), P(u, v)\}$ where $N(u, v)$ is the number of simulated times, $W(u, v)$ is the overall value of simulations, $Q(u, v)$ is the mean value of simulations and $P(u, v)$ denotes the probability to be selected.

Select Each simulation begins at a child node u_1 and finishes when the random simulation reaches a terminal node u_T. At each of these timesteps, $t < T$, the best potential child u_{t+1} from u_t is selected according to its statistics:

$$u_{t+1} = \arg\max_{v \in \mathcal{A}(u_t)} \left(Q(u_t, v) + cP(u_t, v) \frac{\sqrt{\sum_{v' \in \mathcal{A}(u_t)} N(u_t, v')}}{1 + N(u_t, v)} \right) \quad (1)$$

where c is a constant determining the level of exploration. From the perspective of STC, the *select* phase helps to select an unlabeled fragment and produce the most promising candidate label for it at each round.

Expand A node u_t is randomly expanded, and each edge (u_t, u_{t+1}) is initialized to $N(u_t, u_{t+1}) = 0, W(u_t, u_{t+1}) = 0, Q(u_t, u_{t+1}) = 0$ and $P(u_t, u_{t+1}) = \frac{1}{\mathcal{A}(u_t)}$. For STC, the *expand* phase simulates the future labeling of unlabeled fragments.

Evaluate Evaluating the terminal game state u_T aims to offer bonus for a player. For a good game, a proper bonus

[3]Different from the *final game-end node* (after playing), the *terminal node* denotes the simulated game-end state (in playing).

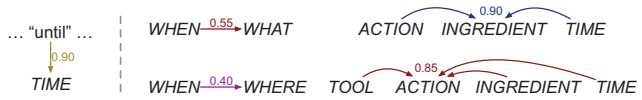

Figure 5: Examples of linguistic clues, including an emission feature, two first-order transition features and two high-order transition features. The observed probabilities are used as the confidences of the clues.

evaluation would push each player to find the optimal label for a certain unlabeled fragment at each round. In other words, the value of bonus should reflect whether the terminal game state (i.e., a labeling result) properly considers the intra-fragment and inter-fragment information. Here we mine two kinds of linguistic clues to evaluate the usage of intra-fragment and inter-fragment information, respectively. Specifically, for the intra-fragment information, we mine emission features which quantify the probability of a label conditioned on the text content of a fragment. Note that triggering an emission feature means that the corresponding fragment can be easily classified. For the inter-fragment information, we mine the k^{th}-order transition features which quantify the probability of a label conditioned on k labels (see Figure 5). Note that triggering a transition feature means the game state successfully captures a context dependency pattern. Apart from manually defined features, we additionally employ the chi-square test to obtain the significance of a linguistic clue in the training data (Sharma et al. 2018). We retain these distinguishing linguistic clues whose testing values exceed a threshold and use their corresponding observed probabilities as the confidences of the clues.

By injecting the linguistic clues into the game state evaluator, after many random expansions, each terminal node u_L can be quantitatively evaluated:

$$\xi(u_T, \mathcal{F}) = \sum_{m=1}^{n} \sum_{i=1}^{|e|} e_i(u_T(\psi_n(m))|F_m) + \sum_{k=1}^{|\mathcal{F}|-1} \sum_{j=1}^{|t^k|} t_j^k(u_T) \tag{2}$$

where $e(\cdot)$ and $t^k(\cdot)$ denote the emission features and the k^{th}-order transition features, respectively; $u(\psi_n(i))$ denotes the type of the piece located at the position $\psi_n(i)$ in u. In summary, the evaluation value denotes the cumulative confidence of all the triggered features, returning a larger value if more features are triggered.

Backpropagate The evaluated value of each terminal node u_T is then backpropagated. The edge statistics are updated in a backward pass from u_T to u_1 for the next decision. The simulation counts are incremented, $N(u_t, u_{t+1}) = N(u_t, u_{t+1}) + 1$, and the simulated value is updated to the mean value, $W(u_t, u_{t+1}) = W(u_t, u_{t+1}) + \xi(u_T, \mathcal{F})$, $Q(u_t, u_{t+1}) = W(u_t, u_{t+1})/N(u_t, u_{t+1})$. For STC, the *backpropagate* phase updates statistics for the decision at the next iteration.

Play At each round, MCTS selects the best child of u that leads to the "most victories" through many simulations. The overall confidence of a node is proportional to the product of its exponentiated simulated value and the prior probability of each placed piece:

$$\epsilon(u_1|u) = \frac{Q(u, u_1)^{\frac{1}{\tau}}}{\sum_{v \in \mathcal{A}(u)} Q(u, v)^{\frac{1}{\tau}}} \cdot \prod_{i=1}^{n} \pi\big(u_1(\psi_n(i))\big) \tag{3}$$

where τ controls the level of exploration and $\pi(a)$ is the prior probability of the piece a when given T_i obtained via the pretrained MLP module. For STC, the *play* phase produces the most confident label for an unlabeled fragment.

At each round, the search tree is reused at subsequent timesteps: the child node corresponding to the played move becomes the new root node, and the subtree below this child is retained along with all its statistics, while the remainder of the tree is discarded. Players iteratively place pieces until the game ends. At that point, the optimal sequential labels can be derived accordingly from the final game-end state u_*:

$$\hat{\mathcal{L}} = \langle u_*(\psi_n(1)), u_*(\psi_n(2)), \cdots, u_*(\psi_n(n)) \rangle \tag{4}$$

Heuristic Speedup Strategies

We now suggest some heuristic speedup strategies to further shorten the search time of the game without much performance penalty. 1) In the *select* phase, we only select one candidate move for extension even if the maximum (Equation 1) corresponds to multiple candidates. 2) In the *expand* phase, we only generate the child nodes whose neighbors contain at least one occupied square, to avoid unnecessary space search. We also expand a state to at most four deeper layers (i.e., an observable game state can be evaluated instead of expanding to a terminal state). 3) In the *evaluate* phase, we use memory-augmented MCTS (Xiao, Mei, and Muller 2018) to reduce the unneeded recalculations. 4) In chi-square testing, we randomly use 10% of the training data to test whether a linguistic clue is significant. If so, we retest it on all the training data.

Evaluation

We conduct extensive experiments to answer the following three research questions:

RQ1 Does our proposed approach, GuGo, outperform the currently state-of-the-art STC solutions?

RQ2 How do the different labeling orders and the speedup strategies affect performance?

RQ3 What are the main differences between jump labeling and successive labeling apart from the labeling order?

Datasets We use three real-world datasets of maintenance manuals (MAM), cooking recipes (COR) and customer reviews (WeBis) to test the methods in different domains.

- **MAM** (Qian et al. 2019) contains manuals from a wiki-based site[4] to teach people to fix various devices such as phones, cameras, cars, etc. For each manual, MAM contains the word-level labels (PERFORMER, ACTION, DEVICE and OTHER) that describe the semantic role of each word.

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

Table 1: Statistics of the datasets used. #D, #T and #C denote the number of text sequences, fragments and categories, respectively. @W/T denotes the the average number of words.

Dataset	Domain	Granularity	#D	#T	#C	@W/T
MAM	Maintenance	Word	2,636	20,605	4	1.00
COR	Cooking	Clause	1,005	2,636	5	5.41
WeBis	User Review	Sentence	3,097	10,660	3	22.84

- **COR** (Feng, Zhuo, and Kambhampati 2018) is a collection of user-generated recipes with textual descriptions of cooking procedures from a food-focused social network[5]. The clause-level labels (e.g., repairing tools, actions and empirical suggestions) are provided.
- **WeBis** (Chen et al. 2019) contains various consumer reviews[6] of movies, books, restaurants, etc. It is collected from Amazon, Yelp, YouTube and Google News. For every sentence in a review, WeBis has a sentiment polarity label (i.e., POSITIVE, NEUTRAL and NEGATIVE).

The statistics of the datasets are summarized in Table 1. These datasets are representative since they cover: 1) different domains; 2) various degrees of context dependencies (finer-grained fragments tend to need more context information); and 3) short and long samples, in which the average length varies from 1.00 (i.e., word-level fragments) to 22.84 (i.e., sentence-level fragments). For comparison, all datasets are divided into train/dev/test sets using an 8:1:1 ratio.

Metrics We follow Kim et al. (2019) and employ two harmonic metrics, *Macro-F_1* (MaF_1) and *Micro-F_1* (MiF_1) to report the performance of STC. MaF_1 is the average F_1-score of each category and is strongly influenced by the performance of categories with fewer documents. MiF_1 is the F_1-score over the whole dataset and depends on the performance of categories with a large number of documents.

Baselines We choose three-group representative baselines:

- **Single Text Classifiers.** This group of methods classifies each fragment separately. It includes a CNN based text classifier (**WordCNN**) that utilizes word-level convolution filters and multiscale region sizes (Zhang and Wallace 2017); and the state-of-the-art short text classifier (**RWMDCC**) that uses the semantic centroid distance in word mover's space (Li, Ouyang, and Li 2019).
- **Sequential Text Classifiers.** This group of methods solves STC with consideration of context information. It includes a parsing-based classifier (**PARPOS**) that uses the parse tree patterns and POS tags as text features (Yeung and Lee 2015); a two-layer RNN+CNN network (**TLRCN**) that incorporates context information (Lee and Dernoncourt 2016); and the state-of-the-art STC solution (**CNNDAC**) that uses a hierarchical CNN+RNN model for dialog act classification (Liu, Han, and others 2017).

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

[6]https://webis.de

Table 2: Experimental results of all methods on the three datasets. ▲ and △ indicate the best and the second-best performing baselines, respectively. The best performance among all methods is highlighted in boldface. $*$ means GuGo achieves significant improvement over the baseline ($p \leq 0.05$).

Methods	MAM		COR		WeBis	
	MaF_1	MiF_1	MaF_1	MiF_1	MaF_1	MiF_1
WordCNN	0.476_*	0.531_*	0.431_*	0.561_*	0.626_*	0.623_*
RWMDCC	0.430_*	0.560_*	0.457_*	0.598_*	0.703_*	0.704_*
PARPOS	0.529_*	0.689_*	0.622_*	0.726_*	0.786_*	0.786_*
TLRCN	$0.676_*^{\triangle}$	0.769_*	0.752_*	0.832_*	0.767_*	0.767_*
CNNDAC	$0.694_*^{\blacktriangle}$	$0.790_*^{\blacktriangle}$	$0.793_*^{\triangle}$	$0.903_*^{\triangle}$	0.802_*	0.802_*
SASLNN	0.570_*	0.654_*	0.697_*	0.775_*	$0.803_*^{\triangle}$	$0.803_*^{\triangle}$
BiLCRF	0.670_*	$0.774_*^{\triangle}$	$0.852_*^{\blacktriangle}$	$0.920_*^{\blacktriangle}$	$0.812_*^{\blacktriangle}$	$0.813_*^{\blacktriangle}$
GuGo	**0.764**	**0.832**	**0.886**	**0.952**	**0.835**	**0.834**

- **General Sequence Labelers.** This group of methods is designed for general sequence labeling. It includes an attention-based neural network (**SASLNN**) that can effectively capture the important semantic roles in a sequence (Tan et al. 2018); and the state-of-the-art general sequence labeler (**BiLCRF**) that employs a CRF layer to label general sequential data (Al-Zaidy, Caragea, and Giles 2019).

Implementation Details We use BERT (Devlin et al. 2019) as the language model and perform average pooling to obtain the fragment embeddings with dimension of 768. For the encoding layer, we also set its output dimension as 768. In the game-playing layer, we set a hard threshold (α=0.05) in chi-squared testing to select emission and transition features for game bonus evaluation (Equation 2). In order to obtain the prior probability of each fragment (Equation 3), we use a one-hidden-layer MLP (the size of hidden layer is 200) with ReLU as an activation function and the Adam optimizer (Kingma and Ba 2015) with learning rate 10^{-4}. The training process includes two main steps: 1) We pretrain the encoding layer and the MLP module with at most 5,000 epochs, a mini-batch size of 32 and the cross entropy as the loss function. Note that the parameters of the BERT language model are not updated; and 2) We use emission features, transition features and the prior probabilities obtained from the pretrained parameters to evaluate the game states for game playing. If not otherwise specified, all our proposed speedup strategies are employed. We implement GuGo via Python 3.7.3 and Pytorch 1.0.1. All of our experiments are run on a machine equipped with an Intel Core i7 processor, 32 GB of RAM, and an NVIDIA GeForce-GTX-1080-Ti GPU. We report the average performance over 5 different initiations and the results of two-tailed paired t-test (Dror et al. 2018) when necessary.

Performance Comparison (RQ1)

Table 2 presents the performance of the proposed GuGo and the compared baselines on the three datasets w.r.t. MaF_1 and MiF_1. From the table, we have several key observations:

- In all cases, our proposed model GuGo significantly outperforms all baselines. In particular, GuGo, compared

with the strongest baseline, improves MaF_1 by 7.0% and MiF_1 4.2% on MAM, and MaF_1 by 3.4% and MiF_1 by 3.2% on COR. These results validate the effectiveness of the proposed method.

- Specifically, GuGo outperforms BiLCRF that achieves relatively satisfying performance with first-order transition features considered in a left-to-right manner. The improvements could be attributed to a better usage of context information, indicating the use of jump labeling.
- GuGo achieves smaller performance improvements on the WeBis dataset as compared to the MAM and COR datasets. This is reasonable since the fragments in WeBis are at sentence-level which might require less context clues than the word-level (MAM) and clause-level (COR) scenarios (Lee and Dernoncourt 2016). Nevertheless, GuGo still outperforms all methods on WeBis, improving MaF_1 and MiF_1 by 2.3% and 2.1%, respectively. This is promising since the weak-dependent scenarios face the *context deficiency* problem (Lee and Dernoncourt 2016), which could harm the performance if the context information is not properly incorporated. The results verify that GuGo can facilitate STC regardless of the degrees of text granularities (or context dependencies).
- Among the baselines, the single text classifiers (WordCNN, RWMDCC) achieve worse performance than the other methods, which show the necessity of considering the context information among fragments. The sequential text classifiers (PARPOS, TLRCN, CNNDAC) perform slightly better than the single text classifiers, suggesting that the context encoding could help to capture inherent clues among fragments. However, the sequential text classifiers perform worse than BiLCRF in most cases, which shows the merit of explicitly considering context information in the labeling process.

Since GuGo achieves state-of-the-art results in terms of F_1-score and consistent performance across all datasets with varying text granularities, we conclude that the designed board-game and the jump labeling mechanism provide GuGo with high accuracy and good generalizability.

Jump Labeling vs. Successive Labeling (RQ2)

We investigate the effectiveness of jump labeling by comparing GuGo with its two variants that are imposed with the left-to-right (↻L2R) and right-to-left (↻R2L) direction restrictions for game playing. Such restrictions downgrade jump labeling to successive labeling, i.e., the variants that label the fragments in left-to-right and right-to-left order, respectively. The results in Table 3 show the performance of GuGo, GuGo↻L2R, and GuGo↻R2L on the three datasets.

We can observe that downgrading the jump labeling to successive labeling causes severe performance drops. For example, as compared to GuGo, GuGo↻L2R decreases the MaF_1 and MiF_1 by 4.3% and 2.5%, respectively, on COR. GuGo↻R2L behaves even worse and decreases the MaF_1 and MiF_1 by 11.8% and 10.4%, respectively, on COR. The main reasons include that 1) the long-history and the future-context information are not fully captured under successive labeling; 2) GuGo can predict the fragments that need less consideration of context information in advance, which enables it to provide bidirectional clues for those fragments that need more consideration of context information. Thus, jump labeling is beneficial to help sequence labeler choose a better labeling order.

Table 3: Performance comparison between GuGo and its two successive labeling variants. The best performance is highlighted in boldface. The statistical significance (two-tailed paired t-test) is indicated with $*$ ($p \leq 0.05$).

Variants	MAM		COR		WeBis	
	MaF_1	MiF_1	MaF_1	MiF_1	MaF_1	MiF_1
GuGo	**0.764**	**0.832**	**0.886**	**0.952**	**0.835**	**0.834**
GuGo↻L2R	0.712_*	0.794_*	0.843_*	0.927_*	0.814_*	0.814_*
GuGo↻R2L	0.669_*	0.761_*	0.768_*	0.848_*	0.781_*	0.782_*

Moreover, we investigate the computation overhead of jump labeling and the effects of the proposed speedup strategies. Figure 6 shows the average running time curves. We can conclude the following: 1) Vanilla jump labeling tends to spend much more time than successive labeling, the main reason is discussed in the following section. 2) The model with speedup strategies is much faster than that without speedup strategies. The speedup strategies are able to reduce the runtime by approximately three-quarter on average. As a result, it is nearly double or triple the time of successive labeling, instead of exponentially. This demonstrates that our proposed speedup strategies effectively shrinks the search spaces and avoids the state space explosion problem to some extent. 3) Although employing the proposed speedup strategies shrinks many search spaces, it does not cause much performance degradation. Therefore, we suggest applying the proposed speedup strategies in real-world applications where a fast response is required.

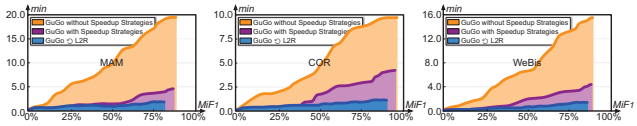

Figure 6: Time costs with/without using the speedup strategies. The point (x, y) on these curves indicates that the model has duration y when it first achieves the MiF_1 point x.

Discussion (RQ3)

To answer RQ3, we conduct a case study for intuitive analysis. A representative cooking recipe from COR is chosen, and the processes of jump labeling are shown in Figure 7. The example intuitively illustrates that jump labeling via board-game playing provides another two advantages:

- GuGo searches four unlabeled fragments and selects the most confident label `T` for the fifth fragment at Step 1. It then searches three unlabeled fragments and selects the second-most confident label `C` for the first fragment at Step 2. Generally, in board-game playing, when deciding

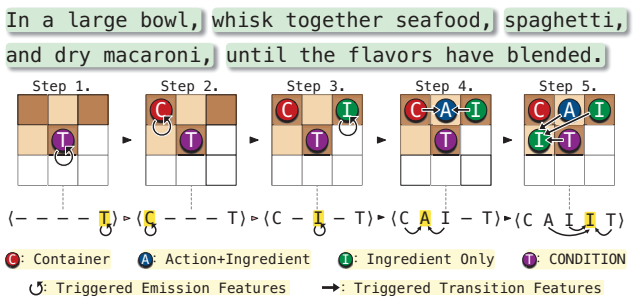

Figure 7: Case study: A representative example to explain why jump labeling via board-game playing is helpful.

the next move, a player simulates playing moves on all unoccupied squares and selects the best one to play for more bonuses. The action naturally corresponds to simulating predicting candidate labels for all unlabeled fragments in STC, i.e., the next prediction takes into consideration all possible unlabeled fragments, rather than only the successive one. In this way, in contrast with traditional successive labeling that searches only one successive unlabeled fragment at each step, GuGo enlarges the search spaces to all unlabeled fragments at each step.

- Owing to jump labeling, the fourth fragment (Step 5) takes the long-distance dependency from the second fragment and the backward dependency from the fifth fragment into consideration. Thus, GuGo is more confident in classifying the fourth fragment into `I` ("*dry*" as an adjective) instead of `A` ("*dry*" as a verb). Generally, successive labeling usually imposes the Markov assumption, which would leave out those right-to-left clues and those left-to-right clues beyond the assumptive scope. In contrast, jump labeling initially prefers to predict the fragments that need less advance consideration of context information, since there are no labeled fragments to consider at the beginning. However, it asymptotically prefers to predict the fragments that need more contexts, since there are enough labeled fragments as additional context clues, which makes it possible to provide bidirectional and farther context clues for the latter ones.

Conclusion and Future Work

We originally proposed the direction-free jump labeling mechanism. In addition, we designed a new board-game by defining proper game rules and suggesting some heuristic speedup strategies. The proposed approach obtains satisfying results in terms of the F_1-score, with better generalizability and effectiveness. We draw two main conclusions. First, compared with traditional successive labeling, jump labeling is more powerful by: a) enlarging the search spaces to choose a better labeling order, and b) equipping the ability to provide various degrees of context information for different fragments. Second, our designed game and speedup strategies are effective for problem transformation, enabling it to utilize game search to find the optimal labeling order with the avoidance of state space explosion problem.

In the future, we would like to formally prove that successive labeling is a linearized variant of jump labeling. We are also interested in applying jump labeling to classify sequential video frames and general sequential data.

Acknowledgments

We thank the three anonymous reviewers for their valuable suggestions. The work was supported by the National Nature Science Foundation of China (No.71690231, No.61472207), Tsinghua BNRist and NExT++ research supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@SG Funding Initiative.

References

Al-Zaidy, R. A.; Caragea, C.; and Giles, C. L. 2019. Bi-LSTM-CRF Sequence Labeling for Keyphrase Extraction from Scholarly Documents. In *the World Wide Web Conference (WWW)*.

Blatat, J.; Mrakova, E.; and Popelinsky, L. 2004. Fragments and Text Categorization. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Chen, Z.; Shen, S.; Hu, Z.; et al. 2019. Emoji-Powered Representation Learning for Cross-Lingual Sentiment Classification. In *the World Wide Web Conference (WWW)*.

Devlin, J.; Chang, M.-W.; Lee, K.; et al. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Dror, R.; Baumer, G.; Shlomov, S.; et al. 2018. The Hitchhiker's Guide to Testing Statistical Significance in Natural Language Processing. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Feng, W.; Zhuo, H. H.; and Kambhampati, S. 2018. Extracting Action Sequences from Texts Based on Deep Reinforcement Learning. In *the International Joint Conference on Artificial Intelligence (IJCAI)*.

Friedrich, A.; Palmer, A.; and Pinkal, M. 2016. Situation Entity Types: Automatic Classification of Clause-level Aspect. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Gildea, D., and Jurafsky, D. 2002. Automatic Labeling of Semantic Roles. In *Computational linguistics*.

Groote, J. F.; Kouters, T. W.; and Osaiweran, A. 2015. Specification Guidelines to Avoid the State Space Explosion Problem. In *Software Testing, Verification and Reliability*.

Ji, G., and Bilmes, J. 2005. Dialog Act Tagging using Graphical Models. In *the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Kalchbrenner, N., and Blunsom, P. 2013. Recurrent Convolutional Neural Networks for Discourse Compositionality. In *arXiv:1306.3584*.

Kim, K.-M.; Kim, Y.; Lee, J.; et al. 2019. From Small-scale to Large-scale Text Classification. In *the World Wide Web Conference (WWW)*.

Kim, S. N.; Cavedon, L.; and Baldwin, T. 2010. Classifying Dialogue Acts in One-on-one Live Chats. In *the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kingma, D. P., and Ba, J. L. 2015. Adam: A Method for Stochastic Optimization. In *the International Conference on Learning Representations (ICLR)*.

Kumar, H.; Agarwal, A.; Dasgupta, R.; et al. 2018. Dialogue Act Sequence Labeling Using Hierarchical Encoder with CRF. In *the AAAI Conference on Artificial Intelligence (AAAI)*.

Lee, J. Y., and Dernoncourt, F. 2016. Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks. In *the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Li, C.; Ouyang, J.; and Li, X. 2019. Classifying Extremely Short Texts by Exploiting Semantic Centroids in Word Mover's Distance Space. In *the World Wide Web Conference (WWW)*.

Liu, Y.; Han, K.; et al. 2017. Using Context Information for Dialog Act Classification in DNN Framework. In *the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ma, X., and Hovy, E. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Melamud, O.; Goldberger, J.; and Dagan, I. 2016. Context2Vec: Learning Generic Context Embedding with Bidirectional LSTM. In *the SIGNLL Conference on Computational Natural Language Learning (CoNLL)*.

Qian, C.; Wen, L.; Long, M.; et al. 2019. Extracting Process Graphs from Texts via Multi-Granularity Text Classification. In *arXiv:1906.02127*.

Quarteroni, S.; Ivanov, A. V.; and Riccardi, G. 2011. Simultaneous Dialog Segmentation and Classification from Human-human Spoken Conversations. In *the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Rabiner, L. R. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *IEEE*.

Ratnaparkhi, A. 1996. A Maximum Entropy Model for Part-Of-Speech Tagging. In *the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Santos, C. N. D., and Zadrozny, B. 2014. Learning Character-level Representations for Part-of-Speech Tagging. In *the International Conference on Machine Learning (ICML)*.

Sharma, R.; Bhattacharyya, P.; Dandapat, S.; et al. 2018. Identifying Transferable Information Across Domains for Cross-domain Sentiment Classification. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; et al. 2017. Mastering the Game of Go without Human Knowledge. In *Nature*.

Stolcke, A.; Ries, K.; Coccaro, N.; et al. 2000. Dialogue Act Modeling for Automatic Tagging and Recognition of Conversational Speech. In *Computational linguistics*.

Straffin, P. D. 1993. Game Theory and Strategy. In *The Mathematical Association of America*.

Tan, Z.; Wang, M.; Xie, J.; et al. 2018. Deep Semantic Role Labeling with Self-Attention. In *the AAAI Conference on Artificial Intelligence (AAAI)*.

Trischler, A.; Ye, Z.; Yuan, X.; et al. 2016. A Parallel-Hierarchical Model for Machine Comprehension on Sparse Data. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Venkataraman, A.; Ferrer, L.; Stolcke, A.; et al. 2003. Training a Prosody based Dialog Act Tagger from Unlabeled Data. In *the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Wang, S.; Mazumder, S.; Liu, B.; et al. 2018. Target-Sensitive Memory Networks for Aspect Sentiment Classification. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Wu, F.; Liu, J.; Wu, C.; et al. 2019. Neural Chinese Named Entity Recognition via CNN-LSTM-CRF and Joint Training with Word Segmentation. In *the World Wide Web Conference (WWW)*.

Xiao, C.; Mei, J.; and Muller, M. 2018. Memory-Augmented Monte Carlo Tree Search. In *the AAAI Conference on Artificial Intelligence (AAAI)*.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; et al. 2016. Hierarchical Attention Networks for Document Classification. In *the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Ye, Z.-X., and Ling, Z.-H. 2018. Hybrid semi-Markov CRF for Neural Sequence Labeling. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Yeung, C. Y., and Lee, J. 2015. Automatic Detection of Sentence Fragments. In *Annual Conference of the Association for Computational Linguistics (ACL)*.

Zhang, Y., and Wallace, B. C. 2017. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. In *arXiv:1510.03820*.

Zhou, G., and Su, J. 2002. Named Entity Recognition using an HMM-based Chunk Tagger. In *Annual Conference of the Association for Computational Linguistics (ACL)*.