Learning to Check Contract Inconsistencies

Shuo Zhang[1], **Junzhou Zhao**[1*], **Pinghui Wang**[1*], **Nuo Xu**[1],
Yang Yang[1], **Yiting Liu**[1], **Yi Huang**[2], **Junlan Feng**[2]
[1] MOE KLINNS Lab, Xi'an Jiaotong University, Xi'an 710049, P. R. China
[2] JIUTIAN Team, China Mobile Research
{zs412082986, l530111398}@stu.xjtu.edu.cn, {junzhouzhao, yang.yang95868}@gmail.com
phwang@mail.xjtu.edu.cn, nxu@sei.xjtu.edu.cn, {huangyi, fengjunlan}@chinamobile.com

Abstract

Contract consistency is important in ensuring the legal validity of the contract. In many scenarios, a contract is written by filling the blanks in a precompiled form. Due to carelessness, two blanks that should be filled with the same (or different) content may be incorrectly filled with different (or same) content. This will result in the issue of *contract inconsistencies*, which may severely impair the legal validity of the contract. Traditional methods to address this issue mainly rely on manual contract review, which is labor-intensive and costly. In this work, we formulate a novel *Contract Inconsistency Checking* (CIC) problem, and design an end-to-end framework, called *Pair-wise Blank Resolution* (PBR), to solve the CIC problem with high accuracy. Our PBR model contains a novel `BlankCoder` to address the challenge of modeling meaningless blanks. `BlankCoder` adopts a two-stage attention mechanism that adequately associates a meaningless blank with its relevant descriptions while avoiding the incorporation of irrelevant context words. Experiments conducted on real-world datasets show the promising performance of our method with a balanced accuracy of 94.05% and an F1 score of 90.90% in the CIC problem.

Introduction

A contract is a legally binding agreement that recognizes and governs the rights and duties of the parties to the agreement. Correctly composing contracts is crucial to ensure its legal validity. In many real-world scenarios, a standard contract is prepared by *filling blanks* in a precompiled form. Due to carelessness, two blanks that should be filled with the same (or different) content may be incorrectly filled with different (or same) content. This will result in *contract inconsistencies*, which may severely impair the legal validity of the contract.

Contract review is widely used by companies to check contract inconsistencies. However, contract review is labor-intensive and costly. Big companies have to hire tens of thousands of lawyers to conduct contract review, and it is estimated that Fortune Global 2000 and Fortune 1000 companies spend about $35 billion annually to review and negotiate contracts (Strom 2019). Therefore, it is desired to design methods to automate the contract review process.

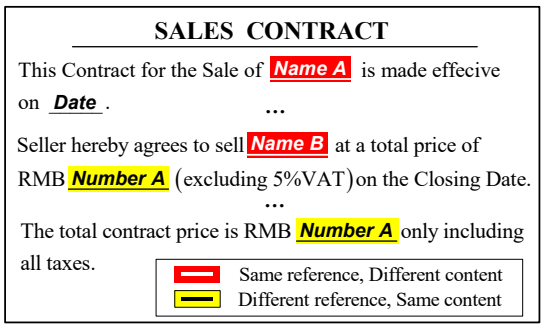


Figure 1: Examples of contract inconsistencies. The two blanks with red background refer to the same item for sale but incorrectly filled with different content. The two blanks with yellow background refer to two different prices (one with tax and the other without tax) but incorrectly filled with the same price.

In this work, we study the *Contract Inconsistency Checking* (CIC) problem, and propose an end-to-end model to solve the CIC problem with high accuracy. As far as we know, the CIC problem has not yet been studied in the AI community and no effective solution exists.

We consider contract inconsistencies caused by incorrectly filling blanks in a precompiled form. For example, in Figure 1, the two blanks with red background actually refer to the same item for sale, but incorrectly filled with different content. The two blanks with yellow background actually refer to two different prices (i.e., one with tax and the other without tax), but incorrectly filled with the same price. Our goal is to find methods that can automatically detect such contract inconsistencies with high accuracy.

A straightforward method to solve the CIC problem is to build a set of rules for each contract, and these rules regularize the allowed content for each blank. Rule-based methods have been used for other contract-related tasks such as checking obligation violations in contracts (Governatori and Milosevic 2005), verifying electronic contracts (Abdelsadiq, Molina-Jimenez, and Shrivastava 2011), and contract formalization (Joanni and Ratiu 2018). However, the rules have to be built manually by experts, and they do not scale well to diverse types of contracts.

Coreference resolution (CR) methods (Sukthanker et al.

*Corresponding Author

2020) aim to identify words or phrases that refer to the same real-world entity in a document. At first glance, CR methods may solve the CIC problem if we view blanks as words or phrases, and hence CR methods can be used to answer whether two blanks refer to the same concept. However, blanks are indeed not words or phrases which can be encoded using well studied word2vec methods, while blanks are meaningless empties and have no semantic meaning at all. Existing CR methods fail in modeling blanks in our case.

In this work, we propose a *Pair-wise Blank Resolution* (PBR) framework to solve the CIC problem with high accuracy. We formulate the CIC problem as a pair-wise binary classification problem. For a pair of blanks in the contract document, our PBR model adopts the Siamese architecture (Bromley et al. 1994; Reimers and Gurevych 2019) to encode each blank separately through the same blank encoder and then predict whether they should be filled with the same content or not. Blank modeling is challenging since it is hard to perform a semantic comparison to associate a meaningless blank to its relevant descriptions. In our PBR framework, we propose a novel `BlankCoder` that extends the Transformer (Vaswani et al. 2017) encoder architecture to address the above issue. `BlankCoder` adopts a two-stage modeling strategy where a blank is first initialized with the more related local context words and then updated by recurrently incorporating relevant information from the global context words. In this way, related descriptions of the blank would be fully introduced and the irrelevant ones would be ignored, yielding an informative blank representation.

To evaluate our method on the CIC task, we build a large-scale Chinese contract dataset of 281 contract documents including $299,621$ training samples (blank pairs). The English contract dataset for element extraction released by Chalkidis et al. (2017) is also used, and we view each element as a filled blank. The experimental results show that our method significantly and consistently outperforms all baseline methods with a promising performance of the balanced accuracy of 94.05% and F1 of 90.90%.

Our contributions are summarized as follows:

(1) We formulate the *Contract Inconsistency Checking* (CIC) problem. As far as we know, this problem has not yet been studied in the AI community.

(2) We propose a novel *Pair-wise Blank Resolution* (PBR) framework to address the CIC problem. In PBR, we propose a `BlankCoder` that extends the Transformer encoder architecture to efficiently model meaningless blanks.

(3) We collected and labeled a large-scale Chinese contract corpus for CIC. The experimental results show the promising performance of our PBR method.

Related Work

Our work is mainly related to three lines of recent researches: *automatic contract analysis*, *coreference resolution*, and *blank modeling*.

Existing automatic contract analysis methods mainly assist legal professionals by information extraction. Early ML-based attempts performed sentence-level classification on clause patterns (Indukuri and Krishna 2010) and service exceptions (Gao, Singh, and Mehra 2011). Recent DL-based methods focus on fine-grained intra-document classification of contract elements (Chalkidis et al. 2019), obligations and prohibitions (Chalkidis, Androutsopoulos, and Michos 2018), and insurance policies (Sun et al. 2019), and cross-document search of relevant clauses (Guo et al. 2020). Previous works do not perform a further comparison on the retrieved related sentences, leaving the detailed checking process to lawyers. In this paper, we automate the CIC process in a fully data-driven and end-to-end manner that significantly speeds up the manual review process.

Coreference resolution (CR) aims to identify the words or phrases (mentions) that refer to the same real-world entity. Existing methods can be classified into three broad categories of mention-pair, entity-mention, and ranking models. Mention-pair models (Wiseman et al. 2015) predict the coreference label for every two mentions. Entity-mention models (Clark and Manning 2016) directly model an entity by clustering mentions. Ranking models (Lee et al. 2017) were further introduced to model the degree of coreference. CIC can be view as a modified CR task that aims to identify the blanks that refer to the same content. However, CIC is much more challenging since the blanks are meaningless empties that can not be addressed with CR methods.

Blank modeling has been investigated in Text Infilling and Zero Pronoun Resolution (ZPR). In text infilling, a blank is usually treated as an out-of-vocabulary token and modeled with sequence models such as BiLSTM (Fedus, Goodfellow, and Dai 2018) and Transformer (Zhu, Hu, and Xing 2019). However, these methods encode a blank with all its context words that contain irrelevant noise descriptions. Similar to CR, ZPR aims to identify words that co-refer with a gap (an omitted pronoun). To encode the gap, Yin et al. (2016) designed a CenteredLSTM that focuses on the more related local words. In their later work (Yin et al. 2018), self-attention was further utilized to enhance the performance. Recently, Aloraini et al. (2020) adopted BERT (Devlin et al. 2019) to encode the gap with its nearest two words. Though able to avoid incorporating irrelevant descriptions, ZPR methods would negligent faraway relevant descriptions.

Contract Inconsistency Checking Problem

We formulate the *Contract Inconsistency Checking* (CIC) problem as a pair-wise binary classification problem, i.e., given a pair of blanks occurred in the contract document, we want to predict whether they should be filled with the same content or not.

Formally, let B denote a set of blanks in the contract document. Each blank $b \in B$ is included in a *surrounding sentence* $s = \{w_1, w_2, \ldots, b, \ldots\}$ consisting of $|s| - 1$ words and one blank[1]. Given two blanks $b_i, b_j \in B$, the CIC problem aims at predicting their *consistency relation* $r_{ij} \in \{0, 1\}$, with $r_{ij} = 1$ meaning that the two blanks b_i and b_j should be filled with the same content; and $r_{ij} = 0$ otherwise.

[1]If a sentence in the contract document contains multiple blanks, we construct the surrounding sentence s by deleting the other blanks and only keep the blank of interest and all the words in the original sentence.

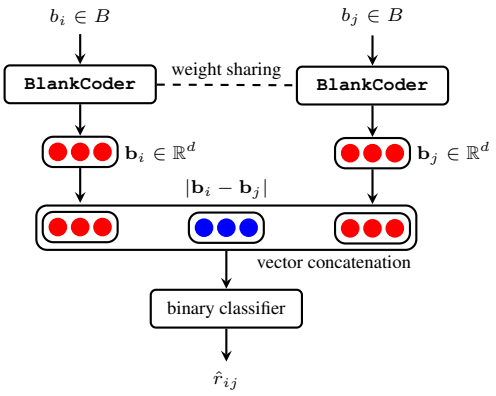


Figure 2: The PBR framework.

Pair-wise Blank Resolution Framework

In this section, we propose a *Pair-wise Blank Resolution* (PBR) framework to solve the CIC problem. The overall structure of our PBR framework is illustrated in Figure 2.

Overview of the PBR Framework

The PBR framework is inspired by *Coreference Resolution* (CR) (Sukthanker et al. 2020). CR aims at identifying words or phrases that refer to the same real-world entity in a document. However, as we explained previously, blanks are different from words or phrases because blanks have no semantic meaning at all. The key challenge for CIC is to represent the meaningless blanks in a contract document.

As illustrated in Figure 2, the proposed PBR framework adopts the Siamese architecture with a novel `BlankCoder` model to encode each blank, and predicts the consistency relation of the blank pair by a binary classifier. The proposed `BlankCoder` is a Transformer variant for blank modeling that could easily generalize to other tasks such as text infilling.

`BlankCoder`

The Transformer architecture (Vaswani et al. 2017) has shown a powerful semantic modeling capability on various natural language processing tasks including the relevant task of text infilling (Zhu, Hu, and Xing 2019). We extend the Transformer encoder architecture, and propose `BlankCoder` to address the challenge of modeling meaningless blanks.

In `BlankCoder`, in order to obtain a good blank embedding, each blank is made to fully utilize its context information in the surrounding sentence using *a two-stage modeling strategy*, i.e., the blank is first initialized based on the local context information (e.g., words in a local region of the blank), and then refined and updated based on the global context information (e.g., words outside of the local region in the surrounding sentence).

Figure 3 depicts the architecture of `BlankCoder`, which mainly consists of three modules: 1) *multi-head context attention*, which associate relevant context words, 2) *local visible pooling*, which initiates the blank representation with local keywords, and 3) *global update*, which refine and update the blank representation with related context words.

In the following, we focus on describing how to obtain the embedding of a blank $b \in B$.

Surrounding Sentence Representation Given a blank $b \in B$ and its surrounding sentence s consisting of n words and one blank (i.e., $|s| = n + 1$), we embed the i-th word $w_i \in s$ to a d-dimension vector $\mathbf{e}_i \in \mathbb{R}^d$. Embeddings $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$ are stacked up to form a matrix $S \in \mathbb{R}^{d \times n}$, which will be the representation of the surrounding sentence. Embedding vector $\mathbf{e}_i$ is the sum of three different embedding vectors capturing different semantic aspects.

The first embedding vector is the *word embedding*, denoted by $\mathbf{e}_i^{\text{word}} \in \mathbb{R}^d$, which is obtained by using a pre-trained word2vec model. Word embedding $\mathbf{e}_i^{\text{word}}$ only contains the linguistic information of word w_i.

The second embedding vector is the *positional embedding*, denoted by $\mathbf{e}_i^{\text{pos}} \in \mathbb{R}^d$, which incorporates the order information of word w_i in surrounding sentence s. We calculate $\mathbf{e}_i^{\text{pos}}$ using the same approach in the vanilla Transformer model (Vaswani et al. 2017).

The third embedding vector is the *segmentation embedding*, denoted by $\mathbf{e}_i^{\text{seg}} \in \mathbb{R}^d$, which distinguishes the preceding and following context words of the blank. This can be viewed that the blank separates the surrounding sentence into two segments, and each segment has one segmentation embedding. We adopted the learned segmentation embeddings as implemented in BERT (Devlin et al. 2019).

Following BERT, the final embedding of the i-th word $w_i \in s$ is calculated by $\mathbf{e}_i \triangleq \mathbf{e}_i^{\text{word}} + \mathbf{e}_i^{\text{pos}} + \mathbf{e}_i^{\text{seg}}$.

Multi-Head Context Attention To adequately incorporate relevant descriptions for blank modeling, it's essential to first model contextual information and correlate relevant words in the surrounding sentence. To this end, in the `BlankCoder`, we first adopt the powerful multi-head self-attention mechanism as implemented in the Transformer encoder for contextual encoding.

The Transformer encoder takes the embedding matrix S as input, and updates the embedding matrix layer by layer. The i-th layer outputs a matrix $H^{(i)} \in \mathbb{R}^{d \times n}$ by

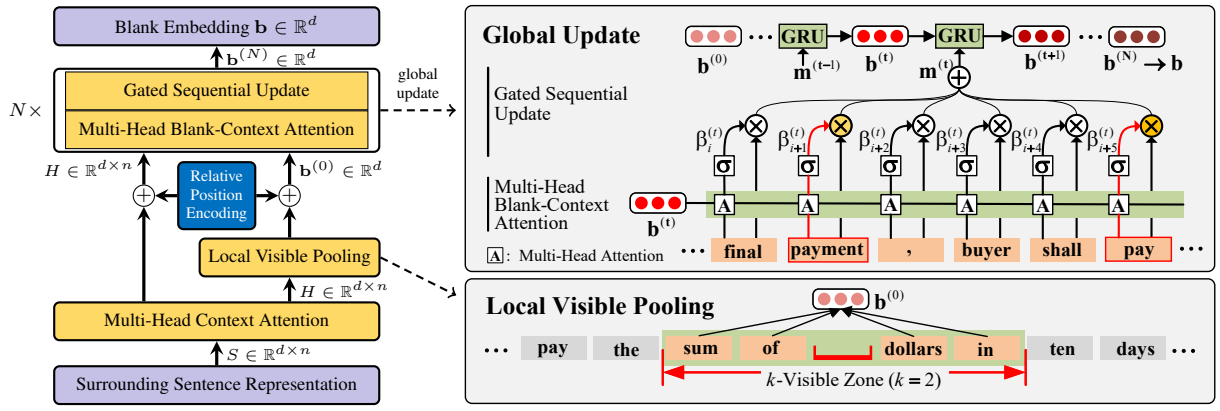
$$H^{(i)} = \texttt{FFN}(\texttt{MultiHeadAtt}(H^{(i-1)})), \quad H^{(0)} = S.$$

Here, `FFN` represents a feed-forward neural network, and `MultiHeadAtt` is the multi-head version self-attention, i.e.,

$$\texttt{SelfAtt}(S) = V \cdot \text{softmax}(\frac{K^\intercal Q}{\sqrt{d_k}})$$
$$Q = W_q S, \quad K = W_k S, \quad V = W_v S.$$

Here $Q \in \mathbb{R}^{d_k \times n}, K \in \mathbb{R}^{d_k \times n}$, and $V \in \mathbb{R}^{d \times n}$ denote the query, key and value matrix, respectively. $W_q \in \mathbb{R}^{d_k \times d}$, $W_k \in \mathbb{R}^{d_k \times d}$, and $W_v \in \mathbb{R}^{d \times d}$ are learnable parameters. We refer to (Vaswani et al. 2017) for more details on the Transformer encoder.

The output of the last Transformer encoder layer, denoted by $H \in \mathbb{R}^{d \times n}$, is a better representation of the surrounding



(a) `BlankCoder` (b) Local visible pooling and global update

Figure 3: The architecture of `BlankCoder`. It extends the Transformer encoder architecture to model meaningless blanks. First, we perform local visible pooling to build an initial blank embedding $\mathbf{b}^{(0)}$ with the more related local context information. Second, we perform global update to recurrently refine the initial blank embedding by gathering and injecting relevant descriptions from the global context words.

sentence, that incorporates the linguistic information, position information, and context information of the surrounding sentence. In the following, we discuss how to obtain the blank embedding using the surrounding sentence embedding H.

Local Visible Pooling Because a blank belongs to its surrounding sentence, a straightforward way to obtain the blank embedding is to sum the embeddings of all the words in the surrounding sentence, i.e., sum the columns of H. However, words in the surrounding sentence may have different importance for describing the blank, and simply summing them ignores their difference and may include too much noise.

Intuitively, the words that are close to the blank should be more related to the blank than words that are faraway in the surrounding sentence. Similar observation has been reported in other language modeling tasks (Yin et al. 2016; Aloraini and Poesio 2020). Therefore, we propose to sum embeddings of these closer words to obtain an *initial blank embedding*, which will be further refined. We refer to this step as *local visible pooling*, as illustrated in Figure 3.

Ideally, we want to find a vector $\boldsymbol{\alpha} \in \mathbb{R}^n$ such that it has larger values for words in a small *local region* around blank b and almost zero values for words outside of this region. Then, blank b's *initial embedding* $\mathbf{b}^{(0)} \in \mathbb{R}^d$ can be represented as

$$\mathbf{b}^{(0)} = H\boldsymbol{\alpha}.$$

We refer to $\boldsymbol{\alpha}$ as the *pooling weights*, and we discuss how to obtain pooling weights $\boldsymbol{\alpha}$ In the following.

We first formally define this local region. Assume blank b is between the i-th and $(i+1)$-th word in the surrounding sentence, $0 \le i \le n$.[2] We define the *k-visible zone* of blank b as the set of (at most) $2k$ closest positions around b, i.e.,

$$\text{zone}_k \triangleq \{\max(1, i-k+1), \ldots, \min(n, i+k)\}.$$

In our design, only words in this k-visible zone have non-zero pooling weights, and even words in the k-visible zone may have different pooling weights, depending on how informative the word could describe the blank. We propose a *masked self-attention* to calculate $\boldsymbol{\alpha}$, i.e.,

$$\boldsymbol{\alpha} = \text{softmax}(\tanh(WH)^\top \mathbf{q}_w + \mathbf{u}).$$

Here $\mathbf{u} \in \mathbb{R}^n$ is a mask vector and $\mathbf{u}_i = 0$ if $i \in \text{zone}_k$; $\mathbf{u}_i = -\infty$ otherwise. $\mathbf{q}_w \in \mathbb{R}^{d_w}$ and $W \in \mathbb{R}^{d_w \times d}$ will be learned during training. That is, surrounding sentence representation H is first fed to a one-layer perceptron to obtain its hidden representation. Then we measure the similarity between the hidden representation and vector $\mathbf{q}_w$. $\mathbf{q}_w$ can be viewed as a fixed query that queries important words over a set of words represented by the hidden representation. Therefore, our design will assign those words that are close and informative to the blank larger pooling weights than faraway and meaningless words in the surrounding sentence.

• **Remark**. The blank b now has an initial embedding $\mathbf{b}^{(0)}$, and words already have embedding matrix H. At this time, we suggest adding the *relative position embeddings* (RPEs) to these embeddings to incorporate the relative position information between words and the blank. RPEs are directly adapted from the PEs in Transformer: the blank's RPE is simply $\mathbf{PE}_0$; if two words have the same distance x to the blank, then the two words have the same RPE, i.e., $\mathbf{PE}_x$. We will slightly abuse the notations and still use $\mathbf{b}^{(0)}$ and H to denote the embeddings that have been added RPEs.

Global Update The local visible pooling step only uses blank b's k-visible zone in the surrounding sentence to obtain an initial embedding $\mathbf{b}^{(0)}$. Note that some words outside of the k-visible zone may be also informative and useful for

[2] Here $i = 0$ (or $i = n$) means that the blank is at the beginning (or end) of the surrounding sentence.

describing the blank. We propose a *global update* step that uses words in the entire surrounding sentence to further refine the initial embedding $\mathbf{b}^{(0)}$. After global update, blank embedding will incorporate more contextual information in the surrounding sentence, and hence will be more representative than $\mathbf{b}^{(0)}$.

The global update step refines initial embedding $\mathbf{b}^{(0)}$ in a recurrent manner, and we denote the blank embedding after t-th refinement by $\mathbf{b}^{(t)}$. In the following, we describe how to refine $\mathbf{b}^{(t)}$ and obtain $\mathbf{b}^{(t+1)}$, at the t-th global update step.

Given the current blank representation $\mathbf{b}^{(t)}$ and surrounding sentence embedding H, we want to find words in the surrounding sentence that are correlated to the blank. This can be achieved by computing a correlation score between the blank and a word, and we propose to compute the correlation score in the following way, i.e.,

$$\boldsymbol{\beta}^{(t)} = \sigma\left(\frac{K\mathbf{q}^{(t)}}{\sqrt{d_k}}\right) \text{ where } \mathbf{q}^{(t)} = W_{uq}\mathbf{b}^{(t)}, K = W_{uk}H$$

Here σ denotes a sigmoid function, $W_{uq} \in \mathbb{R}^{d_k \times d}$ maps vector $\mathbf{b}^{(t)}$ to a query vector, $W_{uk} \in \mathbb{R}^{d_k \times d}$ maps matrix H to a key matrix, and $\boldsymbol{\beta}^{(t)} \in \mathbb{R}^n$ is a scaled inner-product vector measuring the correlation score between the blank and each word in the surrounding sentence. In practice, we use a multi-head mechanism to calculate $\boldsymbol{\beta}^{(t)}$ which will capture similarities from different aspects, similar to Transformer.

$$\boldsymbol{\beta}^{(t)} = \texttt{MultiHeadScore}(\mathbf{b}^{(t)}, H)$$

Note that we do not normalize elements of $\boldsymbol{\beta}^{(t)}$ jointly by a softmax operation, as is used in Transformer. Instead, we normalize elements of $\boldsymbol{\beta}^{(t)}$ individually by a sigmoid operation because it is possible that none of the words in the surrounding sentence is informative for describing the blank. We refer to the above processing by *multi-head blank-context attention*.

Score vector $\boldsymbol{\beta}^{(t)}$ will be used as a memory gate to guard whether we should memorize the corresponding vector in H or not, i.e.,

$$\mathbf{m}^{(t)} = H\boldsymbol{\beta}^{(t)}$$

where $\mathbf{m}^{(t)} \in \mathbb{R}^d$ is the memorized embedding vectors in H and those memorized vectors are correlated to the blank. Hence $\mathbf{m}^{(t)}$ is considered to be informative for describing the blank and should be used to refine $\mathbf{b}^{(t)}$. To this end, we propose to apply the classic GRU (Cho et al. 2014) which takes $\mathbf{m}^{(t)}$ as input and $\mathbf{b}^{(t)}$ as hidden state, i.e.,

$$\mathbf{b}^{(t+1)} = \texttt{GRU}(\mathbf{m}^{(t)}, \mathbf{b}^{(t)}).$$

The above procedure will repeat N times. Thus the initial blank embedding $\mathbf{b}^{(0)}$ will be refined recurrently. We refer to the processing as the *gated sequential update*. The final output $\mathbf{b}^{(N)}$ will be considered as the final embedding of blank b, i.e., $\mathbf{b} \triangleq \mathbf{b}^{(N)}$. The multi-head blank-context attention operation and gated sequential update operation are illustrated in Figure 3.

Classifier

Armed with the elaborately designed `BlankCoder`, we are now able to predict the consistency relation between two blanks following the routing in Figure 2.

Given two blanks b_i and b_j, we first obtain their embeddings $\mathbf{b}_i$ and $\mathbf{b}_j$ using the `BlankCoder`, respectively. Then we concatenate $\mathbf{b}_i, \mathbf{b}_j$, and also a comparative term $|\mathbf{b}_i - \mathbf{b}_j|$ as the representation of the input blank-pair. We adopt a feed-forward neural network-based binary classifier to predict the consistency relation $\hat{r}_{i,j}$, i.e.,

$$\hat{r}_{i,j} = \texttt{FFN}([\mathbf{b}_i : \mathbf{b}_j : |\mathbf{b}_i - \mathbf{b}_j|]) \in [0, 1].$$

Here : denotes vector concatenation, $|\cdot|$ denotes element-wise difference, and `FFN` is a feed-forward neural network that the middle layer activation functions are ReLUs and the last layer activation function is sigmoid. The comparative term works as an implicit contrastive loss that forces consistent blanks to have similar representations and vice versa.

During training, we choose the Focal Loss (Lin et al. 2017) as the objective to address the class imbalance issue in the datasets.

$$\mathrm{FL}(\hat{r}_{i,j}, r_{i,j}) = \begin{cases} -\alpha(1-\hat{r}_{i,j})^{\gamma}\log(\hat{r}_{i,j}), & r_{i,j} = 1, \\ -(1-\alpha)\hat{r}_{i,j}^{\gamma}\log(1-\hat{r}_{i,j}), & r_{i,j} = 0. \end{cases}$$

Here, $r_{i,j} \in \{0, 1\}$ is the ground-truth consistency relation, α and γ are the balanced factor and focusing parameter of the focal loss, respectively.

Experiments

In this section, we conduct experiments on two real-world contract datasets to evaluate our method.

Contract Datasets

• **Chinese Contracts** We have collected 246 open source business contract templates and 35 real contracts from a company. These contracts are written in Chinese, and cover categories such as investment, lease, and labor. Annotation details of templates are reported in our technical appendix.

• **ICAIL Contracts** by Chalkidis et al. (2017). These contracts are written in English, and they were used for the task of tagging contract elements. We treat these extracted elements as filled blanks. The contracts are anonymized where the words are replaced with numbers for privacy concerns.

For each contract document, we collect all the blank pairs, and obtain their consistency relation by comparing the filled content. This allows us to build large training datasets. However, the class labels are imbalanced. For the Chinese Contracts, the ratio between positive samples and negative samples is $1:59$, and for the ICAIL Contracts, the ratio is $1:48$. The statistics of these datasets are summarized in Table 1.

contract dataset	# contracts	# blank-pairs	pos : neg
Chinese Contracts	281	299,621	1 : 59
ICAIL Contracts	1,526	67,765	1 : 48

Table 1: Data statistics

	Chinese Contracts						ICAIL Contracts					
model	AUC	accuracy	precision	recall	F1	MCC	AUC	accuracy	precision	recall	F1	MCC
BiLSTM	98.26	92.58	88.24	86.11	87.16	86.96	96.06	84.86	79.22	70.11	74.39	74.03
Transformer	94.45	88.51	79.47	77.36	78.40	78.04	93.37	83.09	74.36	66.67	70.30	69.82
Transformer-seg	97.15	92.32	85.15	86.90	86.01	85.78	95.26	83.99	77.78	68.39	72.78	72.40
CenteredLSTM	98.35	92.68	88.96	86.59	87.76	87.56	96.15	85.71	78.62	71.84	75.08	74.66
AttnLSTM	98.62	92.85	90.83	85.51	88.09	87.93	96.19	85.76	80.92	70.69	75.46	75.16
CorefBERT	93.06	90.20	53.85	81.58	64.87	65.61	–	–	–	–	–	–
PBR	**98.73**	**94.05**	**93.74**	**88.22**	**90.90**	**90.77**	**96.25**	**86.01**	**81.22**	**72.08**	**76.38**	**76.09**
%Improvement	0.11	1.29	3.20	1.88	3.19	3.23	0.06	0.29	0.37	0.33	1.22	1.24

Table 2: Model evaluation results (%). Note that the accuracy above stands for the balanced accuracy.

Settings

Evaluation Metrics. We use AUC, precision, recall, F1 score, *balanced accuracy*, and *Matthews correlation coefficient* (MCC) (Boughorbel, Jarray, and El-Anbari 2017) as evaluation metrics. Balanced accuracy is defined as the average of recall obtained in each class. MCC is a correlation coefficient between the predicted and ground truth binary classifications, and it has a value between -1 and $+1$, with $+1$ representing a perfect prediction. Note that because the datasets are highly imbalanced, we thus focus on comparing balanced measures: F1 and MCC.

Baselines We compare our method with the following approaches in text infilling and zero pronoun resolution.

- **BiLSTM** (Graves, Mohamed, and Hinton 2013) is a widely adopted bidirectional RNN in text infilling (Fedus, Goodfellow, and Dai 2018; Liu et al. 2019), where a blank is treated as an out-of-vocabulary token in the sentence and modeled together with the context words.
- **Transformer** (Vaswani et al. 2017) is a SOTA attention based language representation model. We adopt the vanilla Transformer encoder and perform blank modeling in the same way as in BILSTM.
- **Transformer-seg** (Zhu, Hu, and Xing 2019) is a segment-aware Transformer for text infilling, where the segments are obtained by splitting the sentence at the blanks.
- **CenteredLSTM** (Yin et al. 2017) is a zero pronoun modeling method that adopts two RNNs with one encoding the preceding context sequentially and the other encoding the following context reversely. A blank is represented with the concatenation of the two last hidden vectors.
- **AttnLSTM** (Yin et al. 2018) extends CenteredLSTM with a self-attention mechanism to generate segment representations that are concatenated to encode the blank.
- **CorefBERT** (Aloraini and Poesio 2020) is a zero pronoun modeling method that adopts a pre-trained BERT (Devlin et al. 2019) for contextual encoding and represents a blank by averaging the embeddings of its nearest two words.

Implementation Details We implement all the benchmarks using Pytorch on a server equipped with 2 Nvidia Tesla V100 GPUs, each with 32GB memory. For the Chinese Contracts, we adopt the Chinese word2vec embeddings released by Li et al. (2018). For the ICAIL Contracts, the word2vec embeddings are provided by Chalkidis et al. (2017). We perform the hyper-parameter search to find the best combinations for all the models. The details are shown in the appendix.[3]

Metrics	AUC	accuracy	precision	recall	F1	MCC
PBR	98.73	94.05	93.74	88.22	90.90	90.77
-no local	97.76	92.31	81.51	87.86	84.57	84.36
-no update	98.14	92.95	87.81	86.11	86.95	86.74
-no cmp	98.38	93.43	93.57	86.96	90.14	90.04

Table 3: Ablation analysis on the Chinese Contracts.

Model Evaluation

We compare the performance of different models, and show the results in Table 2. Note that CorefBERT is not applicable on the ICAIL Contracts due to data anonymization.

We observe that our PBR framework outperforms the others in terms of all evaluation metrics. Specifically, for balanced accuracy, PBR achieves $0.29 \sim 1.29\%$ improvements, for F1 score, PBR achieves $1.22 \sim 3.19\%$ improvements, and for MCC, PBR achieves $1.24 \sim 3.23\%$ improvements.

Compared to the vanilla Transformer encoder and its segment-aware version, our `BlankCoder` in PBR also has significant improvement. The balanced accuracy is improved by $1.87 \sim 6.26\%$, the F1 score is improved by $4.95 \sim 15.94\%$, and MCC is improved by $5.09 \sim 16.31\%$. This demonstrates the effectiveness of the two-stage semantic modeling approach in our `BlankCoder`.

We also observe a general performance decline of all the benchmarks and a limited performance gain of our PBR framework on the ICAIL Contracts. We attribute this to data anonymity and the ambiguous blank pair samples. In the ICAIL dataset, the sentences are not split to fit the task of tagging contract elements. Due to its anonymity, we have to specify a region as a pseudo-sentence, which would contain incomplete or redundant sentences that hinder the performance. Besides, ambiguous blank pairs where both the same and different contents are allowed can introduce extra noise that impairs the performance. Details are illustrated in our technical appendix due to space limitations.

[3]Codes available at https://github.com/ShuoZhangXJTU/CIC

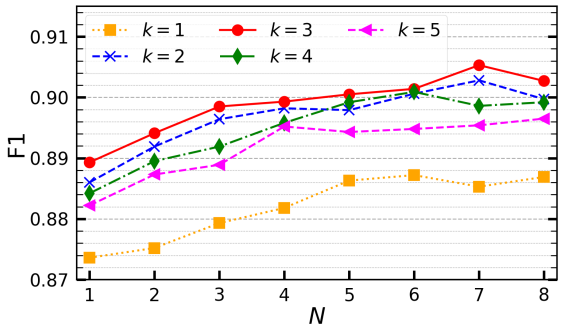


Figure 4: Hyper-parameter sensitivity

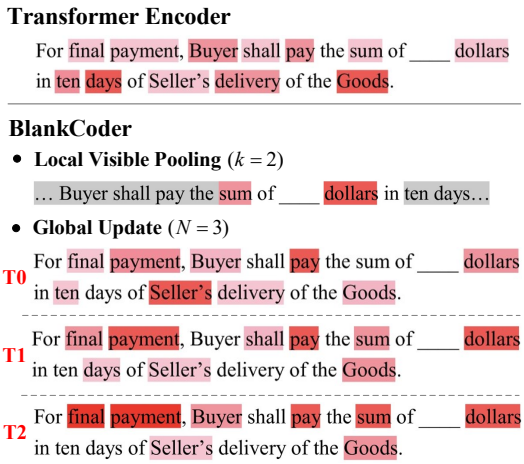


Figure 5: The attention visualization on a case example in a real-word sales contract.

Ablation Experiments

To further investigate the performance of the two-stage blank modeling strategy, we conducted ablation experiments on PBR (see Table 3). Similar results on the ICAIL Contracts are omitted because of space limitations.

To verify the effectiveness of the local visible pooling layer, we build a no-local version PBR (i.e., "-no local"), which directly assigns the whole sentence as the local visible zone. To evaluate the importance of the word-level semantic reasoning, we remove global update (i.e., "-no update"). To evaluate the effectiveness of the comparative term (i.e., $|\mathbf{b}_i - \mathbf{b}_j|$), we build a base PBR with no comparative term in the classifier (i.e., "-no cmp").

In Table 3, we observe that both the local visible pooling and global update are important components of the PBR framework. The local visible pooling is more critical since it determines if there exists noise information in the initial embedding. We also observe that the absence of the comparative term in the classifier leads to performance decrease, which implies the importance of contrastive supervision.

Hyper-parameter Sensitivity

In this section, we study how hyper-parameters in BlankCoder affect the performance, i.e., the length of local visible zone k, and the number of the stacked global update blocks N. The results on the Chinese Contracts are shown in Figure 4. We omit similar results on the ICAIL Contracts due to space limitations.

In Figure 4, we show five different lines where each line denotes the performance (F1) using different N and k. Fixing k, we observe that the stack of multiple global update blocks could consistently improve the performance, which demonstrates the effectiveness of the recurrent update process. Fixing N, we observe that the performance increases with the size of the local region at first and then decreases. This is due to the fact that a minimal local region does not contain enough related information, while a large local region would introduce irrelevant noise words. To reach better performance, a proper local visible zone shall be determined for the `BlankCoder`.

Case Study: Attention in `BlankCoder`

To intuitively show the effectiveness that PBR can extract representative features, we compare the attention of the Transformer encoder and our `BlankCoder`. For simplicity, we average the attention weights for all the Transformer encoder layers and omit the context attention layers of our `BlankCoder`.

Figure 5 shows a case example in a real-word sales contract with the blank referring to the final payment amount, where darker red shades indicate larger attention weights. For the Transformer encoder, we see that irrelevant descriptions like "days" and "delivery" are incorrectly highlighted, which would lead to biased blank representation that hinders further consistency checking. Whereas in our `BlankCoder`, the blank is first initialized with related local keywords (e.g., "dollars") and then recurrently updated with other relevant descriptions (e.g., "final" and " payment"), yielding an accurate and expressive blank representation without noise information.

Conclusion and Future Work

In this work, we formulate the *Contract Inconsistency Checking* (CIC) problem, an automatic contract analysis task with significant practical importance, and we propose a novel end-to-end *Pair-wise Blank Resolution* (PBR) framework to predict the consistency relation for every two blanks with high accuracy. In PBR, we extend the Transformer encoder architecture and propose `BlankCoder`, an off-the-shelf effective blank modeling method that could easily generalize to other tasks such as text infilling. Extensive experiments show that our model can significantly and consistently outperform existing baselines, yielding a promising balanced accuracy of 94.05% and an F1 score of 90.90%. In the future, we plan to consider more complex cases (e.g., ambiguous blank pairs) and explore more complex consistency checking scenarios that require logical reasoning.

Acknowledgments

The research presented in this paper is supported in part by National Key R&D Program of China (2018YFC0830500), National Natural Science Foundation of China (61922067, U1736205, 61902305), MoE-CMCC "Artifical Intelligence" Project (MCM20190701), Natural Science Basic Research Plan in Shaanxi Province of China (2019JM-159), Natural Science Basic Research Plan in Zhejiang Province of China (LGG18F020016).

References

Abdelsadiq, A.; Molina-Jimenez, C.; and Shrivastava, S. 2011. A High Level Model Checking Tool for Verifying Electronic Contracts. In *School of Computing Science Technical Report Series*.

Aloraini, A.; and Poesio, M. 2020. Cross-lingual Zero Pronoun Resolution. In *LREC*.

Boughorbel, S.; Jarray, F.; and El-Anbari, M. 2017. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLOS ONE* 12(6): 1–17.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1994. Signature verification using a siamese time delay neural network. In *NIPS*.

Chalkidis, I.; Androutsopoulos, I.; and Michos, A. 2017. Extracting Contract Elements. In *ICAIL*.

Chalkidis, I.; Androutsopoulos, I.; and Michos, A. 2018. Obligation and prohibition extraction using hierarchical rnns. In *arXiv:1805.03871*.

Chalkidis, I.; Fergadiotis, M.; Malakasiotis, P.; and Androutsopoulos, I. 2019. Neural Contract Element Extraction Revisited. In *NeurIPS Workshop Document Intelligence*.

Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

Clark, K.; and Manning, C. D. 2016. Improving coreference resolution by learning entity-level distributed representations. In *arXiv:1606.01323*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Fedus, W.; Goodfellow, I.; and Dai, A. M. 2018. MaskGAN: Better text generation via filling in the _. In *arXiv:1801.07736*.

Gao, X.; Singh, M. P.; and Mehra, P. 2011. Mining Business Contracts for Service Exceptions. *IEEE Transactions on Services Computing* 5(3): 333–344.

Governatori, G.; and Milosevic, Z. 2005. Dealing with Contract Violations: Formalism and Domain Specific Language. In *EDOC*.

Graves, A.; Mohamed, A.-r.; and Hinton, G. 2013. Speech recognition with deep recurrent neural networks. In *IEEE ASSP*.

Guo, H.; An, B.; Guo, Z.; and Su, Z. 2020. Deep Semantic Compliance Advisor for Unstructured Document Compliance Checking. In *IJCAI*.

Indukuri, K. V.; and Krishna, P. R. 2010. Mining E-contract Documents to Classify Clauses. In *COMPUTE*.

Joanni, A.; and Ratiu, D. 2018. Formalization of RAM Contracts for Advanced Consistency and Completeness Checking. In *ESREL*.

Lee, K.; He, L.; Lewis, M.; and Zettlemoyer, L. 2017. End-to-end neural coreference resolution. In *arXiv:1707.07045*.

Li, S.; Zhao, Z.; Hu, R.; Li, W.; Liu, T.; and Du, X. 2018. Analogical Reasoning on Chinese Morphological and Semantic Relations. In *ACL*.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *ICCV*.

Liu, D.; Fu, J.; Liu, P.; and Lv, J. 2019. TIGS: An inference algorithm for text infilling with gradient search. In *arXiv:1905.10752*.

Reimers, N.; and Gurevych, I. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *arXiv:1908.10084*.

Strom, R. 2019. This Startup Is Coming for $7 Billion in Contract Review. https://news.bloomberglaw.com/us-law-week/this-startup-is-coming-for-7-billion-in-contract-review, last accessed on 05/20/2020.

Sukthanker, R.; Poria, S.; Cambria, E.; and Thirunavukarasu, R. 2020. Anaphora and coreference resolution: A review. *Information Fusion* 59: 139–162.

Sun, L.; Zhang, K.; Ji, F.; and Yang, Z. 2019. Toi-CNN: A solution of information extraction on Chinese insurance policy. In *NAACL*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*.

Wiseman, S. J.; Rush, A. M.; Shieber, S. M.; and Weston, J. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *ACL-NLP*.

Yin, Q.; Zhang, W.; Zhang, Y.; and Liu, T. 2016. A deep neural network for Chinese zero pronoun resolution. In *arXiv:1604.05800*.

Yin, Q.; Zhang, Y.; Zhang, W.; and Liu, T. 2017. Chinese zero pronoun resolution with deep memory network. In *EMNLP*.

Yin, Q.; Zhang, Y.; Zhang, W.; Liu, T.; and Wang, W. Y. 2018. Zero pronoun resolution with attention-based neural network. In *ACL*.

Zhu, W.; Hu, Z.; and Xing, E. 2019. Text infilling. In *arXiv:1901.00158*.