applied sciences

MDPI

Article

Smart Contract Generation Assisted by AI-Based Word Segmentation

Yu Tong [1], Weiming Tan [1], Jingzhi Guo [1], Bingqing Shen [2], Peng Qin [1] and Shuaihe Zhuo [3,*]

[1] Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Macau 999078, China; yb87462@umac.mo (Y.T.); wade.tan@connect.um.edu.mo (W.T.); jzguo@umac.mo (J.G.); yb77428@connect.um.edu.mo (P.Q.)
[2] School of Software, Shanghai Jiao Tong University, Shanghai 200240, China; sunniel@sjtu.edu.cn
[3] School of Business, Macau University of Science and Technology, Macau 999078, China
* Correspondence: shzhuo@must.edu.mo

Abstract: In the last decade, blockchain smart contracts emerged as an automated, decentralized, traceable, and immutable medium of value exchange. Nevertheless, existing blockchain smart contracts are not compatible with legal contracts. The automatic execution of a legal contract written in natural language is an open research question that can extend the blockchain ecosystem and inspire next-era business paradigms. In this paper, we propose an AI-assisted Smart Contract Generation (AIASCG) framework that allows contracting parties in heterogeneous contexts and different languages to collaboratively negotiate and draft the contract clauses. AIASCG provides a universal representation of contracts through the machine natural language (MNL) as the common understanding of the contract obligations. We compare the design of AIASCG with existing smart contract generation approaches to present its novelty. The main contribution of AIASCG is to address the issue in our previous proposed smart contract generation framework. For sentences written in natural language, existing framework requires editors to manually split sentences into words with semantic meaning. We propose an AI-based automatic word segmentation technique called Separation Inference (SpIn) to fulfill automatic split of the sentence. SpIn serves as the core component in AIASCG that accurately recommends the intermediate MNL outputs from a natural language sentence, tremendously reducing the manual effort in contract generation. SpIn is evaluated from a robustness and human satisfaction point of view to demonstrate its effectiveness. In the robustness evaluation, SpIn achieves state-of-the-art F1 scores and Recall of Out-of-Vocabulary (R_OOV) words on multiple word segmentation tasks. In addition, in the human evaluation, participants believe that 88.67% of sentences can be saved 80–100% of the time through automatic word segmentation.

Keywords: smart contract; collaborative drafting; semantic understanding; automatic word segmentation

Citation: Tong, Y.; Tan, W.; Guo, J.; Shen, B.; Qin, P.; Zhuo, S. Smart Contract Generation Assisted by AI-Based Word Segmentation. *Appl. Sci.* **2022**, *12*, 4773. https://doi.org/10.3390/app12094773

Academic Editor: Valentino Santucci

Received: 4 April 2022
Accepted: 29 April 2022
Published: 9 May 2022

1. Introduction

In the past decade, the blockchain smart contract has transcended beyond its initial application to the finance sector. It provides automatic, immutable, traceable, and decentralized solutions to various real-world applications, including healthcare data sharing [1], supply chain management [2], Internet of Things (IoT) services [3], and energy management [4]. Despite its prevalence, most existing blockchain smart contracts are directly represented in programming languages of different levels of abstraction [5]. Thus, contracting parties without knowledge of smart contract languages may struggle to read and understand the smart contracts. Furthermore, the complexity of the contractual content may not be fully understood by the programmers implementing the smart contract, leading to extra negotiation and revision about involved entities in the smart contract and decreasing the smart contract development efficiency.

Moreover, considering the drafting of legal contracts in the real world, participants from heterogeneous contexts (e.g., stakeholders, lawyers, and external service providers)

and speaking different languages are involved, raising the concerns of semantic consistency and common consent.

1. Semantic consistency: because contracting parties are in different contexts, they may perceive the terms in a contract clause differently, i.e., the sense of ambiguity in contract content. For example, the word "refrigerator" can mean either a household appliance for preserving foods at low temperature, or customized cooling devices or spaces to store specific objects, such as biological samples.
2. Common consent: because different languages contain heterogeneous grammar rules and unique grammar features, the translation of a contract clause between two languages cannot always preserve its original meanings completely. The information lost during translation and the sense of ambiguity of contract terms can lead to disputes regarding the future execution of a contract.

As the preliminary work of this paper, *Intelligible Description Language Contract* (IDLC) was proposed as a new smart contract paradigm, using the formal language representation *Machine Natural Language* (MNL) to guarantee semantic consistency and common consent [6,7]. Nevertheless, IDLC requires manual input of the smart contracts written in MNL sentences: the terms in each sentences are selected one-by-one from a common dictionary (CoDiC). However, for contracts written in natural language without explicit boundaries between words, such as Chinese, IDLC requires editors to first split the sentence into a sequence of words. The well-segmented words are then converted to terms predefined in CoDiC, and manually input to generate an MNL sentence. IDLC demands considerable manual effort and lacks efficiency when drafting complex contracts. Furthermore, manual contract generation is error-prone when one is fatigued or distracted.

In this paper, we explore an alternative to IDLC's manual contract generation. We consider the contract generation problem as to how to quickly and automatically construct a sentence (i.e., a contract obligation) that requires minimal manual adjustment by contract editors on word meaning and sentence grammar. Grounded on MNL and IDLC, the overall methodology is to automate the process of converting contract clauses written in arbitrary natural languages into MNL sentences, which are interpretable by smart contract code scripts and understandable by a human. Particularly, a critical issue is the correct identification of terms and their senses which are the building blocks of MNL sentences. Utilizing artificial intelligence (AI), a novel technique called *Separation Inference* (SpIn) is introduced to perform automatic word segmentation (WS) for natural language contracts, which is a core supporting tool in AIASCG that facilitates efficient and effortless smart contract generation.

The main contributions of this paper are as follows:

1. We proposed AIASCG as a novel smart contract generation paradigm allowing unambiguous, collaborative drafting of document-based contracts that can be readily used by IDLC;
2. We designed SpIn to perform WS and make the execution of IDLC efficiently on natural language documents;
3. We rigorously evaluated the robustness of SpIn in multiple languages and measured its usefulness through human evaluations.

The remaining of the paper is organized as follows: Section 2 first briefly introduces the evolution of smart contracts and then presents existing WS techniques. Section 3 introduces the fundamental research on which AIASCG is grounded, including MNL and IDLC. Section 4 presents the components and workflow of AIASCG and qualitatively compares it with other smart contract generation approaches. Section 5 introduces the word segmentation mechanism and network model of SpIn, while Section 6 evaluates SpIn in multiple experiments. Lastly, Section 7 concludes the paper.

2. Related Works

2.1. Smart Contract Generation

Before discussing smart contract generation, we briefly introduce the concept of the smart contract. The smart contract was originally proposed by Nick Szabo [8], who considered that legal contract clauses can be embedded into computer hardware and software, such that obligations of the contract were automated and bleach-proof. In this sense, a smart contract was an extension of electronic contracts (e-contracts) that emphasized automatic contract execution [9]. Meanwhile, the legal binding concept in Szabo's design was also elaborated in many follow-up works. For example, the Ricardian Contract was a legal, digital, and verifiable contract system for issuance with the capability for human understanding and code execution [10]. Smart contract templates were proposed to convert legal documents into program codes in [11]. Borrowing the concepts of smart contract codes and smart legal contracts from [12], the authors argued that the operational aspect of a legal contract can always be automated as smart contract codes. The extracted operational parameters and the legal prose were the core of a smart contract template.

Although Szabo conceptualized many aspects of the smart contract in [8] and related research, the smart contract remained a theoretical artifact until the emergence of blockchain in the past decade. Blockchain-based smart contracts were collections of agreements written in code scripts and deployed on blockchain platforms, which were automatically executed upon the arrival of mandatory data [13]. A blockchain smart contract can be written in two types of high-level programming languages: imperative and declarative languages [14]. Imperative languages were more common in existing blockchain smart contract systems. Although they can implement fairly complex operations, they were not straightforward for human reading and understanding. Meanwhile, declarative languages were like rules and logic, so they are more intuitive for human interpretation. Various studies had advocated its adoption into the existing blockchain ecosystem [9,14].

Smart contract generation is the initial step of the life cycle of a smart contract consisting of generation, deployment, and execution [5,9]. Four types of generation approaches had been observed in the literature: arbitrary constraints to the smart contract, formally specified constraints to the smart contract, business processes to the smart contract, and legal agreements as smart contracts. Their differences lie mainly in the choices of smart contract representations and the levels of formalism.

Arbitrary Constraints to Smart Contracts (AC-SC) are the most primitive type of smart contract generation, relying on a programmer to read and understand the smart contract requirements in arbitrary formats, then design, implement, and test the corresponding smart contracts. Differently, *Formally specified Constraints to Smart Contracts* (FC-SC) can achieve (partially) automatic smart contract generation, because the contractual constraints were usually presented in a formal and structured format, e.g., via domain-specific ontologies and semantic rules [15]. Therefore, well-defined transformation rules, such as a domain-specific language (DSL) [16], can be applied. The generated results were usually smart contract skeletons that still depended on the manual implementation of the core logic. These two types of approaches usually produce smart contracts written directly in a programming language.

By contrast, *Business Processes to Smart Contracts* (BP-SC) and *Legal Agreements to Smart Contracts* (LA-SC) utilized logical smart contracts as an intermediate representation, achieving the separation of a smart contract's specification and implementation. The smart contracts were generated from the constraints of business process [17,18] and legal agreement [11,19], respectively, which were often too complex to be directly encoded as program codes but suitable to be described logically. Meanwhile, their implementations mostly relied on FC-SC to produce computer-executable smart contract codes.

While these generation methods have their own merits and use cases, the aforementioned approaches relied heavily on human effort in smart contract generation, e.g., the prepared smart contract codes in AC-SC or the formal representation in FC-SC, BP-SC, and LA-SC. Moreover, they did not recognize how AI can be incorporated into the human

editing process to enhance smart contract generation efficiency. Furthermore, they did not discuss the collaboration process during smart contract generation, which may involve participants from heterogeneous contexts. In this case, effective mechanisms to facilitate the universal understanding of a smart contract should be provided.

2.2. Word Segmentation

Generally, word segmentation was the ability to recognize the sense of words in context in a computational manner [20]. WS can be described as: given a sentence $S = \{t_1, t_2, \ldots, t_n\}$, determined the sense(s) of token $t_i \in S$. For languages such as Chinese and Japanese, t_i usually consisted of multiple atomic units, such as Chinese characters .

WS is essential for many applications in natural language processing (NLP), including information retrieval, statistical machine translation, question answering, and so on [21–25]. It is also fundamental to the interoperation of heterogeneous information in the business process [26].

Segmentation is particularly for languages that lack explicit boundaries between characters, such as Chinese, Japanese, and Korean. Current dominant algorithms considered word segmentation as the sequence-labeling task. Maximum Entropy (ME) models [27] and Conditional Random Fields (CRF) [28,29] models were leveraged as the tagger. Various tagging schemas were explored, such as BMES [30] (Begin, Middle, End, Single), BIES (Begin, Inside, End, Single) [31], SEP-APP (Separate, Append) [32], and SC (Separation, Combination) [33]. Despite the notations varying in different tagging schemas, the goal of describing each atomic unit's position in a word segment is similar. Moreover, rich features were exploited for achieving more accurate segmentation results. Word features [34] and bigram features [29,32,35–37] were well-perceived to be two of the most common and effective features. Besides the bigram and word features mentioned above, language-specific features were utilized [38] to obtain accurate segmentation results. Moreover, the extra information was leveraged through the semi-supervised models to achieve better segmentation results [37–40]. With the development of versatile pre-trained models, such as BERT [41], the performance of the WS task is boosted and the cost of feature engineering is significantly reduced. Based on powerful feature representations, more WS algorithms were proposed [42,43] that focused on improving the structure of the network.

Although these methods improve the performance of WS, there is a critical issue with the existing tagging schemas. For the widely applied BMES and BIES tagging schemas, there is a restriction of tag-to-tag transition. For example, tag "B" cannot be transferred to tag "B" or "S". Moreover, for SEP-APP and SC, the first character in a sentence must be predicted as "SEP" or "Separation". Although the rich features and the improved network structure improved the performance, the inherent problems of the existing tagging schemas, the restriction of tag transition, and the implicit constraint for the first tag were not well solved.

3. Preliminary Work

3.1. Machine Natural Language (MNL)

In [6], *Machine Natural Language* (MNL) is proposed to address two critical issues in semantic document exchange:

1. Semantic consistency across users in heterogeneous contexts;
2. Universal representation of complex documents.

Specifically, semantic consistency considers both concept semantic consistency and document semantic consistency. The former implies that the sense of a concept in a contract, which is either a word or multiple adjacent words expressing a meaning unitedly, can be ambiguous as explained by the example word "refrigerator" in Section 1. The latter means that different languages usually have different sentence grammar and contain unique grammatical features, and thus, certain information embedded in the grammatical structure may be lost during the translation between two languages. For universal document representation, MNL not only handles the grammatical diversities of different languages,

but also addresses the limitations of preceding research, that is, the limited capacity to process complex sentences dealing with table-based document exchange [44].

In MNL, concept semantic consistency is addressed by the *Collaborative Signs Dictionary* (CoDic) [45], while the *Universal Case Grammar* (UGC) can ensure document semantic consistency by parsing sentences written in different languages into a language-agnostic representation.

3.1.1. Collaborative Signs Dictionary (CoDic)

CoDic is inspired by Saussure's dyadic sign model [46] and Peirce's triadic sign model [47]. The design of a sign in CoDic guarantees that it is human and computer-readable and understandable [48]. Furthermore, by using collaborative editing systems [45] as the common contexts, a common concept of a sign can be interpreted by interpreters from heterogeneous contexts.

Each sign is expressed as a concept in the CoDic. A concept is a word or a phrase corresponding to an entity in reality, with the corresponding Part-of-Speech (PoS) tag encoded, so that the case label can be derived to construct MNL sentences using the UCG. Moreover, any PoS is a special sign defined in CoDic. Internally, each sign is indexed by an *Indexical Identifier* (IID) in CoDic, which is a compact binary coding schema with each bit containing information to index a concept.

With CoDic, a natural language sentence can be converted into a *Human Machine Language* (HML) form, where a sentence is a sequence of IIDs:

$$\text{HML}_i := (iid_0, iid_1, \ldots, iid_n)$$

where i indicates a particular natural language. This conversion is an essential process in MNL-mediated collaborative editing.

3.1.2. Universal Case Grammar (UCG)

UCG for MNL is rooted in the classical grammar theories, such as Fillmore's case grammar theory [49] and Chomsky's universal grammar theory [50], with the emphasis on processing natural language for both human understanding and machine execution. In the UCG, each concept contains an intrinsic case and an extrinsic case. The former encodes the PoS tag of the concept, whereas the latter specifies how the PoS tag combines with other PoS tags to generate a meaning group. Furthermore, HML bundles with corresponding HML-MNL grammar rules to readily convert a sentence input between these two forms. The grammar rules have two purposes:

1. Label each sign in an HML input with a corresponding case to indicate its grammatical roles;
2. Determine each sign's center-modifier relations with every other sign in the same HML input.

By applying UCG, an HML is transformed into its MNL form:

$$\text{MNL} := (S, eiid_0(eiid_{00}, eiid_{01}, \ldots, eiid_{0k}), \ldots,$$

$$eiid_n(eiid_{n0}, eiid_{n1}, \ldots, eiid_{nk}))$$

where *eiid* stands for *extended iid* with additional information of the case, modifying relations, etc., S indicates the MNL sentence type, and $eiid_i(eiid_{i0}, \ldots, eiid_{ik})$ indicates the center-modifier relations between signs. Thus, for two parties residing in different contexts, MNL serves as an intermediate representation to facilitate the unambiguous translation of documents between them.

3.2. Intelligible Description Language Contract (IDLC)

IDLC [7] is a new smart contract paradigm achieving two objectives: (1) Human readable and editable smart contracts; (2) Common consent on smart contracts. The core

concept of IDLC is the *Supervised Sentence Contract* (SSC) consisting of MNL sentences, which resolved: (1) ambiguous descriptions of contract executions in contract clauses; (2) ambiguous definitions of temporal constraints in contract clauses; (3) lack of cross-reference mechanism among contract clauses.

In [7], the representation and execution models of IDLC are proposed. For representation, the obligation syntax defines an MNL-based model covering the critical information in a legal contract, such as contracting parties, contract objects, required activities, and so on. Using MNL, a common understanding of a contract clause for contracting parties in heterogeneous contexts is achieved. Furthermore, the obligation syntax ensures that the legal binding of a contract is well captured. For execution, the obligations listed in the SSC are executed sequentially according to predefined constraints. When the verified set of evidence of an obligation is submitted by contracting parties, corresponding smart contract codes will be triggered. To guarantee the integrity and traceability of contract executions, the hash values of a digitally signed SSC, its chained obligation evidence, and the corresponding smart contract codes are stored on a blockchain. Furthermore, the verification and management of obligation evidence are assumed to be conducted in off-chain systems to ensure truthful obligations.

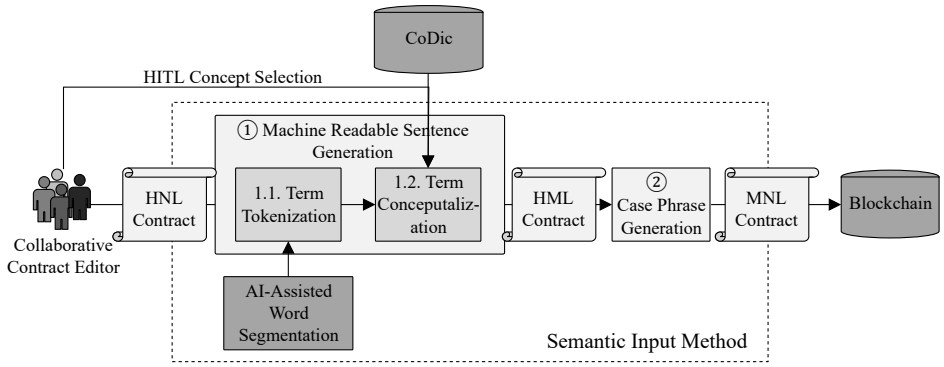


Figure 1. The structure of the AIASCG framework.

4. Ai-Assisted Smart Contract Generation Approach (AIASCG)

4.1. Framework Overview

The proposed AI-Assisted Smart Contract Generation Approach (AIASCG) is a novel solution to fast and reliable smart contract generation for IDLC. It facilitates collaborative contract drafting by contracting parties, who use MNL to translate between natural languages to unambiguously negotiate the content. As shown in Figure 1, AIASCG consists of two processes: machine-readable sentence generation and case phrase generation. While they inherit the MNL sentence generation processes introduced in [6], AIASCG automates the original design with AI-assisted WS, avoiding manually keying in natural language text repeatedly. In this manner, AIASCG follows the so-called *human-in-the-loop* (HITL) practice [51], allowing contracting parties to supervise and modify the computer-generated outputs when necessary with minimal manual efforts. A typical workflow of AIASCG is depicted in Figure 1.

First, contracting parties from heterogeneous contexts prepare the contract clauses for negotiation in their languages, producing a human natural language (HNL) contract, whose content is a list of clauses.

Then, the machine-readable contract generation will convert the HNL contract into an HML contract. Specifically, the HNL contract is first tokenized clause-by-clause. With AI-assisted WSD, a clause is automatically split into a collection of tokens called terms, the sense is predicted by the AI model from potential sense. Then, by referring to the CoDic, the terms are converted into concepts, whose definitions follow the specifications in [45,48]. Notably, the automatic generation output of the AI-assisted generation process is only the

prediction of HNL–HML conversion, which the contracting parties can examine and adjust so that the semantic consistency between the HNL and HML contract is maintained.

Finally, the HML contract will be translated into an MNL contract by case phrase generation, and the MNL contract can be translated to HNL language in different languages by the MNL translation rules so that the contracting parties in different languages can understand others' contract clauses and collaboratively revise them. The process of case phrase generation and MNL translation have been discussed in [6]. Once the final version of a contract is universally approved, its MNL form, i.e., the SSC, will be published to a blockchain for execution. The execution model is beyond the scope of this paper and has been presented in [7].

The machine-readable sentence generation and AI-assisted WS are wrapped into the *Semantic Input Method* (SIM), which provides feedback and control flows, allowing the users to supervise the entire process. Currently, the inaugural version of SIM has been implemented in [52], and the next update is in progress to adapt to MNL specifications. Moreover, the languages supported by AIASCG depend on the vocabularies supported by CoDiC, which currently supports Chinese and English.

Table 1 illustrates different representations of a smart contract produced in the AIASCG approach for a fragment of a procurement contract. Firstly, an HNL contract consists of contract clauses written in natural language, i.e., English in this example. Each clause is the execution conditions of activity for contract fulfillment. For example, the two clauses in Step 1 of Table 1 define the conditions of two activities: shipment of the contractual goods by the seller and notification of the shipment with supporting documents from the seller to the buyer. Secondly, an HML contract is produced by tokenization and conceptualization. The tokenization process segments a clause into multiple segments carrying semantic meaning. As the core component in AIASCG, a solution based on AI-assisted word segmentation model will be introduced in Section 5. Then, the conceptualization process searches for the best-match concept of each segment in CoDic and reconstructs the HML contract clause with the matched concepts in CoDic. The conversion from an HNL contract to an HML contract eliminates the sense of ambiguity of words in the original contract document, because each concept in CoDic is unambiguously defined in meaning, sense, and part-of-speech. Furthermore, the HML contract can be understood by both humans and computers, because the indexing of concepts in CoDic follows predefined and well-structured parsing rules [48]. Lastly, an MNL contract is generated from an HML contract by case phrase generation rules designed in [6]. An MNL contract serves two purposes in AIASCG. On the one hand, it is an intermediate representation that facilitates an HNL contract to translate into other natural languages while preserving the semantics. On the other hand, it is understandable by computers with its internal representation of eIID; therefore, it can be parsed into executable codes on a blockchain platform.

Table 1. An example of different representations for a contract fragment.

Step	Contract Format		Contract Example
1.	HNL contract (clause-based)		[Clause Y.1] Seller shall ship contract object based on Clause X of shipment on or before 15 August 2019 at Shanghai port and deliver to the destination port of Hong Kong. [Clause Y.2] Seller shall notify buyer the shipment details including shipment date, shipping company, vessel name, and the digital hash of the original digitally signed bill of lading within 24 h after the shipment stipulated in clause Y.1
2.	HML contract	After term tokenization	[Clause Y.1], seller, ship, contract object, clause X, on or before, 15 August 2019, shanghai port, destination port of Hong Kong [Clause Y.2], seller, notify, buyer, shipment details, including, shipment date, shipping company, vessel name, digital hash, original, digitally, signed, bill of lading, within, 24 h, after the shipment stipulated in clause Y.1
		After term conceptualization	[Clause Y.1] =>(id:001-Y.1, ncm), seller =>(id:s11, nop), ship =>(ship, vtr), contract object =>(id:001-m, ncm), clause X =>(id:001-X, ncm), on or before =>(:<, adp), 15 August 2019 =>(date: 15 August 2019, ntv), Shanghai port =>((shipment port, ncm):(Shanghai port, ngp), ntv), destination port of Hong Kong =>((destination port, ncm):(Hong Kong port, ngp), ntv) [clause Y.2] =>(id:001-Y.2, ncm), seller =>(id:s11, nop), notify =>(notify, vdi), buyer =>(id:b11, nop), shipment details =>(shipment details, npr), shipment date =>(shipment date, ncm), shipping company =>(shipping company, ncm), vessel name =>(vessel name, ncm), digital hash =>(hash:(original =>(original, adj), digitally =>(digitally, adj), signed =>(signed, adj), bill of lading =>(bill of lading, ncm)), ntv), within =>(:<, adp), 24 h =>(hour:24, ntv), after the shipment stipulated in clause Y.1 =>(Y.2.after.Y.1, adp)
3.	MNL contract		id:001-Y.1:= (id:s11).n(id:001-x).b (date:2019/08/15:=<).btp at((shipment port).l:Shanghai).blp ship.p id:001-m (to.blp ((destination port).l:(Hong Kong)).cv; id:001-y.2:=id:s11 (hour:24:<).blp notify.p (id:b11).d (information:(id:001-x):((001-Y.2) :executed)).a and.cn ((shipment date).l, shipper.l, (vessel name).l, hash:(digitally.b signed.b original.g (bill of lading).l))).cn

4.2. Smart Contract Generation Approach Comparison

We compare AIASCG with other smart contract generation approaches qualitatively to present its advantage. Smart contract generation involves three types of interdependent activities: the iterative negotiation of contractual terms by involved parties, the documentation of the contractual agreements in natural languages, and the conversion of the natural language documents into smart contracts [5,7,9]. The negotiation and documentation are usually intertwined as an iterative process to prepare the contractual agreements [5], which are usually drafted by humans following certain specifications. In negotiation, the unambiguity of the exchanged contractual content is critical for achieving a universal contractual agreement. Meanwhile, documentation concerns the tools and methods to assist the manual drafting. Finally, conversion implements the methodologies to produce computer-executable smart contracts from the contractual agreements. It also validates that the generated computer-executable smart contracts are correct and truthful to the original contractual agreements. The correctness requires that the generated smart contract contains no execution error. The truthfulness requires that the execution of the generated smart contract should be consistent with the intention of the original contractual agreement. Therefore, to cover these activities, the comparison is conducted on five dimensions: input word segmentation; human effort in contract drafting; the expressiveness of smart contract specification; the validation capacity of the generation approach; and the executability of output. The comparison results are presented in Table 2.

Table 2. Comparison of smart contract generation methods.

Approach Type	Input Word Segmentation	Human Effort in Contract Drafting	Expressiveness of Smart Contract Specification	Validation Capacity of a Generation Approach	Output Executability
Arbitrary constraints to smart contracts	N/A	Very demanding	Program code correctness	Can validate correctness, hard to validate truthfulness	Not executable as smart contract skeleton; Limited executability as scripted-base languages; High executability as Turing-complete languages
Formally specified constraints to smart contracts	N/A	Demanding	Program code correctness	Can validate correctness, hard to validate truthfulness	Not executable as smart contract skeleton; Limited executability as scripted-base languages; High executability as Turing-complete languages
Business processes as smart contracts	N/A	Demanding	Key business process parameters	Can validate correctness and truthfulness theoretically	Logically executable
Legal agreements as smart contracts	N/A	Demanding	Temporal and operational constraints	Can validate correctness and truthfulness theoretically	Logically executable
AIASCG	Yes	Less demanding	Obligation constraints as specified by IDLC	Can validate correctness and truthfulness theoretically, with language-agnostic human readability and understandability	Logically executable as specified by IDLC

Input word segmentation evaluates whether the contractual agreements can be automatically segmented from the sentence to the words. In AIASCG, SpIn provides the accurate word recommendation to assist the lookup of best-matched term in CoDic as soon as possible. The automatic conversion from documents to the sequence of words can greatly improve the efficiency of manually keying contractual agreements by editors.

Human effort in contract drafting measures the amount and intensity of manual work involved in drafting a smart contract following certain specifications. Overall, the other four approaches in Table 2 demand a higher level of manual labor compared to AIASCG. AC-SC heavily relies on a programmer's knowledge and experience to interpret the contractual constraints, implement them directly as executable smart contracts, and test and optimize the implementation [14,19]. On the other hand, FC-SC expects the contractual constraints to be organized following formal and structured specifications, such that any input can be readily converted into smart contract codes automatically, e.g., via a domain-specific language (DSL) [16]. Therefore, it requires the contracting parties to be familiar with the underlying specifications to manually draft the input. Similarly, BP-SC and LA-SC require the input to follow certain specifications, although there are many design and editing tools available for modeling business processes and drafting legal contracts, such as diagram editing tools [17] and contract templates [53], which can increase the editing efficiency. AIASCG improves the editing workflow of contracts in IDLC. Instead of using SIM to manually edit a smart contract when MNL sentences are revised, with the AI-assisted word segmentation, the contracting parties only need to adjust the AI-generated output. This minimizes the requirement of manually inputting the natural language documents.

The expressiveness of smart contract specification concerns the capability to model certain types of constraints. The expressiveness of AC-SC and FC-SC depends on the underlying smart contract languages that are either imperative or declarative [14]. BP-SC concerns the key parameters of a business process to be properly modeled [17,18,54]. Meanwhile, LA-SC emphasizes the expressions of operational and temporal constraints in the legal documents [19]. AIASCG inherits the specifications of SSC proposed in [7] that describe the contracting parties and objects, activities, conditions, temporal constraints, and cross-reference of obligations.

The validation capacity of a generation approach examines whether the correctness and truthfulness of the output of the smart contract can be well validated. The smart contract codes produced in AC-SC and FC-SC can easily satisfy the correctness validation because the underlying programming languages usually contain sufficient validation tools [55,56]. For BP-SC, LA-SC, and AIASCG, the outputs of smart contracts are formally defined; therefore, the correctness can be validated by theoretical proof [57]. The truthfulness

validation, on one hand, requires understanding the semantic of the generated smart contracts; therefore, it is closely related to the human readability and understandability of the generation output. Because computer codes are not intuitive to be read and understood, supporting techniques such as formal semantic of codes [56] are required to mitigate this drawback. On the other hand, a logical smart contract that is the output of BP-SC and LA-SC is easier to validate its truthfulness because a logical representation is easier to read and understand. For AIASCG, the generated MNL contracts are not only readable and understandable by a human but can also be translated into different human natural languages, allowing language-agnostic human validation.

Finally, *output executability* evaluates if the smart contract produced by a generation approach can be executed as a computer program. For AC-SC and FC-SC, which generate smart contract codes as output, the codes are not always directly executable, because the generation results can be smart contract skeletons that require manual completion of its core execution logic [15]. Meanwhile, the implementation of executable smart contract codes can be based on script-based languages (e.g., BitML [58] and Simplicity [59]) or Turing-complete languages (e.g., Solidity (https://soliditylang.org/, accessed on 20 April 2022). As argued in [60], script-based languages are limited to executing financial transactions, whereas Turing-complete languages are more versatile. As for BP-SC and LA-SC, the output as logical smart contracts can achieve logical execution, i.e., the formally specified rules and instructions. Although their implementations may only produce smart contract code skeletons [17,19,54], with the logical smart contract as the intermediate representation, the logical execution can be decoupled with the implementation details, allowing more flexible and complex smart contract development. For AIASCG, the generated SSCs can also achieve logical execution as described in [7].

Overall, AIASCG introduces novel input word segmentation into the creation of smart contracts from natural language documents and effectively reduces human effort. Similar to other approaches utilizing the logical smart contract as the intermediate representation of a smart contract, AIASCG can express fairly complex contractual constraints clearly and concisely. The logic smart contract representation also allows intuitive, manual, and logical validation of a smart contract's correctness and truthfulness. Admittedly, the SSC generated by AIASCG can currently achieve logical executions as other logical smart contracts, while the actual implementation has been omitted and will be studied in separate research.

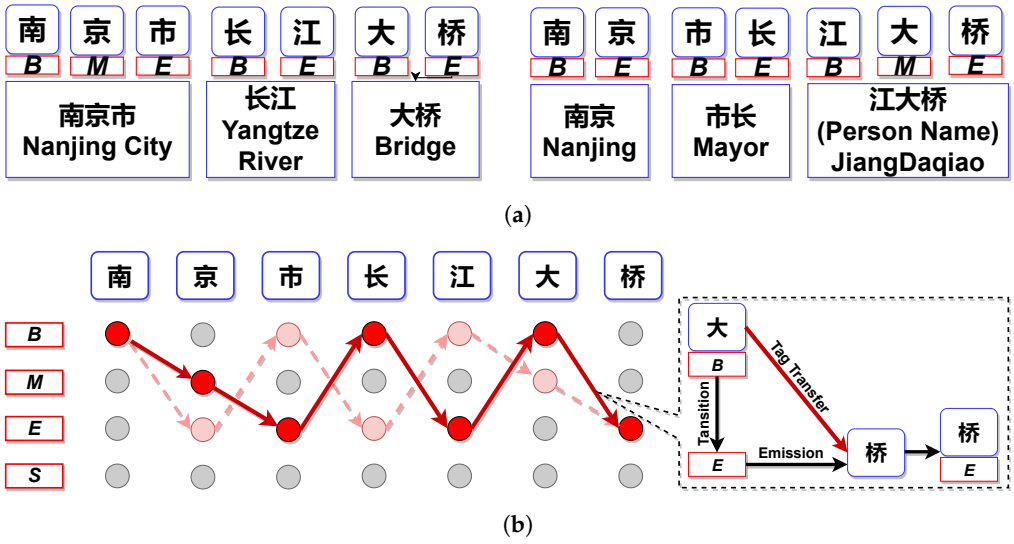
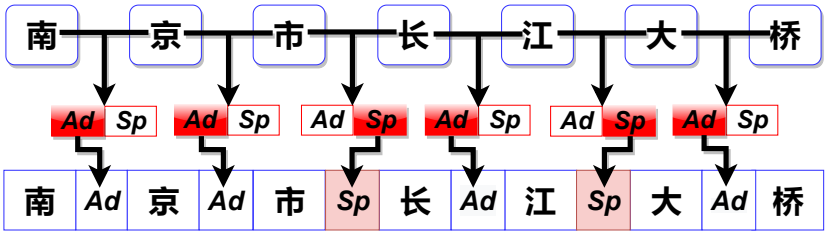


Figure 2. *Cont.*



(**c**)

Figure 2. Existing WS algorithm v.s. Proposed SpIn. The Chinese sentece indicates "Nanjing Yangtze River Bridge". (**a**) is the input sentence with its ground-truth segmentation at the left half and its potential segmentation on the right half. The proper word segmentation (left, about the bridge) and the improper segmentation (right, about the mayor) of the input. In (**b**), red arrows and circles stand for the correct tag sequence while the pink ones are improper. Existing WS algorithm based on the *BMES* tagging schema. In (**c**), states between characters rendered by red are the correct ones. Substituting the light red "Sp" with segmentation notations will give the word segmentation result. Conceptual Workflow of WS through Separation Inference.

5. Word Segmentation by Separation Inference

In the proposed AIASCG framework, AI-assisted word segmentation is a core component. Based on our previous work [61], we further integrate it into AIASCG and demonstrate that our proposed WS algorithm can effectively promote the efficiency of editors and reduce human effort. This section presents its implementation, including the underlying mechanism, network model, loss function, its advantages over existing methods, and our extension of the evaluation experiment.

5.1. SpIn-Based Word Segmentation Mechanism

WS by SpIn is the process of annotating the word boundaries with a set of predefined tags, as illustrated in Figure 2a, where segmentation is enclosed by the "*B*" and "*E*" tags. The correct and wrong segmentations are posited on the left and the right, respectively, in Figure 2a. Further observing the correct word segmentation (indicated by the red arrow) and the wrong word segmentation (indicated by the pink arrow) in Figure 2b, we can find that the inaccurate tag transition brings ambiguity. The potential segmentations caused by ambiguities are the segmentation of "mayor" and "bridge" are in the right half. The ambiguity is caused by the weakness of the word segmentation algorithm. Therefore, improving the word segmentation algorithm is critical for ensuring accurate semantic exchange. In addition, it is essential for providing accurate recommendation words to editors, especially in the smart contract generation that requires semantic consistency and common consent. Since existing methods treat WS as the sequence tagging task, different tagging schemas such as "BMES", "BIS", "START-NONSTART", and "Sep-App" are introduced. Despite different meanings, all these tagging schemas present implied position information of the current character (e.g., "Begin", "Middle", or "End" in the segmentation). The implied position limits transitions between tags. Take the "BMES" tagging schema for an instance, the tag "B" must be followed by "M" or "E". In addition, the tag of the first character in a sentence must be "B". Therefore, CRF is leveraged to constrain unreasonable tag transition. Applied CWS methods based on tagging schemas aim to attain the optimal tag transferring sequence rendered in red arrows as in Figure 2b. The CRF has alleviated the unreasonable tag prediction to some degree. However, the intrinsic weakness of existing tagging schemas, which is the implied restriction of position, is not well addressed.

Relying on the state-of-the-art performance of our proposed SpIn, we, therefore, introduce it into the AIASCG framework to ensure the accuracy of the recommendation word. As depicted in Figure 2c, for every two characters (Chinese characters in this example), they are either adhered to form (a part of) a complete semantic concept (a Chinese word in the example) or separated as they belong to distinct concepts. In other words, the connection state between every two neighboring characters can be described as

either adhesion (*Ad*) or separation (*Sp*). Therefore, we leverage bigram to adapt *"Ad-Sp"* to describe the separation state of every two characters. *"Ad-Sp"* eliminates the implied position restriction compared with existing tagging schemas. The separation state of the current bigram is not affected by the previous one. Meanwhile, there is no restriction on the first state in a sentence; it can be *Ad* or *Sp*.

Hence, our SpIn network can be built upon: (1) the bigram feature indicating the representation of two neighboring characters; (2) a classification network to explicitly learn the adhesion or separation state according to the current and all the other bigram features. The proposed SpIn concerns the adhesion or separation states between characters, such that it vanishes the constant requirement of involving complex networks or n-gram features to revise the tag transitions in WS. Elimination of extra contexts features and replacement of CRF network makes SpIn simple. In the following, we will introduce the architecture of the SpIn in detail.

5.2. The Structure of SpIn

The network of SpIn is illustrated in Figure 3. Based on their purposes, the network layers in SpIn can be divided into three groups: feature extraction and aggregation, dimension reduction, and separation inference.

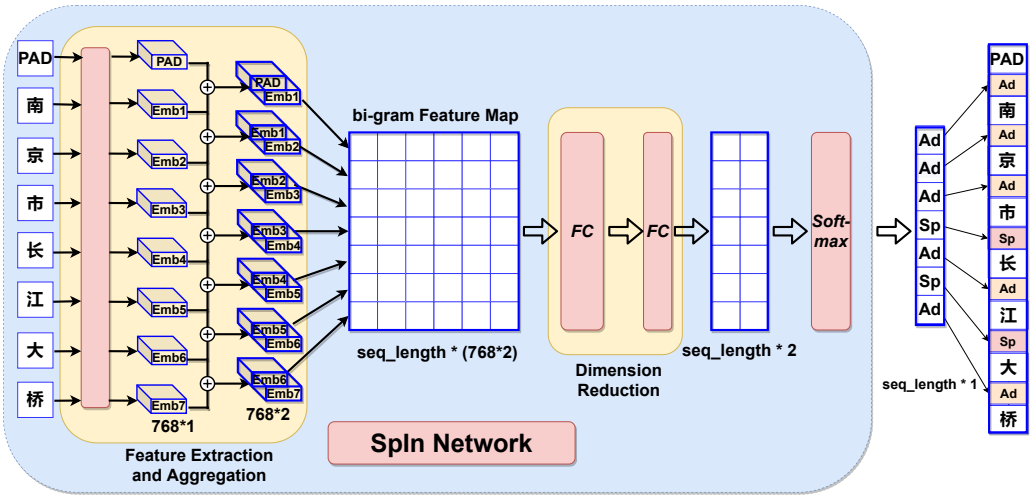


Figure 3. The network of Separation Inference. The Chinese sentece indicates "Nanjing Yangtze River Bridge".

5.2.1. Feature Extraction and Aggregation

Given an input sentence, SpIn first applies BERT [41], a well-recognized pre-trained model, to extract the embedding of each character (unigram embedding). Then, two neighboring unigram embeddings are concatenated to form a vector regarded as the bigram embedding. Notably, to match the length of the original input, we join the "PAD" token with the first character to form the first bigram, which is always classified as "adhesion". Finally, the bigram embeddings are aggregated to form the feature map of the entire input sentence. A unigram embedding has a size of $(768 * 1)$, a bigram embedding is $(768 * 2)$, and the bigram feature map has a size of seq_length $*(768 * 2)$, where seq_length is the max length of input sentences.

5.2.2. Dimension Reduction

The size of the feature map does not satisfy the input accepted by the inference layer. In addition, the bigram feature map relies on the unigram features generated by BERT, which will inevitably introduce nuisances if they are directly adopted for downstream tasks, because BERT is not elaborately refined for WS. Therefore, following previous works [42,43], the fully connected (FC) layers are designed to transform the bigram feature map for separation inference. We employ two adjacent Fully Connected (FC) layers with

the ReLU activate function to make the feature map into the inference size. The first FC layer changes the bigram feature map into $(n-1)*768$ dimensional fetaure. The second FC layer receives the result of the first FC layer and further reshapes the size into $(n-1)*2$. Therefore, the FC layers transform the feature map into the size suitable for inference. Meanwhile, with the size shrinkage, these two adjacent FC layers squeeze out the nuisances in the feature map. In this way, FC layers deal with the noise and size issues in the raw feature map.

5.2.3. Inference

In the end, the softmax layer receives the reshaped seq_length*2 feature map and predicts the sequences of tags. The softmax layer calculates the cross-entropy loss regarding the whole sentence. The inference results will be directly inserted between every two characters. Then, the final word segmentation results can be obtained.

5.3. The Loss Function of SpIn

Traditionally, considering tag y_i depends on the value of y_{i-1}; for the *"BMES"* tagging schemas in Figure 2b, existing WS algorithms always leverage the CRF network [28] to optimize tag transferring through viewing the tag sequence as the hidden states. Borrowing the notations in the CRF model, we denote the input sentence as the observation sequence $X = \{x_1, x_2, x_3 \dots x_i\}$, $(i \in N^+)$, the corresponding tags for word segmentation as the hidden state sequence $Y = \{y_1, y_2, y_3 \dots y_i\}, (i \in N^+)$ and $y_i \in \{\text{“}B\text{”}, \text{“}M\text{”}, \text{“}E\text{”}, \text{“}S\text{”}\}$ for $\forall y_i$ in Figure 2b. The CRF tries to find the optimal tag sequence Y^* by simultaneously learning the emission and transition matrices:

$$Y^* = \arg\max_{Y \in L^n} Score(X, Y) \tag{1}$$

where L^n are all the potential transferring sequences, $Score(X, Y) = Score\_Emit(X, Y) + Score\_Trans(X, Y)$, and these two scores on the right can be written as:

$$Score_{Emit}(X, Y) = \sum_i E_{(x_i, y_i)} \tag{2}$$

$$Score_{Trans}(X, Y) = \sum_i T_{(y_{i-1}, y_i, x_i)} \tag{3}$$

where E is the emission probability that projects its tag y_i to character x_i and T is the transition probability that tag y_{i-1} transits to y_i, when having x_i.

As there is no restriction of tag-transition for our proposed *"Ad-Sp"*, the probability of yielding adhesion and separation state sequence Y for the input sentence X is simplified as the sum of the conditional probability (equivalent to emission probability in CRF). For the sake of simplicity, we directly borrow the emission Score function from CRF.

$$s(X, Y) = \sum_i E_{(x_i, y_i)} = \sum_i p(y_i x_i) \tag{4}$$

Therefore, the CRF layer can be substituted with the softmax layer, and the loss function is rewritten as:

$$P(YX) = \frac{e^{s(X,Y)}}{\sum_{\bar{Y} \in L^n} e^{s(X,\bar{Y})}} \tag{5}$$

Furthermore, simplify Equation (5) through applying log-likelihood, we can have:

$$log(P(YX) = s(X, Y) - log(\sum_{\bar{Y} \in L^n} e^{s(X,\bar{Y})}) \tag{6}$$

where $-log(P(YX)$ is the loss we use in training the SpIn network.

5.4. Comparison with Existing Methods

For completeness, we introduce the differences in the mechanism between the proposed SpIn and existing WS algorithms. As introduced in Section 2.2, existing WS methods consider WS as the sequence labeling task and introduce various tagging schemas. Although researchers tried to investigate rich context features (i.e., context information, language-specific knowledge such as dictionaries, external knowledge) or complex network structure (e.g., Glyce [42] and WMSeg [43]) to achieve better WS, they all based their research on the tagging schema. Regardless of the different tagging schemas applied, each tag indicates the position of the current character in a segment. Therefore, the implied information (such as "Begin", "Middle", and "End" in the example in Figure 2b) restricts the tag-to-tag transition, requiring CRF or rich contexts information to handle inaccurate tag transitions.

Differently, in SpIn, the WS task is refined as finding all the separations in a sentence. Since the "*Ad*/*Sp*" states between any two neighboring characters do NOT rely on previous states and directly yield word segmentation results, the inference of "*Ad*/*Sp*" is independent of the previous states. Therefore, "*Ad-Sp*" gets rid of the restriction of the position of existing tagging schemas. Instead of leveraging rich context or external knowledge, only the current bigram (the concatenation of every two neighboring characters) is required to suit the "*Ad-Sp*". The state of every bigram is a classification task. Therefore, the widely applied CRF network is replaced with softmax (Equation (5)).

6. Evaluation of the SpIn Model

The performance of SpIn is evaluated from two perspectives: robustness and satisfaction. For robustness, SpIn is compared with state-of-the-art WS methods through automatic evaluation on multiple WS tasks, including Chinese Word Segmentation (CWS), Japanese Word Segmentation (JWS), Korean Word Segmentation (KWS), and Thai Word Segmentation (TWS), while for satisfaction, human evaluation is conducted. The state-of-the-art evaluation results achieved on CWS, JWS, KWS, and TWS prove that SpIn is universal (i.e., capable of processing multiple languages) and robust regardless of language.

6.1. Robustness Evaluation

To prove that SpIn is universal and robust regardless of languages, we evaluate the performance of SpIn on multiple WS tasks involving four languages and nine datasets.

Datasets. Five benchmark datasets are evaluated for CWS task, namely Chinese Penn Treebank 6.0 (CTB6) [62] and CITYU, AS, PKU, and MSR from the SIGHAN 2005 bakeoff task [63]. PKU, MSR, and CTB6 are in simplified Chinese whereas AS and CITYU are in traditional Chinese. SpIn is also verified on the JWS dataset BCCWJ version 1.1 (short for Balanced Corpus of Contemporary Written Japanese) [64] and KWS dataset UD_Korean-GSD corpora (https://github.com/emorynlp/ud-korean/tree/master/google, accessed on 8 April 2018) and Kaist (https://github.com/UniversalDependencies/UD_Korean-Kaist, accessed on 3 November 2021). BCCWJ covers various domain data. We follow the same dataset split with the Project Next NLP to attentively avoid data bias. UD_Korean-GSD and Kaist are two widely used datasets in syntactic parsing tasks and are automatically converted from structural trees in the Google UD Treebank [65] and the KAIST Treebank [66]. We extract words according to syntax structure, which carries implicit segmentation information. Moreover, we evaluate SpIn on the Thai Word Segmentation to further demonstrate its effectiveness and robustness. The evaluation of TWS is conducted on the InterBEST-2010 (Benchmark for Enhancing the Standard of Thai language processing) dataset following the previous work [67]. We follow the same data split with the work in [67]. The training and test set are 90% and 10%, respectively.

Parameter settings and evaluation metrics. For the SpIn network, we uniformly set the sequence length as 128; the learning rate as 5×10^{-5}, the batch size as 32, and the epochs of training as 15. We employ the early stop mechanism to avoid over-fitting and leverage the Adam as the optimizer during the training process. In the experiments, all

the above-mentioned parameters are set still. Furthermore, following the widely accepted evaluation methodologies, the average Micro F1 score of multi-time experimental results is adopted in our experiments as the metric for exhibiting reliability. Moreover, another essential metric, which is the Recall of Out-of-Vocabulary (R_OOV) words, is leveraged to evaluate the generalization of the word segmentation algorithm.

Evaluation results. Tables 3–6 presented the results of CWS, JSW, KWS, and TWS, respectively. We adopt the average score of the experiments repeated 10 times for solid evaluations. Following previous works, the results are accurate to one decimal place. A single word segment is considered as the unit for calculating the F1 score and R_OOV. SpIn achieves state-of-the-art results on all nine datasets regardless of language. Compared with the previous methods, SpIn brought a **+1.3%** (PKU dataset) improvement in the F1 score. The least improvement of SpIn on the CWS task is a **+0.4%** F1 score (on the MSR dataset). Moreover, SpIn achieves the best result on the JWS task as well. Considering the lofty baselines in CWS and JWS, these promotions are significant enough in proving the universality of SpIn on WS tasks. Of note, the switch_LSTMs_CWS [68] in Table 3 also exploited the bigram feature, but are fallen far behind SpIn. Therefore, the bigram feature is not the undercovered actuator of SpIn.

Table 3. Comparisons between SpIn and previous state-of-the-art results on the CWS task.

	CITYU		AS		PKU		MSR		CTB6	
	F1	**R_oov**	**F1**	**R_oov**	**F1**	**R_oov**	**F1**	**R_oov**	**F1**	**R_oov**
DGRNN [69]	-	-	-	-	96.1	-	96.3	-	95.8	-
Bi-LSTMs_CWS [31]	97.2	87.5	96.2	70.7	96.1	78.8	97.4	80.0	96.7	85.4
Glyce [42]	**97.9**	-	**96.7**	-	**96.7**	-	**98.3**	-	-	-
WMSEG [43]	97.8	**87.57**	96.58	78.48	96.51	**86.76**	98.28	**86.67**	97.16	**88.00**
SpIn	**98.6**	**90.57**	**97.5**	**81.36**	**98.0**	**93.53**	**98.7**	**93.13**	**98.6**	**93.90**

Table 4. Comparison between SpIn and previous state-of-the-art methods on the JWS task.

	BCCWJ	
	F1	**R_oov**
LSTM_JWS [70]	98.42	-
Word_Attention_JWS [71]	98.93	-
SpIn	**98.94**	**93.01**

Table 5. Comparison between SpIn and previous state-of-the-art methods on the KWS task.

	KAIST		GSD	
	F1	**R_oov**	**F1**	**R_oov**
BMES+Unigram_Feature	87.62	78.34	87.12	78.27
SpIn	**92.37**	**83.81**	**91.19**	**82.24**

Table 6. Comparison between SpIn and previous state-of-the-art methods on the TWS task.

	InterBEST-2010	
	F1	**R_oov**
Syllable-based-TWS [67]	95.59	67.42
SpIn	**95.61**	**70.83**

Moreover, SpIn also notably escalates the recall rate of OOV words in Tables 3 and 4. On every single dataset for the above four WS tasks, SpIn achieves the best OOV performance. The results demonstrate the effectiveness of SpIn on OOV words. The eye-catching OOV recall improvement is **+6.77%** on the PKU dataset. For the AS dataset, there is still a **+2.88%** boost. Such massive promotions on the OOV recall ensure the generalization capability of SpIn.

Compared with the work [42], which involves rich pictographic feature representations, our features are simple bigram features via concatenation of unigram features. Instead of exploiting rich external knowledge and leveraging wordhood information to incorporate with the framework as in [43], our SpIn is closed without involving additional information.

Compared with previous works using the word dictionary and linguistic information (such as character type information), SpIn, without any extra common knowledge, gains state-of-the-art results on the JWS task as well.

As there is no related segmentation work about these Korean datasets, we compare SpIn with traditional methods, which widely employ character features and character-based tagging schemas and report the results in Table 5. SpIn achieves a better performance on both datasets. Especially, SpIn boosts up to **+4.75%** F1 score improvement on the GSD dataset. Furthermore, we observe that SpIn performs better for OOV words on both datasets.

The evaluation of TWS is the extension work for verifying the effectiveness of SpIn. As opposed to [67], which leverages syllable embeddings to capture linguistic features, our SpIn merely considers text features. We follow the recent SOTA work and conduct comparative experiments. Instead of the work [67] that leverages syllable embeddings to capture linguistic features, our SpIn merely considers text features. Table 6 reports the evaluation results and demonstrates that the SpIn also works for TWS.

Although contextual features have been hotly discussed and proved to be effective in existing methods , they may result in a performance loss of the model. As the existing tagging schemas are character-based, employing context features (e.g., bigram, trigram, and quadrigram feature) may introduce noise for the learning model. However, bigram is the natural choice in our proposed SpIn, since we model the connection state of two adjacent characters. For the fixed separation inference, no other context or lexical information is desired, and the connections between characters are fully explored and exploited. Moreover, the restriction of the tag-to-tag transition and the limitation of the first tag in existing tagging schemas make the tag inference more complicated and intractable. This inherent problem causes inaccuracy during the inference. In contrast, our proposed "Ad-Sp" eliminates the implicit restriction of the position in the existing tagging schemas. Therefore, the appropriate combination of our proposed "Ad-Sp" and its specially tailored bigram features brings model promotion.

6.2. Human Evaluation

Besides automatic evaluation on regular WS datasets, human evaluation was conducted to prove the robustness of SpIn in processing natural language expressions collected from the legal contract. The evaluation was mainly conducted in the Chinese document. Particularly, 200 sentences were randomly selected from the collected data for the evaluation. We mainly conducted human evaluation from the following two aspects:

- How much editing time can SpIn's automatic word segmentation save? We provided five intervals to describe the percentage of time that can be saved.
  - [0–20%)
  - [20–40%)
  - [40–60%)
  - [60–80%)
  - [80–100%]
- How satisfied are you with SpIn? The degree of satisfaction in this survey was as below:
  - Very useless
  - Useless
  - Generally useful
  - Useful
  - Very useful

The evaluation was completed by native Chinese speakers from 25 to 40 years old with sufficient knowledge. These participants were proficient in MNL and IDLC. Considering the rigorousness of the questionnaire, 200 sentences were evaluated by three editors simultaneously. The sentence length was between 10 and 50. The evaluation of time-saving is reported in Table 7. We can conclude that multiple editors believed that 88.67% of sentences can save 80–100% of the time through automatic word segmentation. In addition, we recruited 50 undergraduate and graduate students majoring in law that are familiar with the legal contract. Before conducting the questionnaire, they were trained with MNL and IDLC. The evaluation of satisfaction is listed in Table 8. In total, 46 of 50 people rated SpIn as very useful. Up to 92% of editors considered automatic word segmentation to be very practical.

Table 7. Time-saving assessment. There are 200 total sentences. Each interval represents the number of sentences that save the corresponding time of each editor. Avg_sent indicates the average sentences located in each interval. The percentage represents the fraction of sentences that save the corresponding time.

Time Saving	[0–20%)	[20–40%)	[40–60%)	[60–80%)	[80–100%)
Editor 1	0	0	7	19	174
Editor 2	0	1	5	12	182
Editor 3	0	1	6	17	176
Avg_sent	-	0.67	6	16	177.3
Percentage	-	0.3%	3%	8%	88.67%

Table 8. The evaluation of the satisfaction of SpIn.

Satisfaction	Very Useless	Useless	Generally Useful	Useful	Very Useful
Editors	0	0	0	4	46

7. Conclusions and Future Work

In the past decade, the smart contract has been well recognized as the most promising application of blockchain, providing the advantages of automation, traceability, and immutability to the executions of on-chain business activities. Moreover, smart contract's emergence has brought up the challenging issue of its validity as a legal agreement considering the significance of its contractual objects and transaction volume.

An open research question relevant to the legal validity of the smart contract is how a legal document can be represented as an executable smart contract. In our preliminary work, the representation problem is addressed by IDLC, which defines the formal model of the smart contract representation as SSC, the transformation mechanism from natural language contracts to SSCs via MNL mediation, and the execution model. Nevertheless, the contract generation phase of IDLC requires the users to manually input the initial documents as MNL sentences, which is exhausting and error-prone. In this paper, we argue that most of the manual editing in smart contract generation can be automated, including its process and output. In this way, the contracting parties can simply review and revise the computer-generated smart contracts to ensure their quality, significantly increasing the efficiency of smart contract generation.

We conceptualize the AIASCG framework as a new smart contract generation method. AIASCG emphasizes AI-assistant editing and provides automatic split of sentence into words. Hence, we proposed an AI-based model named SpIn that can efficiently perform automatic word segmentation on a natural language document. In the robustness evaluation, SpIn achieves state-of-the-art F1 scores and Recall of Out-of-Vocabulary (R_OOV) words on multiple word segmentation tasks, while in the human evaluation, participants believe that 88.67% of sentences can be saved 80–100% of the time through automatic word segmentation. With its high accuracy and robustness, SpIn is integrated into the

AIASCG framework as a core component to facilitate the human-supervised automatic smart contracts generation.

In the future, the proposed AIASCG framework can be improved from at least two directions: interoperability of SIM and completeness of SpIn language support. Firstly, the current implementation of SIM in [52] has not been actively maintained, so its compatibility with existing software may be unsatisfying. To enhance its interoperability with external software and tools, it is necessary to redesign SIM with well-defined application programming interfaces (APIs). Secondly, SpIn currently supports WS in Chinese, Japanese, Korean, and Thai documents. Nevertheless, SpIn can be trained on the corpus in other languages to be more versatile.

Author Contributions: Conceptualization: Y.T.; Methodology: Y.T.; Validation: Y.T.; Formal analysis: Y.T.; Investigation: Y.T. and W.T.; Writing—original draft preparation: Y.T. and W.T.; Writing—review and editing: B.S. and P.Q.; Supervision: J.G. and S.Z.; Funding acquisition: S.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Macau University of Science and Technology FRG (Grant No. FRG-20-024-MSB).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shen, B.; Guo, J.; Yang, Y. MedChain: Efficient healthcare data sharing via blockchain. *Appl. Sci.* **2019**, *9*, 1207. [CrossRef]
2. Chang, S.E.; Chen, Y.-C.; Lu, M.-F. Supply chain re-engineering using blockchain technology: A case of smart contract based tracking process. *Technol. Forecast. Soc. Chang.* **2019**, *144*, 1–11. [CrossRef]
3. Baqa, H.; Truong, N.B.; Crespi, N.; Lee, G.M.; le Gall, F. Semantic smart contracts for blockchain-based services in the internet of things. In Proceedings of the 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 26–28 September 2019.
4. Wang, X.; Yang, W.; Noor, S.; Chen, C.; Guo, M.; van Dam, K.H. Blockchain-based smart contract for energy demand management. *Energy Procedia* **2019**, *158*, 2719–2724. [CrossRef]
5. Zheng, Z.; Xie, S.; Dai, H.-N.; Chen, W.; Chen, X.; Weng, J.; Imran, M. An overview on smart contracts: Challenges, advances and platforms. *Future Gener. Comput. Syst.* **2020**, *105*, 475–491. [CrossRef]
6. Qin, P.; Guo, J. A novel machine natural language mediation for semantic document exchange in smart city. *Future Gener. Comput. Syst.* **2020**, *102*, 810–826. [CrossRef]
7. Qin, P.; Tan, W.; Guo, J.; Shen, B. Intelligible description language contract (IDLC)—A novel smart contract model. *Inf. Syst. Front.* **2021**._2018.2018.00183 [CrossRef]
8. Szabo, N. Formalizing and securing relationships on public networks. *First Monday* **1997**, *2*, 9. [CrossRef]
9. Governatori, G.; Idelberger, F.; Milosevic, Z.; Riveret, R.; Sartor, G.; Xu, X. On legal contracts, imperative and declarative smart contracts, and blockchain systems. *Artif. Intell. Law* **2018**, *26*, 377–409. [CrossRef]
10. Grigg, I. The ricardian contract. In Proceedings of the First IEEE International Workshop on Electronic Contracting, San Diego, CA, USA, 6 July 2004.
11. Clack, C.D. Smart contract templates: Legal semantics and code validation. *J. Digit. Bank.* **2018**, *2*, 338–352.
12. Stark, J. Making Sense of Blockchain Smart Contracts. 2016. Available online: https://www.coindesk.com/making-sense-smart-contracts (accessed on 22 July 2021).
13. Khan, S.N.; Loukil, F.; Ghedira-Guegan, C.; Benkhelifa, E.; Bani-Hani, A. Blockchain smart contracts: Applications, challenges, and future trends. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 2901–2925. [CrossRef]
14. Idelberger, F.; Governatori, G.; Riveret, R.; Sartor, G. Evaluation of logic-based smart contracts for blockchain systems. In *Rule Technologies. Research, Tools, and Applications*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 167–183.
15. Choudhury, O.; Rudolph, N.; Sylla, I.; Fairoza, N.; Das, A. Auto-generation of smart contracts from domain-specific ontologies and semantic rules. In Proceedings of the 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 30 July–3 August 2018.
16. Seijas, P.L.; Nemish, A.; Smith, D.; Thompson, S. Marlowe: Implementing and analysing financial contracts on blockchain. In *Financial Cryptography and Data Security*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 496–511.

17. Garamvolgyi, P.; Kocsis, I.; Gehl, B.; Klenik, A. Towards model-driven engineering of smart contracts for cyber-physical systems. In Proceedings of the 2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Luxembourg, 25–28 June 2018.
18. Dolgui, A.; Ivanov, D.; Potryasaev, S.; Sokolov, B.; Ivanova, M.; Werner, F. Blockchain-oriented dynamic modelling of smart contract design and execution in the supply chain. *Int. J. Prod. Res.* **2019**, *58*, 2184–2199. [CrossRef]
19. Frantz, C.K.; Nowostawski, M. From institutions to code: Towards automated generation of smart contracts. In Proceedings of the 2016 IEEE 1st International Workshops on Foundations and Applications of Self Systems (FASW), Augsburg, Germany, 12–16 September 2016.
20. Navigli, R. Word sense disambiguation: A survey. *ACM Comput. Surv.* **2019**, *41*, 2. [CrossRef]
21. Pillai, L.R.; Veena, G.; Gupta, D. A combined approach using semantic role labelling and word sense disambiguation for question generation and answer extraction. In Proceedings of the 2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC), Bangalore, India, 9–10 February 2018; pp. 1–6.
22. Pu, X.; Pappas, N.; Henderson, J.; Popescu-Belis, A. Integrating weakly supervised word sense disambiguation into neural machine translation. *Trans. Assoc. Comput. Linguist.* **2018**, *6*, 635–649. [CrossRef]
23. Seifollahi, S.; Shajari, M. Word sense disambiguation application in sentiment analysis of news headlines: An applied approach to forex market prediction. *J. Intell. Inf. Syst.* **2019**, *52*, 57–83. [CrossRef]
24. Hristea, F.; Colhon, M. The long road from performing word sense disambiguation to successfully using it in information retrieval: An overview of the unsupervised approach. *Comput. Intell.* **2020**, *36*, 1026–1062. [CrossRef]
25. Wang, Y.; Wang, M.; Fujita, H. Word sense disambiguation: A comprehensive knowledge exploitation framework. *Knowl.-Based Syst.* **2020**, *190*, 105030. [CrossRef]
26. Guo, J.; Xu, L.D.; Xiao, G.; Gong, Z. Improving multilingual semantic interoperation in cross-organizational enterprise systems through concept disambiguation. *IEEE Trans. Ind. Inform.* **2012**, *8*, 647–658. [CrossRef]
27. Low, J.K.; Ng, H.T.; Guo, W. A maximum entropy approach to chinese word segmentation. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, Jeju Island, Korea, 14–15 October 2005.
28. Lafferty, J.D.; McCallum, A.; Pereira, F.C.N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*; Ser. ICML '01; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2001, pp. 282–289.
29. Zhao, H.; Huang, C.; Li, M. An improved chinese word segmentation system with conditional random field. In Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing, Sydney, Australia, 22–23 July 2006; pp. 162–165.
30. Yang, J.; Zhang, Y.; Liang, S. Subword encoding in lattice lstm for chinese word segmentation. *arXiv* **2018**, arXiv:1810.12594.
31. Ma, J.; Ganchev, K.; Weiss, D. State-of-the-art Chinese word segmentation with Bi-LSTMs. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, 31 October–4 November 2018; pp. 4902–4908.
32. Yang, J.; Zhang, Y.; Dong, F. Neural word segmentation with rich pretraining. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 839–849.
33. Ma, J.; Hinrichs, E. Accurate linear-time chinese word segmentation via embedding matching. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), Beijing, China, 26–31 July 2015; Association for Computational Linguistics: Stroudsburg, PA, USA, 2015; pp.1733–1743.
34. Zhang, Y.; Clark, S. Chinese segmentation with a word-based perceptron algorithm. In Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic, 23–30 June 2007; pp. 840–847.
35. Chen, X.; Qiu, X.; Zhu, C.; Liu, P.; Huang, X.-J. Long short-term memory neural networks for chinese word segmentation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015; pp. 1197–1206.
36. Pei, W.; Ge, T.; Chang, B. Max-margin tensor neural network for Chinese word segmentation. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Baltimore, MD, USA, 22–27 June 2014; pp. 293–303.
37. Zhang, L.; Wang, H.; Sun, X.; Mansur, M. Exploring representations from unlabeled data with co-training for chinese word segmentation. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 311–321.
38. Sun, W.; Xu, J. Enhancing chinese word segmentation using unlabeled data. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Minneapolis, MI, USA, 27–31 July 2011; pp. 970–979.
39. Wang, Y.; Kazama, J.; Tsuruoka, Y.; Chen, W.; Zhang, Y.; Torisawa, K. Improving Chinese word segmentation and POS tagging with semi-supervised methods using large auto-analyzed data. In Proceedings of 5th International Joint Conference on Natural Language Processing, Chiang Mai, Thailand, 8–13 November 2011; pp. 309–317.
40. Liu, Y.; Zhang, Y. Unsupervised domain adaptation for joint segmentation and pos-tagging. In Proceedings of the COLING, 2012: Posters, Mumbai, India, 8–15 December 2012; pp. 745–754.
41. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.

42. Meng, Y.; Wu, W.; Wang, F.; Li, X.; Nie, P.; Yin, F.; Li, M.; Han, Q.; Sun, X.; Li, J. Glyce: Glyph-vectors for chinese character representations. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 2746–2757.
43. Tian, Y.; Song, Y.; Xia, F.; Zhang, T.; Wang, Y. Improving chinese word segmentation with wordhood memory networks. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020 ; pp. 8274–8285.
44. Yang, S.; Guo, J.; Wei, R. Semantic interoperability with heterogeneous information systems on the internet through automatic tabular document exchange. *Inf. Syst.* **2017**, *69*, 195–217. [CrossRef]
45. Guo, J.; Lam, I.H.; Chan, C.; Xiao, G. Collaboratively maintaining semantic consistency of heterogeneous concepts towards a common concept set. In Proceedings of the 2nd ACM SIGCHI Symposium on Engineering Interactive Computing Systems—EICS, Berlin, Germany, 19–23 June 2010.
46. Saussure, F.D. *Course in General Linguistics*; Columbia University Press: New York City, NY, USA, 2011.
47. Peirce, C.S. *Peirce on Signs: Writings on Semiotic*; UNC Press Books: Chapel Hill, NC, USA, 1991.
48. Guo, J. SDF: A sign description framework for cross-context information resource representation and interchange. In Proceedings of the 2014 Enterprise Systems Conference, Shanghai, China, 2–3 August 2014.
49. Fillmore, C. The case for case. UC Berkeley Linguistics. ERIC. 1967; p. 135 Available online: http://linguistics.berkeley.edu/~syntax-circle/syntax-group/spr08/fillmore.pdf (accessed on 1 April 2022).
50. Cook, V.J. Chomsky's universal grammar and second language learning. *Appl. Linguist.* **1985**, *6*, 2–18. [CrossRef]
51. Xin, D.; Ma, L.; Liu, J.; Macke, S.; Song, S.; Parameswaran, A. Accelerating human-in-the-loop machine learning. In Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning, Houston, TX, USA, 15 June 2018.
52. Xiao, G.; Guo, J.; Gong, Z.; Li, R. Semantic input method of chinese word senses for semantic document exchange in e-business. *J. Ind. Inf. Integr.* **2016**, *3*, 31–36. [CrossRef]
53. Tateishi, T.; Yoshihama, S.; Sato, N.; Saito, S. Automatic smart contract generation using controlled natural language and template. *IBM J. Res. Dev.* **2019**, *63*, 1–12. [CrossRef]
54. Zupan, N.; Kasinathan, P.; Cuellar, J.; Sauer, M. Secure smart contract generation based on petri nets. In *Blockchain Technology for Industry 4.0*; Springer: Singapore, 2020; pp. 73–98.
55. Bhargavan, K.; Delignat-Lavaud, A.; Fournet, C.; Gollamudi, A.; Gonthier, G.; Kobeissi, N.; Kulatova, N.; Rastogi, A.; Sibut-Pinote, T.; Swamy, N.; et al. Formal verification of smart contracts. In Proceedings of the 2016 ACM Workshop on Programming Languages and Analysis for Security, Vienna, Austria, 24 October 2016.
56. Bartoletti, M.; Zunino, R. Formal models of bitcoin contracts: A survey. *Front. Blockchain* **2019**, *2*, 8. [CrossRef]
57. Dwivedi, V.; Pattanaik, V.; Deval, V.; Dixit, A.; Norta, A.; Draheim, D. Legally enforceable smart-contract languages. *ACM Comput. Surv.* **2021**, *54*, 1–34. [CrossRef]
58. Bartoletti, M.; Zunino, R. BitML: A Calculus for Bitcoin Smart Contracts. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018.
59. O'Connor, R. Simplicity: A new language for blockchains. In Proceedings of the 2017 Workshop on Programming Languages and Analysis for Security, Dallas, TX, USA, 30 October 2017.
60. Hu, B.; Zhang, Z.; Liu, J.; Liu, Y.; Yin, J.; Lu, R.; Lin, X. A comprehensive survey on smart contract construction and execution: Paradigms, tools, and systems. *Patterns* **2021**, *2*, 100179. [CrossRef]
61. Tong, Y.; Guo, J.; Zhou, J. Separation inference: A unified framework for word segmentation in east asian languages. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2022**, *30*, 1521–1530. [CrossRef]
62. Xue, N.; Xia, F.; Chiou, F.-D.; Palmer, M. The penn chinese TreeBank: Phrase structure annotation of a large corpus. *Nat. Lang. Eng.* **2005**, *11*, 207–238. [CrossRef]
63. Emerson, T. The second international chinese word segmentation bakeoff. In Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing, Jeju Island, Korea, 14–15 October 2005.
64. Maekawa, K.; Yamazaki, M.; Ogiso, T.; Maruyama, T.; Ogura, H.; Kashino, W.; Koiso, H.; Yamaguchi, M.; Tanaka, M.; Den, Y. Balanced corpus of contemporary written japanese. *Lang. Resour. Eval.* **2014**, *48*, 345–371. [CrossRef]
65. McDonald, R.; Nivre, J.; Quirmbach-Brundage, Y.; Goldberg, Y.; Das, D.; Ganchev, K.; Hall, K.; Petrov, S.; Zhang, H.; Täckström, O.; et al. Universal dependency annotation for multilingual parsing. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Sofia, Bulgaria, 4–9 August 2013; pp. 92–97.
66. Choi, K.-S.; Han, Y.S.; Han, Y.G.; Kwon, O.W. Kaist tree bank project for korean: Present and future development. In Proceedings of the International Workshop on Sharable Natural Language Resources, Nara, Japan, 10–11 August 1994; pp. 7–14.
67. Chormai, P.; Prasertsom, P.; Cheevaprawatdomrong, J.; Rutherford, A. Syllable-Based Neural Thai Word Segmentation. In *Proceedings of the 28th International Conference on Computational Linguistics*; International Committee on Computational Linguistics: Barcelona, Spain, 2020; pp. 4619–4637. Available online: https://aclanthology.org/2020.coling-main.407 (accessed on 8 December 2020).
68. Gong, J.; Chen, X.; Gui, T.; Qiu, X. Switch-lstms for multi-criteria chinese word segmentation. *Proc. Aaai Conf. Artif. Intell.* **2019**, *33*, 6457–6464. [CrossRef]
69. Xu, J.; Sun, X. Dependency-based gated recursive neural network for chinese word segmentation. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Berlin, Germany, 7–12 August 2016; pp. 567–572.

70. Kitagawa, Y.; Komachi, M. Long short-term memory for japanese word segmentation. *arXiv* **2017**, arXiv:1709.08011.
71. Higashiyama, S.; Utiyama, M.; Sumita, E.; Ideuchi, M.; Oida, Y.; Sakamoto, Y.; Okada, I. Incorporating word attention into character-based word segmentation. In Proceedings of the 2019 Conference of the North, Minneapolis, MI, USA, 1–6 September 2019.