

SPECIAL ISSUE PAPER

Adaptive-capacity and robust natural language watermarking for agglutinative languages

Mi-Young Kim* and Randy Goebel

University of Alberta, Department of Computing Science, Edmonton, Alberta, Canada

ABSTRACT

We present a robust and adaptive-capacity watermarking algorithm for agglutinative languages. All processes, including the selection of sentences to be watermarked, watermark embedding, and watermark extraction, are based on syntactic dependency trees. We show that it is more robust to use syntactic dependency trees than the surface forms of sentences in text watermarking. For the agglutinative languages, we embed watermark using the two main characteristics of the languages. First, because a word consists of several morphemes, we can watermark sentences using morphological division/combination without deep linguistic analysis. Second, they permit relatively free word order, so we can move a syntactic constituent within its clause. Finally, to increase the information-hiding capacity, we adaptively compute the number of watermark bits to be embedded for each sentence.

We perform three kinds of evaluation: perceptibility, robustness, and capacity of our method. High capacity is achieved by dynamically determining possibly embedded watermark bits for each sentence. The secret rank based on a syntactic dependency tree strengthens robustness of our method. Finally, we show that the displacement of syntactic constituents and morphological division/combination does not affect the style and naturalness of the text. Copyright © 2011 John Wiley & Sons, Ltd.

KEYWORDS

natural language watermarking; agglutinative language; morphological division; syntactic dependency tree

*Correspondence

Mi-Young Kim, University of Alberta, Department of Computing Science, Edmonton, Alberta, Canada.

E-mail: colorful@postech.ac.kr

1. INTRODUCTION

Text watermarking is an emerging research area that combines natural language processing and information security. The goal is to embed additional information into the natural text, for subliminal communication, hidden information transmission, content, and authorship authentication, and generally enrich the text with metadata [1].

Watermarking techniques have been extensively explored for multimedia documents in the last decade [2]. In contrast, the studies on text watermarking are just starting.

For example, in [3–6], the techniques of synonym substitution for watermarking have been addressed, and various attack scenarios have been described. The ambiguity induced on the word precision by the synonym substitution technique has led Topkara *et al.* [7] to syntax-based natural language watermarking. Their technique basically focuses on the syntactic sentence paraphrasing. It turns out that the syntactic approach offers the richest set of text watermarking tools.

Note that agglutinative language differs significantly from Indo-European languages in that it permits relatively

free word order and that a word in the agglutinative languages consists of several morphemes. In the agglutinative languages, morphological division/combination and the movement of a syntactic constituent within its clause do not change the meaning or style of the text. For this reason, we believe that agglutinative languages are an appropriate target for text watermarking based on morphological division/combination and syntactic adverbial movement.

We believe that the robustness of watermarking is improved when one uses a metarepresentation, such as syntactic structures, rather than the surface sentences. Because watermarking is performed based on syntactic structures, it is difficult for an attacker to find out those watermark patterns from the original sentences.

Among syntactic structure types, we use syntactic dependency trees for watermarking of the agglutinative languages, because they are simple, easily constructable, and more popular for representing the syntactic structures of agglutinative languages.

Within this structure of using dependency trees, we include a method to adaptively determine the number of watermark bits to be embedded for each sentence, based on

the characteristics of the dependency tree. This provides a basis for both high and adaptive information-hiding capacity.

2. ADAPTIVE-CAPACITY AND ROBUST TEXT WATERMARKING

2.1. Overall process

F1 As shown in Figure 1, watermark embedding involves several steps. First, we performed syntactic analysis for the original text. Second, we adaptively decided the number of watermark bits to be embedded for each sentence. Third, we chose sentences to be watermarked according to their secret ranks, in ascending rank order and embed a watermark by modifying syntactic dependency trees. Finally, from the modified syntactic trees, we generated marked sentences.

The following are the parameters and functions for watermarking:

| | |
|-----------------------------|---|
| $T = \{s_1, s_2, \dots\}$: | text document |
| $W = \{W_1, \dots, W_n\}$: | n watermark bits |
| $K = \{0, 1\}^*$: | the key set. |
| $h1$: | $K \times \{0, 1\}^* \rightarrow \{0, 1\}^{160}$, a keyed hash function with 160-bit output. |
| syn_rank : | $s_i \rightarrow \{0, 1\}^*$ |
| syn_wm : | $s_i \times t \rightarrow \{0, 1\}$, where t is a number |

We note that $h1$ can be easily constructed from standard cryptographic hash functions such as SHA-1.

Algorithm 1 shows the watermark embedding algorithm.

Algorithm 1 Watermark embedding.

- 1: **Input:** $T = \{s_1, s_2, \dots\}$: text document,
 - 2: $K = \{0, 1\}^*$: the key set.
 - 3: **Output:** $T_w = \{S'_1, S'_2, \dots\}$
 - 4: Compute $R_i = h1(K, syn_rank(s_i))$ for each sentence.
 - 5: Here R_i is used as a secret rank of s_i .
 - 6: We determine the number of watermark bits to be
 - 7: embedded for each sentence based on its dependency
 - 8: tree.
 - 9: Finally, in the text T , rewrite the m sentences $\{s_1, s_2$
 - 10: $, s_3, \dots, s_m\}$ with least ranks to $\{s'_1, s'_2, s'_3, \dots, s'_m\}$
 - 11: by modifying syntactic dependency trees such that
 - 12: $syn_wm(s'_1, 1) = W_1, syn_wm(s'_1, 2) = W_2 \dots$
 - 13: $syn_wm(s'_1, p) = W_p, \dots, syn_wm(s'_m, 1) = W_{n-q+1}$,
 - 14: $syn_wm(s'_m, q) = W_n$, and
 - 15: $syn_rank(s_i) = syn_rank(s'_i)$.
-

In Algorithm 1, the sentence s'_1 embeds p watermark bits, and the sentence s'_m embeds q watermark bits.

To determine p and q , we computed the number of possible morphological division/combinations and syntactic constituent movements from each syntactic dependency tree of s_1 and s_m .

In line 4 of Algorithm 1, syn_rank is used to assign a secret rank to each sentence. For this function, we used not only the surface form of a sentence but also the syntactic structure of the sentence. We used the length of each word from the surface form of the sentence and the depth of each node from the syntactic tree. Because the surface form of the sentence can be easily revealed to an attacker, depth information is more secure than the word length. Therefore, we made the syn_rank value more sensitive to the “hidden” depth value by raising the depth value to the power of two.

The function syn_rank is computed as follows:

$$syn_rank(s) = \sum_i 2^{depth(i)} \times length(i), \quad (1)$$

where $depth(i)$ is the depth of the i -th node and $length(i)$ is the length of the word in the i -th node.

Because an authorized person should obtain the same secret rank from the marked sentence, the output of the function syn_rank should not be changed after watermark embedding, as describe in line 15 of Algorithm 1. That means that the morphological division/combination and the movement of syntactic constituents should not change the output of syn_rank .

The depth of each node is not changed by syntactic constituent movement, because the syntactic constituent only changes its position into one of its siblings' positions.

However, after morphological division/combination, a node can be divided, or two nodes can be combined. To ensure that $syn_rank(s) = syn_rank(s')$, we combined divided tree nodes into one node by chunking. A detailed explanation will be given Section 2.4.

Consider syn_wm in line 12 of Algorithm 1. The output of syn_wm is the watermark bit, and it should reflect the change of a syntactic dependency tree, which is computed as follows.

If t -th watermark bit of the sentence s is embedded by morphological division/combination,

$$syn_wm(s, t) := \begin{cases} 1 & \text{if the node for } t\text{-th watermark bit} \\ & \text{is a chunk node in the syntactic} \\ & \text{tree of } s, \\ 0 & \text{otherwise, if the node for } t\text{-th} \\ & \text{watermark bit is a normal node in} \\ & \text{the syntactic tree of } s. \end{cases}$$

Otherwise, if the t -th watermark bit is embedded by movement of a syntactic adverbial,

$$syn_wm(s, t) := \begin{cases} 1 & \text{if the syntactic adverbial for} \\ & t\text{-th watermark bit is the first} \\ & \text{child in its clause in the} \\ & \text{syntactic tree of } s, \\ 0 & \text{otherwise} \end{cases}$$

If syn_wm is the same with the watermark bit, there is no change in the syntactic tree. Otherwise, the transformation of the syntactic tree is performed.

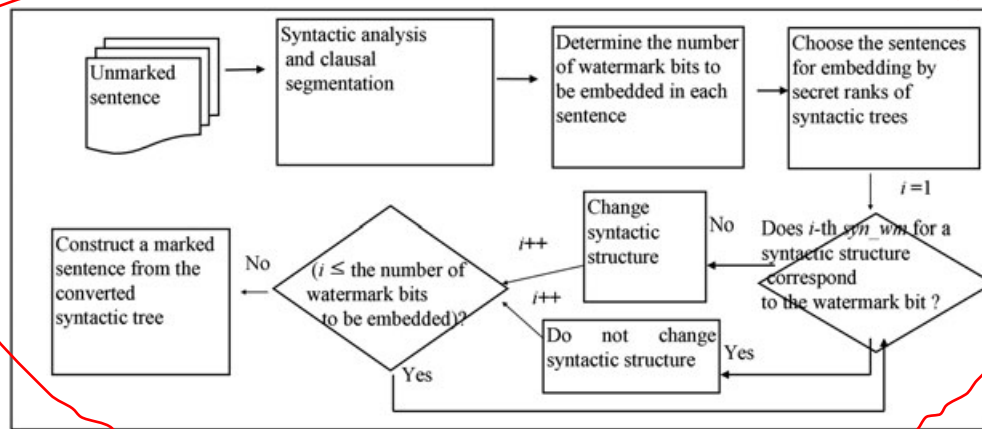


Figure 1. Text watermarking procedure.

The resulting textual document T_w is the finally marked document. After s_i was rewritten to s'_i , its literal expression was changed such that the watermark bits are “embedded”; meanwhile, the depths of tree nodes and the lengths of the words in the tree nodes do not change, and hence, the ranks of the rewritten sentences do not change.

To extract the embedded watermark, using the key K , an authorized person uses *Algorithm 2*.

Algorithm 2 Watermark extraction.

- 1: **Input:** K, T_w
 - 2: **Output:** Watermark bits $W = \{W_1, \dots, W_n\}$
 - 3: Compute $R_i = h1(K, \text{syn_rank}(s_i))$ for each sentence in T_w .
 - 4: Determine the number of embedded watermark bits for each sentence by the characteristic of its dependency tree.
 - 5: Obtain n -bit string W from the sentences with least R_i values using *syn_wm* functions.
-

We will explain in detail how we dynamically determine the number of embedded watermark bits and the two kinds of transformation of syntactic trees in the next subsection.

2.2. Syntactic dependency parsing and clausal segmentation

We performed text watermarking for Korean, which is a representative agglutinative language.

A syntactic dependency parser was used to determine the syntactic relation between words in a sentence. To obtain a syntactic dependency tree, we used the Korean syntactic parser of Kim *et al.* [8]. Figure 2 shows an example of a syntactic dependency tree. In that tree, a parent node functions as the syntactic governor of its child nodes, and each child node functions as the syntactic dependent of its parent node.

Although the agglutinative languages allow relatively free word order, the boundaries within which a word can

move are still constrained, but we can move a word within the clause in which it belongs. To determine the scope of possible moves, we performed clausal segmentation after syntactic dependency parsing.

The clausal segmentation procedure is as follows. First, we automatically detected all the predicates in the obtained parse tree. Then, we constructed one clause for each predicate by including all the child nodes of a predicate node. In Figure 2, the oval nodes indicate predicates. Because five predicates exist in Figure 2, we can obtain five clauses from the syntactic tree. The clauses are shown in Figure 3.

We then calculated the number of watermark bits to be embedded according to the syntactic tree. For syntactic constituent movement, we chose the constituents that satisfy the conditions in Section 2.3. For morphological division/combination, we chose nominal predicate nodes explained in Section 2.4. Finally, the number of constituents that are chosen for movement and that of nominal predicates become the number of watermark bits to be embedded in the sentence.

2.3. Watermarking by the movement of a syntactic constituent

We selected a syntactic constituent from each clause that meets the following target constituent conditions.

- (1) It is an adverbial.
- (2) It should have at least one sibling that is not an adverbial.
- (3) It is not a conjunctive adverbial.
- (4) Its syntactic governor is a predicate, not noun/adverb/etc.
- (5) A comma is not attached to the adverbial.
- (6) Its final ending is not a topical marker.

Amongst syntactic constituents, adverbials can move more freely in a sentence without semantic distortion [9]. Therefore, we chose adverbial constituents from a

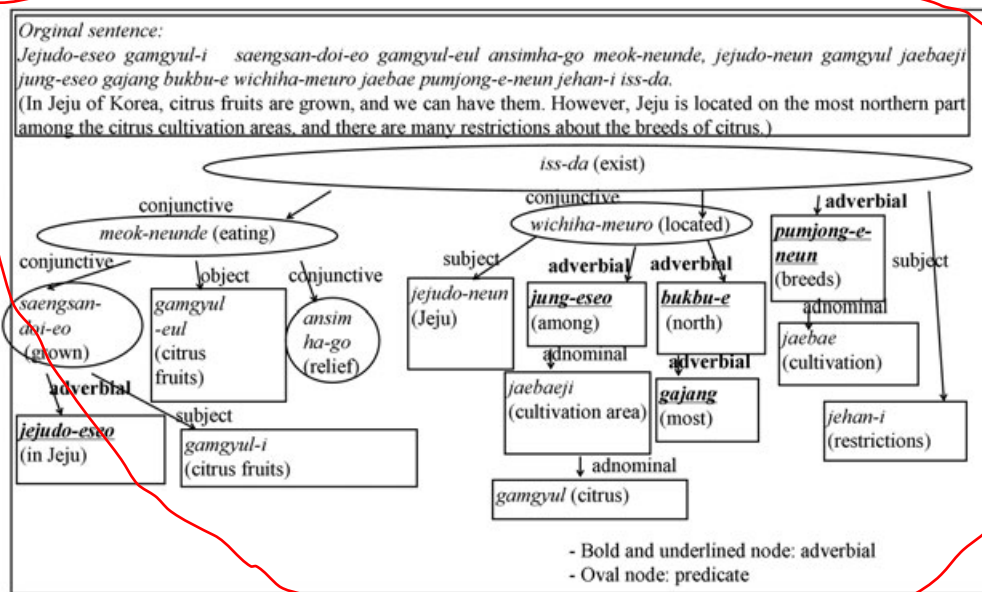


Figure 2. Example of a syntactic dependency tree.

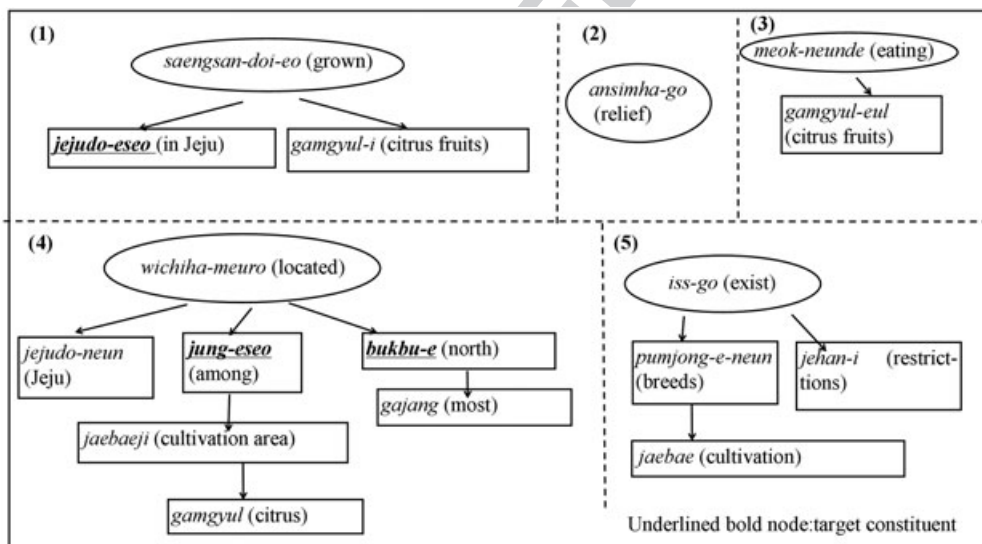


Figure 3. Clausal segmentation from a syntactic dependency tree.

syntactic tree as target constituents for movement. Second, it should have at least one sibling to swap positions. If the sibling is also an adverbial, the interchange between two adverbials does not give any hint whether an adverbial is moved or not. Therefore, a target adverbial should have at least one sibling that is not an adverbial. Third, a conjunctive adverbial, exceptionally, is located in the fixed position in a sentence. Therefore, it is not permitted to move freely. Moreover, when an adverbial's syntactic governor is not a predicate but a noun/adverb/sentence/etc., the position of the adverbial adds some semantic nuance to the meaning of the sentence. Lastly, in the case

that a comma is attached to an adverbial, or the final ending of an adverbial is a topical marker, the position of the adverbial also tends to indicate some semantic nuance. When the position of an adverbial indicates some semantic nuance, we excluded these types of adverbials from the target candidates. Because our purpose is to move one target constituent into the first position among its siblings, we do not need two target constituents in a clause. If more than one target constituent candidate exists in a clause, we randomly selected one.

In Figure 2, the underlined bold words indicate those that function as adverbials. Five adverbials exist in Figure 2.

Amongst them, some adverbials are excluded from the set of target constituents. For example, the adverbial “*gajang* (most)” is excluded from the target constituents because its governor is not a predicate but a noun.

According to *syn_wm* in Section 2.1, if the watermark bit is 1, we moved the syntactic adverbial into the first position in its clause. Otherwise, no adverbial should be in the first position.

F4 Figure 4 shows the movement for the target constituent “*jung-eseo* (among)”. If the watermark bit is 1, then the constituent is moved. When moving a target adverbial node, we should also move all its children. For example, in Figure 4, the nodes “*jaebaeji* (cultivation area)” and “*gamgyul* (citrus)” are child nodes of the target adverbial node “*jung-eseo* (among)”, so these were also moved. The converted tree is shown in the right side of Figure 4. After

completing target adverbial movement, we finally obtained the converted parse tree as shown in Figure 5.

F5

In the extreme case, even though all moved adverbials appear first in their clauses, it is difficult to find out watermark patterns in the marked sentences, because one sentence consists of many clauses, and an adverbial does not appear first in the surface form of the sentence if the adverbial has child nodes.

2.4. Watermarking by morphological division/combination

As mentioned, because Korean is an agglutinative language, a single word consists of several morphemes. Typically, one word consists of a content morpheme and a function morpheme. However, some words have more than one

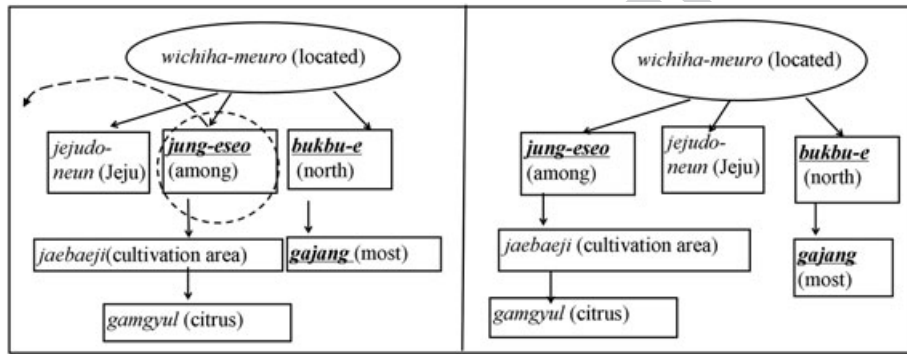


Figure 4. Example of syntactic constituent movement.

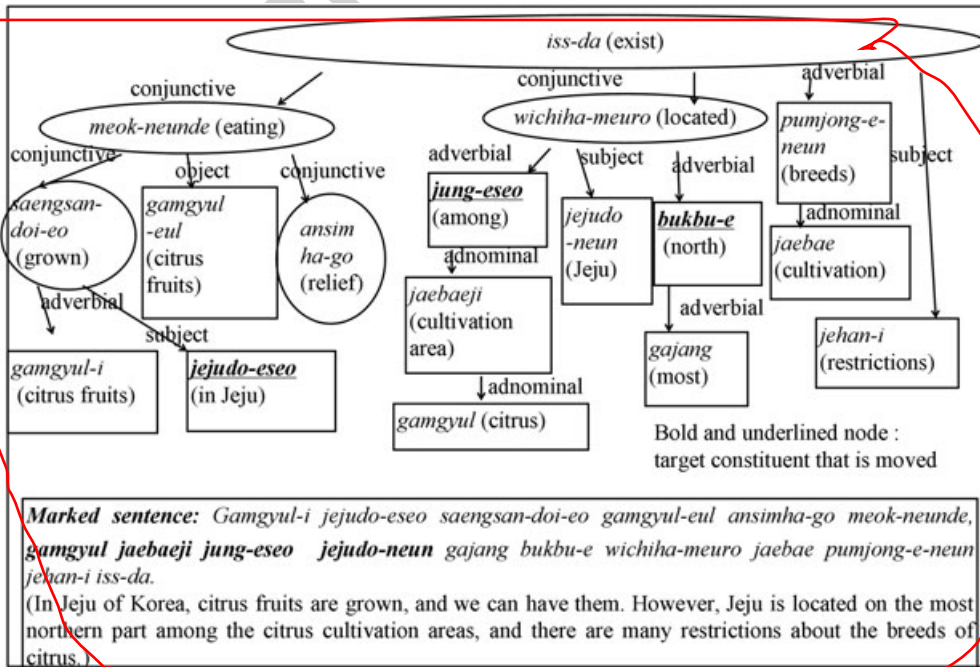


Figure 5. Final converted tree and marked sentence.

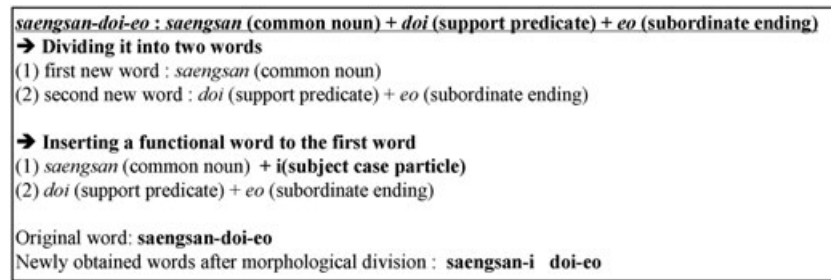


Figure 6. Example of morphological division.

content morpheme. Therefore, words that have more than one content morpheme are divided into new words.

In Korean, a predicate nominal has two content morphemes, a nominal and a predicate. We divided a predicate nominal into two new words and inserted a function morpheme for the first new word that does not have a function morpheme. An example is shown in Figure 6. A predicate nominal consists of “content morpheme1 (nominal) + content morpheme2 (support predicate) + function morpheme”. In the example of Figure 6, a subordinate ending is used as a function morpheme. As shown in Figure 6, after division, the first new word consists of only a nominal, and the second new word consists of a predicate and a function morpheme. Subsequently, the first word does not have a function morpheme. Therefore, we inserted a relevant function morpheme based on the relationship between the nominal and the predicate, determined as follows.

- (1) If the support predicate indicates active voice (e.g. “*ha*”, “*siki*”), then the nominal functions as the object of the predicate. Therefore, we inserted an object case particle, “*eul*” or “*reul*”.
- (2) If the support predicate indicates passive voice (e.g. “*doi*”), then the nominal functions as the subject of the predicate. Therefore, we inserted a subject case particle, “*i*” or “*ga*”.

In Figure 6, we can see that a subject case particle is inserted into the first word because the support predicate (“*doi*”) indicates a passive voice. Let us perform morphological division for the three nominal predicates indicated in Figure 2. Finally, the example tree after morphological division is shown in Figure 7, and it shows that all the watermark bits for these three nominal predicates are 1. To guarantee $syn_rank(s) = syn_rank(s')$, a single node was assigned for the divided two words by chunking as shown in Figure 7, and the morphological division/combination does not change the syntactic structure. To make the word length unchanged after morphological division, word length for a chunk node was calculated as “word length1 + word length2 – 1”. Because one function morpheme has been added, we subtracted 1.

The morphological combination process is simply the reverse process. We selected a chunk node from a syntactic tree: if the watermark bit is 0, we combined the

two words in the chunk node into one new word, by deleting the function morpheme in the first word. We then removed the chunk node and inserted a predicate normal node with the new word.

We adjusted the process if *syn_wm* in Section 2.1 does not correspond to the watermark bit.

2.5. Generation of a marked sentence

We then recombined the nodes of a parse tree and obtained a marked sentence as described in the bottom of Figure 5.

3. EXPERIMENTAL RESULTS

3.1. Performance of our system

We evaluated the performance of our system on perceptibility, robustness, and capacity by the following methods.

- (1) Coverage of marked sentences (capacity)
- (2) Naturalness of marked sentences (perceptibility)
- (3) Information-hiding capacity (capacity)
- (4) Resistance to attack (robustness)

For evaluation, we used 3025 declarative sentences and 1975 news sentences in the corpus of Matec99 (Morphological Analyzer and Tagger Evaluation Contest in Korean). As shown in Table I, the average number of words/sentence is 18.24 for declarative sentences and 11.00 for news. To see whether our method works well even for short sentences, we also performed watermarking for short news sentences.

To measure the distortion of marked sentences, we used a subjective evaluation by humans, as done by Meral *et al.* [1]. Their evaluation method has humans examine the texts and record their editing actions. The subjects were given marked texts and asked to edit them for improved intelligibility and style. This is a blind test because the subjects were not aware that text watermarking has taken place. In our study, three people checked for unnatural sentences. We also computed the information-hiding capacity for our method, and the experiments were executed on an AMD Phenom II X3 710 2.6GHz CPU with 8 GB of memory in the Ubuntu Linux environment. The results are shown below.

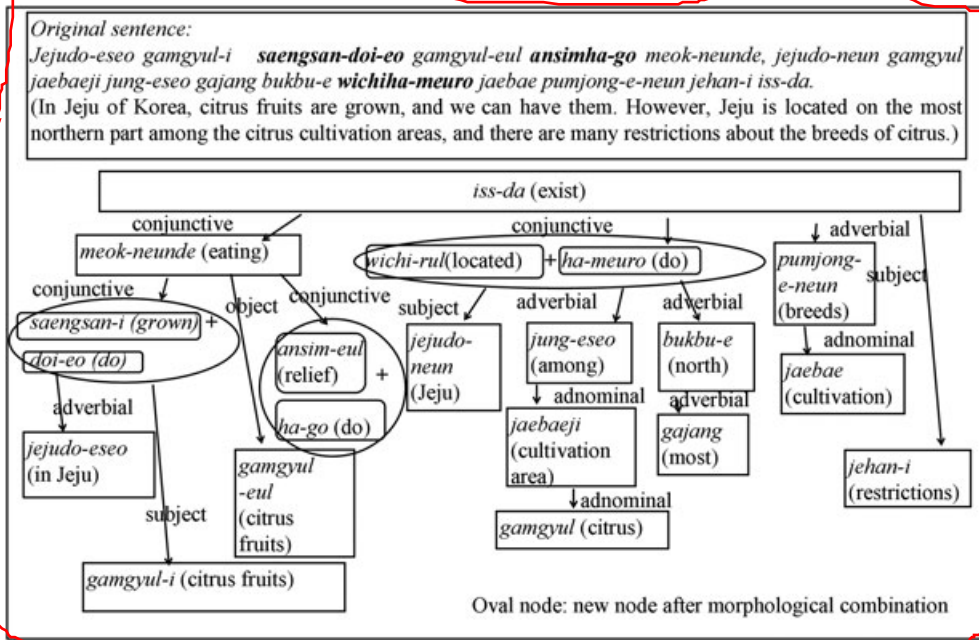


Figure 7. Example of morphological combination.

Table I. Edit rate of our system.

| | News | Declarative sentences | Overall |
|---|--------|-----------------------|---------|
| The number of sentences | 1975 | 3025 | 5000 |
| Average number of words/sentence | 11.00 | 18.24 | 15.38 |
| Sentences that can be watermarked | 90.25% | 100% | 96.15% |
| Edit rate of marked sentences | 13.62% | 16.93% | |
| Edit rate of original sentences before watermarking | 12.48% | 17.86% | |

- T3 (1) The relation of information-hiding capacity is 1:5.41 (see Table III).
- T2 (2) Average 2.84 bit/sentence of watermark is embedded. (Table II)
- (3) The coverage about the sentences selected for embedding watermark bit is 96.15% (see Table I).
- (4) For the declarative sentences, the marked sentences showed lower edit rate (see Table I).
- (5) All the watermark bits are correctly extracted.
- (6) This watermark embedding is robust in sentence movement, retyping, printing, and font change.

Table II. Comparison of performances for information-hiding capacity.

| Systems | Information-hiding capacity (bit/sentence) |
|---|--|
| Our system | 2.84 |
| Topkara <i>et al.</i> [3] | 0.67 |
| Atallah <i>et al.</i> [6], Topkara <i>et al.</i> [7] | 0.5 |
| Stutsman <i>et al.</i> [11] | 0.33 |
| Meral <i>et al.</i> [12] | 0.81 |
| Murphy <i>et al.</i> [13] | 0.31 |

3.2. Discussion

Table III shows the ratio of information-hiding capacity, which means for how many words of text, one bit of watermark can be hidden. The ratio of information-hiding capacity of our approach is 1:5.41, which is an improvement over that of Chiang *et al.* [10].

Table I shows the rate of unsuitable sentences among marked sentences and that among original sentences. After watermarking, the naturalness of marked sentences was

reduced in short news sentences. For news sentences, whereas the morphological division/combination did not cause any unnaturalness in short sentences, the movement

of adverbial sometimes caused unnaturalness. However, for longer sentences, we found that the naturalness of marked sentences was even higher than original sentences. Table I also shows that the coverage of our method is 96.15%.

In our system, the information-hiding capacity is 2.84 bits/sentence outperforming all the previous methods as shown in Table II. In previous methods, Topkara *et al.* [3] showed capacity of 0.67 bit/sentence using lexical substitution, and Atallah *et al.* [6] and Topkara *et al.* [7] embedded average 0.5 bits/sentence of watermarks using a syntactic method. Stutsman *et al.* [11] used a machine translation method for watermarking and showed 0.33 bits/sentence of information-hiding capacity. Meral *et al.* [12] embedded 0.81 bits/sentence of watermarks using morphosyntactic alteration, and Murphy *et al.* [13] embedded 0.31 bits/sentence of watermarks using syntactic transformations. Whereas the previous systems used one method for watermark embedding, our system uses both syntactic movement and morphological division/combination, and it adaptively determines watermark bits to be embedded for each sentence according to the characteristics of its syntactic structure. Therefore, we reduced the failure rate of watermark embedding and maximized the information-hiding capacity.

The more a system ensures high information-hiding capacity, the less robust it may be to adversary attacks. To ensure high robustness, all the watermarking processes were performed based on syntactic dependency trees. Therefore, our system becomes more robust by preventing an attacker from finding out watermarking patterns from surface forms of the sentences.

Given T_w , the adversary can compute syn_wm for each sentence. The string $\{syn_wm(s)\}$ includes the watermark. For sentences not rewritten by the watermark embedding process, their syn_wm value can be regarded as a random bit. Therefore, given T_w , the adversary cannot tell what is the watermark, which sentences carry the watermark, nor can the attacker infer any information about the key K .

Our method can withstand printing and font changing, **Q2** all optical character recognition techniques, retyping, and moving/switching sentences. However, it is not resistant to sentence insertion/deletion. We therefore need to add an error correction code to cope with this kind of attack, in future work.

We conclude that our natural language watermarking based on syntactic dependency trees has satisfying performance without semantic and stylistic distortion.

4. RELATED WORK

Atallah *et al.* [6,14] proposed a technique for information hiding in natural language text. They established the basic technique for embedding a resilient watermark in text by combining a number of information assurance and security techniques with the advanced methods and resources of natural language processing. A semantics-based scheme significantly improves the information-hiding capacity of

English text by modifying the granularity of meaning of individual terms/sentences. However, this scheme is suitable only for English, and it was merely conceptual.

In other studies, a technique of embedding secret data is proposed, which has no major impact on the meaning of a text, and operates by replacing words in the cover text with synonyms [3–5]. However, there is deterioration in documents in which importance is attached to delicate nuance when synonyms have been substituted. In addition, because of domain dependence, the selection of suitable synonyms is not easy. Moreover, the method needs a large synonymy dictionary and a huge collocation database [15].

Grothoff *et al.* [16] suggested the insertion of plausible mistakes, in addition to synonym substitution. However, this approach does not ensure robustness, because the watermarked sentences have mistakes, and thus, they can be detected by attackers.

There are several other methods suggested for agglutinative languages. Meral *et al.* [1] proposed 21 syntactic tools for Turkish text watermarking. However, they do not have a sentence selection process based on syntactic trees, and an authorized person has to estimate the applied syntactic transformations to extract the watermark. Topkara *et al.* [7] also proposed syntax-based natural language watermarking using syntactic sentence paraphrasing. They insisted that the syntactic approach is useful for natural language watermarking without semantic distortion. Kim *et al.* [17] proposed the use of an adverbial displacement in a sentence. That method shows good coverage and naturalness of modified sentences, but information-hiding capacity is low.

We conclude that our extended method for text watermarking based on syntactic trees is more effective. Whereas the previous methods used only syntactic structure for sentence transformation, we used it for the overall process to ensure more robustness. We selected sentences to be watermarked based on their syntactic trees, and this ensures that sentence movement by attackers does not damage the watermark. In the same way, an authorized person extracts a watermark from syntactic dependency trees. To ensure high capacity, we adaptively determined the number of watermark bits to be embedded in each sentence.

5. SUMMARY AND CONCLUSION

We proposed natural language watermarking for agglutinative languages, based on syntactic dependency trees. By exploiting characteristics of agglutinative languages, we embedded watermark into sentences using syntactic constituent movement, as well as morphological division/combination. The overall procedure consists of several steps. First, we performed syntactic dependency parsing of the original text. Next, we obtained clauses by segmenting the syntactic tree. Then, we dynamically determined the number of watermark bits to be embedded

for each sentence based on its syntactic tree. Fourth, we chose the sentences to be marked according to their secret ranks. Then, we embedded a watermark bit by morphological division/combination, and syntactic constituent movement. Finally, we obtained marked text from the modified parse trees.

The experimental results showed that the coverage of our method is 96.15% and the ratio of information-hiding capacity is 1:5.41, which outperforms previous systems. We also achieved an information-hiding capacity with 2.84 bits/sentence, which is the highest amongst existing watermarking systems. After watermarking, the naturalness of marked sentences was reduced in short news sentences. For the whole collection of news sentences, although the morphological division/combination did not cause any unnaturalness in short sentences, the movement of adverbial sometimes caused unnaturalness. However, for longer sentences, we found that the naturalness of marked sentences was even higher than original sentences. By using syntactic structures, our method achieves the highest capacity compared with previous systems and is resistant to sentence movement, retyping, printing, and font changes.

We conclude that our syntactic tree-based method is useful in watermarking of agglutinative languages. Other types of languages, such as Indo-European languages, also share similar characteristics to which our method can apply. For example, some constituents can move freely in limited boundaries. By using those characteristics, we can attempt to demonstrate the effectiveness of our method to other languages. In addition, we need to explore how to maintain robustness in sentence addition/deletion by attackers and to improve the error correction codes.

REFERENCES

1. Meral HM, Sevinc E, Unkar E, Sankur B, Ozsoy AS, Gungor T. Syntactic tools for text watermarking. In *Proceedings of the SPIE International Conference on Security, Steganography, and Watermarking of Multimedia Contents*, Vol. **6505**, Delp EJ, III, Wong PW (eds). SPIE: Bellingham, WA, 2007. Q3
2. Cox I, Miller ML, Bloom JA, Kaufman M. *Digital Watermarking*. Morgan Kaufmann Publishers Inc.: San Francisco, CA, 2002.
3. Topkara M, Taskiran CM, Delp EJ. Natural language watermarking. In *Proceedings of the SPIE International Conference on Security, Steganography and Watermarking of Multimedia Contents*, Vol. **5681**. Delp EJ, III, Wong PW (eds). SPIE: Bellingham, WA, 2005; 441–452.
4. Taskiran CM, Topkara M, Delp EJ. Attacks on linguistic steganography systems using text analysis. In *Proceedings of the SPIE International Conference on Security, Steganography and Watermarking of Multimedia Contents*. Delp EJ, III, Wong PW (eds). SPIE: Bellingham, WA, 2006; 313–336.
5. Topkara U, Topkara M, Atallah MJ. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. *Proceedings of the ACM Multimedia and Security Conference*. ACM: New York, NY, 2006.
6. Atallah MJ, Raskin V, Crogan M, Hempelmann C, Kerschbaum F, Mohamed D, Naik S. Natural language watermarking: design, analysis, and proof-of-concept implementation. *Proceedings of the 4th International Information Hiding Workshop*, Pittsburgh, Pennsylvania, 25–27 April 2001.
7. Topkara M, Topkara U, Atallah MJ. Words are not enough: sentence level natural language watermarking. *Proceedings of 4th ACM International Proceedings of ACM Workshop on Content Protection and Security* (in conjunction with ACM Multimedia). ACM: New York, NY, 2006.
8. Kim MY, Kang SJ, Lee JH. Resolving ambiguity in inter-chunk dependency parsing. In *Proceedings of NLPRS*, 2001; 263–270.
9. Kwon JI. The study of Korean grammar. *Bak-I-Jeong*, 1994. Q4
10. Chiang YL, Chang LP, Hsieh WT, Chen WC. Natural language watermarking using semantic substitution for Chinese text. *Lecture Notes in Computer Science* 2004; 129–140.
11. Stutsman R, Atallah MJ, Grothoff C, Grothoff K. Lost in just the translation. *Proceedings of the Annual Symposium on Applied Computing (SAC)*. ACM: New York, NY, 2006; 338–345.
12. Meral HM, Sankur B, Ozsoy AS, Gungor T, Sevinc E. Natural language watermarking via morphosyntactic alterations. *Computer Speech & Language* 2009; **23**:107–125.
13. Murphy B, Vogel C. The syntax of concealment: reliable methods for plain text information hiding. In *Proceedings of the SPIE International Conference on Security, Steganography and Watermarking of Multimedia Contents*, Vol. **6505**, Delp EJ, III, Wong PW (eds). SPIE: Bellingham, WA, 2007; 6505Y.1–6505Y.12
14. Atallah M, Raskin V, Hempelmann CF, Karahan M, Sion R, Triezenberg KE, Topkara U. Natural language watermarking and tamperproofing. In *Lecture Notes in Computer Science, Proceedings of the 5th International Information Hiding Workshop*. Springer: Berlin/Heidelberg, 2002; 7–9.
15. Takizawa O, Makino K, Matsumoto T, Nakagawa H, Murase I. Method of hiding information in agglutinative language documents using adjustment to new

- line positions. *Knowledge-based and Intelligent Engineering Systems* 2005; **3**: 1039–1048.
16. Grothoff C, Grothoff K, Alkhutova L, Stutsman R, Atallah M. Translation-based steganography. *Proceedings of Information Hiding Workshop (IH 2005)*, Barcelona, Spain, 2005.
17. Kim MY. Natural language watermarking for korean using adverbial displacement. *International Workshop on Interactive Multimedia and Intelligent Services in Mobile and Ubiquitous Computing (IMIS)*. IEEE Computer Society: Washington, DC, 2008.

UNCORRECTED PROOF

Author Query Form

Journal: Security and Communication Networks







Article: sec_336

Dear Author,

During the copyediting of your paper, the following queries arose. Please respond to these by annotating your proofs with the necessary changes/additions.

- If you intend to annotate your proof electronically, please refer to the E-annotation guidelines.
- If you intend to annotate your proof by means of hard-copy mark-up, please refer to the proof mark-up symbols guidelines. If manually writing corrections on your proof and returning it by fax, do not write too close to the edge of the paper. Please remember that illegible mark-ups may delay publication.

Whether you opt for hard-copy or electronic annotation of your proofs, we recommend that you provide additional clarification of answers to queries by entering your answers on the query sheet, in addition to the text mark-up.

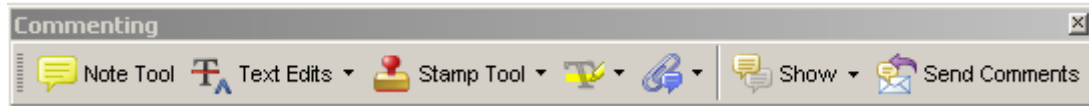
| Query No. | Query | Remark |
|-----------|---|---|
| Q1 | AUTHOR: Please provide a suitable figure (abstract diagram or illustration selected from the manuscript or an additional eye-catching figure) and a short 'GTOC' abstract (maximum 80 words or 3 sentences) summarising the key findings presented in the paper for Table of Content (TOC) entry. |  |
| Q2 | AUTHOR: 'Optical character recognition'. Is this the correct definition for 'OCR'? Please change if incorrect. |  |
| Q3 | AUTHOR: Please check and confirm that References 1, 3, 4, 6, 13, and 14 have been presented correctly. |  |
| Q4 | AUTHOR: If Reference 9 is a book, please provide publisher name and publisher location. (Author. Book Title. Publisher name: Publisher location, Publication year.). |  |
| Q5 | AUTHOR: Figures 1,2,5 & 7 contains overlapping data. Please correct and resupply the figure. Please check required artwork specifications at http://www.blackwellpublishing.com/authors/digill.asp |  |
| Q6 | AUTHOR: Please check the suitability of the suggested short title. |  |

USING E-ANNOTATION TOOLS FOR ELECTRONIC PROOF CORRECTION

Required Software

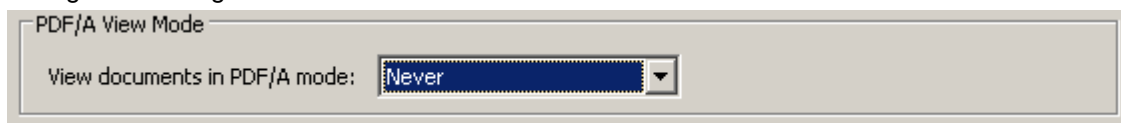
Adobe Acrobat Professional or Acrobat Reader (version 7.0 or above) is required to e-annotate PDFs. Acrobat 8 Reader is a free download: <http://www.adobe.com/products/acrobat/readstep2.html>

Once you have Acrobat Reader 8 on your PC and open the proof, you will see the Commenting Toolbar (if it does not appear automatically go to Tools>Commenting>Commenting Toolbar). The Commenting Toolbar looks like this:



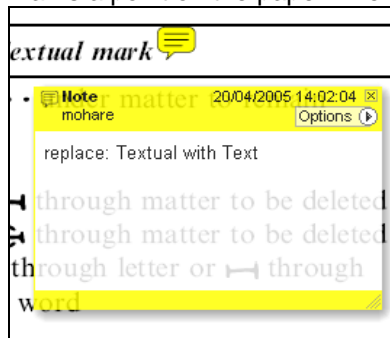
If you experience problems annotating files in Adobe Acrobat Reader 9 then you may need to change a preference setting in order to edit.

In the "Documents" category under "Edit – Preferences", please select the category 'Documents' and change the setting "PDF/A mode:" to "Never".



Note Tool — For making notes at specific points in the text

Marks a point on the paper where a note or question needs to be addressed.

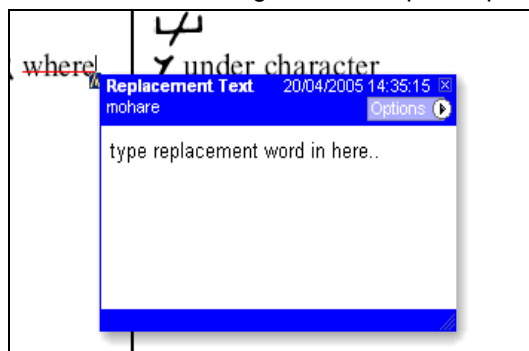


How to use it:

1. Right click into area of either inserted text or relevance to note
2. Select Add Note and a yellow speech bubble symbol and text box will appear
3. Type comment into the text box
4. Click the X in the top right hand corner of the note box to close.

Replacement text tool — For deleting one word/section of text and replacing it

Strikes red line through text and opens up a replacement text box.

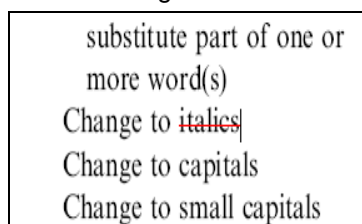


How to use it:

1. Select cursor from toolbar
2. Highlight word or sentence
3. Right click
4. Select Replace Text (Comment) option
5. Type replacement text in blue box
6. Click outside of the blue box to close

Cross out text tool — For deleting text when there is nothing to replace selection

Strikes through text in a red line.



How to use it:

1. Select cursor from toolbar
2. Highlight word or sentence
3. Right click
4. Select Cross Out Text

Approved tool — For approving a proof and that no corrections at all are required.

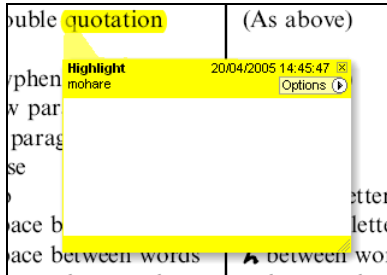


How to use it:

1. Click on the Stamp Tool in the toolbar
2. Select the Approved rubber stamp from the 'standard business' selection
3. Click on the text where you want to rubber stamp to appear (usually first page)

Highlight tool — For highlighting selection that should be changed to bold or italic.

Highlights text in yellow and opens up a text box.

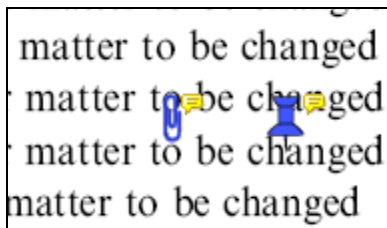


How to use it:

1. Select Highlighter Tool from the commenting toolbar
2. Highlight the desired text
3. Add a note detailing the required change

Attach File Tool — For inserting large amounts of text or replacement figures as a files.

Inserts symbol and speech bubble where a file has been inserted.

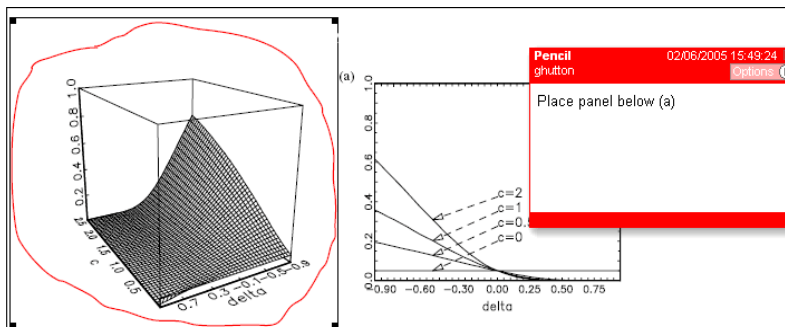


How to use it:

1. Click on paperclip icon in the commenting toolbar
2. Click where you want to insert the attachment
3. Select the saved file from your PC/network
4. Select appearance of icon (paperclip, graph, attachment or tag) and close

Pencil tool — For circling parts of figures or making freeform marks

Creates freeform shapes with a pencil tool. Particularly with graphics within the proof it may be useful to use the Drawing Markups toolbar. These tools allow you to draw circles, lines and comment on these marks.



How to use it:

1. Select Tools > Drawing Markups > Pencil Tool
2. Draw with the cursor
3. Multiple pieces of pencil annotation can be grouped together
4. Once finished, move the cursor over the shape until an arrowhead appears and right click
5. Select Open Pop-Up Note and type in a details of required change
6. Click the X in the top right hand corner of the note box to close.

Help

For further information on how to annotate proofs click on the Help button to activate a list of instructions:

