Co-reference Resolution for Thai Short Stories

Ponrudee Netisopakul **Patipan Wikaha**

Knowledge Management and Knowledge Engineering Laboratory
Faculty of Information Technology
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
Email: ponrudee@it.kmitl.ac.th, patipan.x@gmail.com

Abstract

A task of identifying speakers from text is usually done by selecting a right speaker from a list of actors in a story. This actor list is called an avatar list, which so far, have been manually created for each story. This paper proposes a new methodology to identify speaker without using an avatar list. This can be achieved by incorporating multiple natural language processing techniques. Those are noun phrase chunking, speaker candidate extraction using speech verbs, and heuristic rules for identifying the speaker of each quote text.

Keywords: speaker identification, multiple children stories, text annotation, speech verbs

1 Motivation

In early 2012, a group of instructors and researchers at Faculty of Information Technology, KMITL initiated a project called ESSVIBS – Emotional Speech Synthesis for Visibility Impaired: From-Book-to-Speech [1]. The project is meant as a framework to explore a number of interesting issues related to speech and text processing, including Thai OCR, natural language text understanding, machine learning and speech synthesis.

In brief, input to ESSVIBS are Thai short children story books, output is a computer generated speech with multiple voices representing males, females, children and adults voices. At the initial state, forty Thai children stories are collected for experiments.

One of problems which is a focus in this paper is a problem of marking speakers in a story. We have examined some related works in a field of speaker identification from text [2, 3, 4, 5] and found that:

1. Most of the researches work with a long story with leading actor/actress and supporting actors/actresses.
2. For most of the work above except one, before a speaker can be identified, a list of actors/actresses above must be manually created and input to the speaker identification process – this list is called *an avatar list*.
3. Only the work in [2] identified actors using name entity extraction.

At this point, it is clear to us that an approach with an avatar list is inappropriate for our work because, first, we aim for our algorithm to work with a large collection of short stories, even with stories have not seen before. Second, Thai names in children stories are not very different from common noun phrases, hence, a name entity extraction may not work very well.

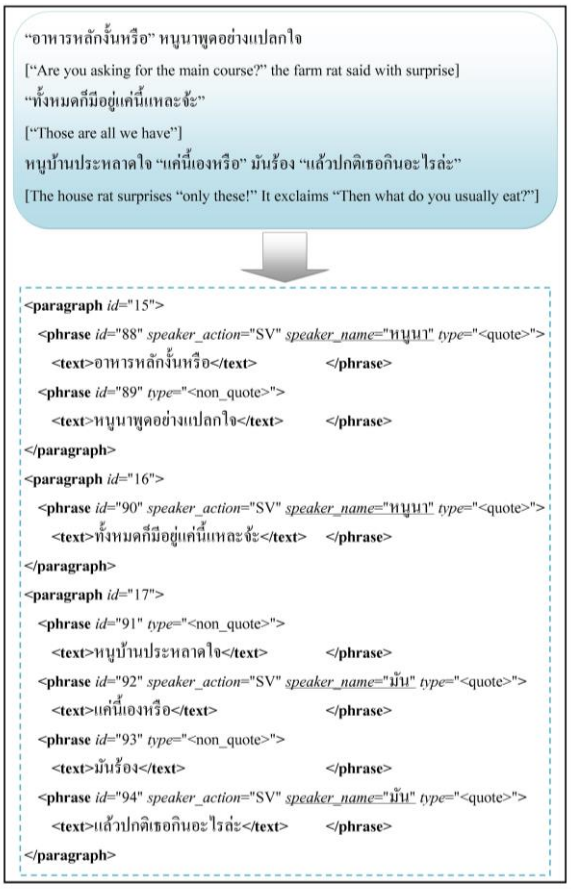
For these reasons, our research proposed a novel method to extract actors from multiple Thai short stories and using this list instead of an avatar to identify speakers.

Figure 1 shows an example input and desired output of our system - a Speaker Identification from Multiple Short Story without an Avatar – SIMS-woA. An input is excerpt from a short story. The brackets are English translation. An output is in xml format with speakers of each quote identified.

Hence, the research questions are:

- How to extract noun phrases which are actors in the story without extracting every common noun? This is an automatic avatar list construction task. When this is done, we can use the list to annotate the actors in the story. This is called an actor annotation task.
- How to correctly assign a speaker to each quote that appear in the story, especially when

there are more than one actor adjacent to the quote or when there is no actor in the same paragraph of the quote? This is called a speaker assignment task.


“อาหารหลักงั้นหรือ” หนูนาพูดอย่างแปลกใจ

[“Are you asking for the main course?” the farm rat said with surprise]

“ทั้งหมดก็มีอยู่แค่นี้แหละจ้ะ”

[“Those are all we have”]

หนูบ้านประหลาดใจ “แค่นี้เองหรือ” มันร้อง “แล้วปกติเธอกินอะไรล่ะ”

[The house rat surprises “only these!” It exclaims “Then what do you usually eat?”]

```
<paragraph id="15">
  <phrase id="88" speaker_action="SV" speaker_name="หนูนา" type="<quote>">
    <text>อาหารหลักงั้นหรือ</text>          </phrase>
  <phrase id="89" type="<non_quote>">
    <text>หนูนาพูดอย่างแปลกใจ</text>        </phrase>
</paragraph>
<paragraph id="16">
  <phrase id="90" speaker_action="SV" speaker_name="หนูนา" type="<quote>">
    <text>ทั้งหมดก็มีอยู่แค่นี้แหละจ้ะ</text>    </phrase>
</paragraph>
<paragraph id="17">
  <phrase id="91" type="<non_quote>">
    <text>หนูบ้านประหลาดใจ</text>           </phrase>
  <phrase id="92" speaker_action="SV" speaker_name="มัน" type="<quote>">
    <text>แค่นี้เองหรือ</text>              </phrase>
  <phrase id="93" type="<non_quote>">
    <text>มันร้อง</text>                   </phrase>
  <phrase id="94" speaker_action="SV" speaker_name="มัน" type="<quote>">
    <text>แล้วปกติเธอกินอะไรล่ะ</text>       </phrase>
</paragraph>
```

Figure 1 Example of input and desired output

The outline of this paper is as follow. Section 2 reviews approaches by previous research. Section 3 explains in detail our proposed methodology. Section 4 explains experiments and results of evaluating our approach. Conclusion and future work are presented in section 5.

2 Related Work

There are common processes for the task of identifying speakers from text [2, 3, 4]. First, quote texts are identified then actors are identified. Finally, speaker of each quote are identified. The differences among these researches are mainly the methods used to identifying actors. The work in [2] used pattern matching to extract proper names (or character names in the story) from the whole text. Assuming that the speaker must be in the same paragraph as the quote, [2] assigned a speaker from a preceding phrase or a following phrase of the quote. This work resolved neither a missing speaker issue when no character name is found around the quote, nor a multiple candidate speaker issue when there are more than one character names found around the quote.

The work in [3] created hierarchical phrase structure of each paragraph, beginning by separating quote text from narrative text. Narrative texts are further processed and tagged as noun phrase (subject), main verb, punctuation and so on. These structures are manually built from seeding text and used as input to learn (or merge) a set of generalized rules for finding speech verbs, actors and speakers, respectively. The paper showed that this approach did not obtain good result for a different author test set.

The work in [4] improved upon [3] by applying a scoring technique, for each phase of annotation, based on set of features. For example, features such as main verb, hypernyms, adjacent sentence, and proximity to quote are used for speech verb annotation. Features such as subject or object, noun or pronoun, proper noun, abbreviation and distance from verb are used for actor annotation. In addition, a hand-code decision tree is used to resolve speaker ambiguous.

Note that, even with these complicate processes in [3, 4], they both must still employ manually created avatar list to reduce errors when choosing speaker for each quote.

The work in [5] is the only speaker identification work found processing Thai language. Training set composed of adjacent quote phrases with POS tags and manually tagged actors and speakers. The learning features are “language model” and n-gram. The work can only identify speaker from a simple sentence. A speaker previously identified is not taken into account to identify a next speaker.

In our work, we improve upon previous works in the following aspects.

1. We classify verbs into 4 classes and prioritized them based on their possibility of being pointers to speakers.
2. When an avatar list is absent, an actor list can be automated created instead using verb classes as clues.
3. This actor list can be used to scan for candidate speakers of quote texts, when speech verbs are absent from those phrases.
4. Speaker selection decision is based on a sim-

ple verb class priority and proximity to quote text.

3 Proposed Methodology

Since our intended system must work with a number of short Thai children books. Each has its own set of characters, which conventionally must be manually identified for each story in advance. However, we propose that a labor work of creating an avatar list for each story can be avoided. An alternative method to automatically identify actors and speakers of multiple short stories without the need to manually create many avatar lists is laid out in detail in this section.

An overall idea is based on a heuristic that a noun phrase representing an actor who speaks usually appears adjacent to the quote, either in the same paragraph or in the adjacent paragraph, as shown in Figure 2. Therefore, the first main task is to identify a set of candidate actors from paragraphs with quote and adjacent to quote. This list can be used in place of an avatar list. Next main task is to identify only candidate speakers for each quote. Last task is to use a heuristic to assign a speaker of that quote from the previous list of candidates.

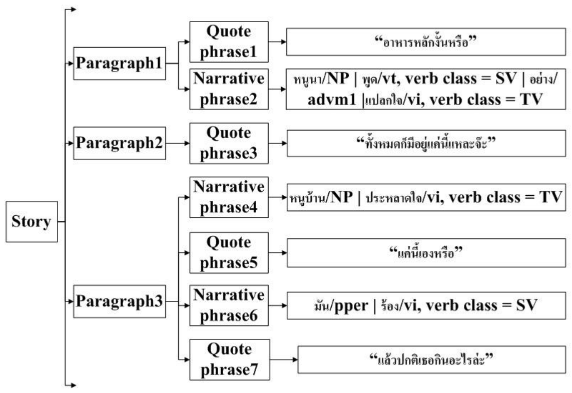


Figure 2 Hierarchical view of text structure in SIMS-woA

Figure 2 shows the hierarchical view of text structure, our system - SIMS-woA - working with. A story composes of many paragraphs; SIMSwoA focuses on paragraphs with quote text and adjacent to quote text. It is reasonable to assume that to find a speaker of a quote; the process should only look nearby the quote. It should not have to process every paragraph.

A paragraph may have both quote text and narrative text or it may have only quote text or narrative text. Figure 2 shows a paragraph with a quote text following by a narrative text. In paragraph 1, the quote text is “อาหารหลักงั้นหรือ” and the narrative text is “หนูนาพูดอย่างแปลกใจ”. In order to know who speak the quote text, first, the system identifies pairs of (noun phrase, verb class) which either precede or follow the quote text, within the same paragraph or in the adjacent paragraph. The noun phrases from these pairs are candidate actors for the quote. These candidates are prioritized based on their associate verb class.

To automatically construct a list of actors without using manually created avatar, one of the novelties of SIMS-woA is to classify verbs into four verb classes and prioritize them based on their possibilities to be an indicator for a candidate speaker.

These verb classes are speech verbs (SV), thinking/feeling verbs (TV), action verbs (AV), and other verbs. Table 1 shows the number of each verb classes. There are 43 speech verbs, 10 thinking/feeling verbs and 54 action verbs. Speech verbs are verbs indicating the action of saying, such as พูด (say) ว่า (said that) กล่าว (state) ประกาศ (pronounce) พึมพำ (mumble) บ่น (complain) ดุ (rebuke) ชม (praise). If a pair of (noun phrase, SV) is found next to a quote, it is pretty clear that the noun phrase has a high possibility of being a speaker of that quote. Therefore, this noun phrase is given the highest priority to be assigned as a speaker of the quote.

However, from our preliminary investigation, we found that a verb class SV alone is not enough to detect actors or speakers in the story. Many narrative phrases adjacent to quote texts do not have any explicit speech verb, but they do have either thinking verbs or action verbs. Thinking verbs are verbs indicate mental actions of actors, such as คิด(think) นึก(contemplate) รำพึง(cogitate) สงสัย(doubt). In a narrative children story, when an actor thinks, the quote text must also be read out loud. In this sense, a thinking verb class (TV) has a second highest priority to be assigned as a speaker.

Action verbs are verbs indicating actions of actors, such as ยิ้ม (smile) พยักหน้า (nod) เสนอ (offer). When telling a story, instead of a speech verb, an action verb is often place there to indicate tone, feeling or movement of the speaker. Hence, a noun phrase with an action verb class (AV) has the next priority for a speaker assignment. For those verbs not belong to these three verb classes, the process labeling them as ‘other’ verbs.

Table 1 The number of each verb classes

Classes	Count	Examples
SV	43	พูด (say), ว่า (said that), …
TV	10	คิด(think), สงสัย(doubt), …
AV	54	ยิ้ม (smile), พยักหน้า (nod), …
Other	-	คอย (wait), วิ่ง (ran), …

Note that list of verbs can be collected during preprocessing step but their associated verb class as defined here is currently done manually. We are investigating whether this can be done automatically using pre-labeled learning set.

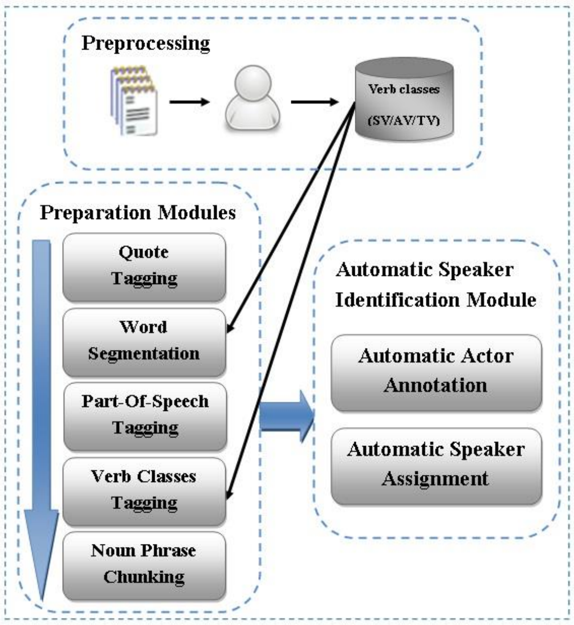


Figure 3 Overall architecture of SIMS-woA

Figure 3 shows an overall architecture of SIMS-woA. A preparation module on the left is an NLP pipeline to prepare Thai text for automatic speaker identification module on the right.

Input to the preparation module is a story in text form. First, quote text are marked everywhere and will not be processed further. The rest of the text goes through word segmentation and POS tagging using a publicly available Thai segmentation tool [6] and a Thai POS tagger [7,8]. Then, verbs are mapped into verb classes as previously explained.

One last important step during preparation is noun phrase chunking. We do not use name entity extraction technique to find an actor here because in most of children stories, actors are common nouns with attributes such as หนูนา (farm rat) หนูบ้าน (house rat) คนตัดต้นไม้ (woodsman). Therefore, more than a hundred regular expression rules, as partially shown in Figure 4, are used to construct a noun phrase from word tags.

For example, “หนู/ncn บ้านncn” are chunked into “หนูบ้าน/NP” and “คน/ncn” “ตัด/vt” “ต้นไม้/ncn” are chunked into “คนตัดต้นไม้/NP”.

```
NP      : {<ncn><ncn><ncn><cl><adj>}
        : {<ncn><prep><ncn><ncn>}
        : {<ncn><vt><ncn>}
        : {<ncn><ncn><adj>}
        : {<ncn><cl><adj>}
        : {<ncn><ncn>}
        : {<ncn><npn>}
        : {<ntit><ncn>}
        : {<npn>}
        : {<ncn>}
```

Figure 4 Example of regular expression rules for NP chunking

On the right of Figure 3, an automatic speaker identification module has two sub-modules: automatic actor annotation and automatic speaker assignment. For actor annotation, Figure 5 shows that there are two tasks for finding actors, who will be candidates for speakers. Those are actor extraction task and actor annotation task.

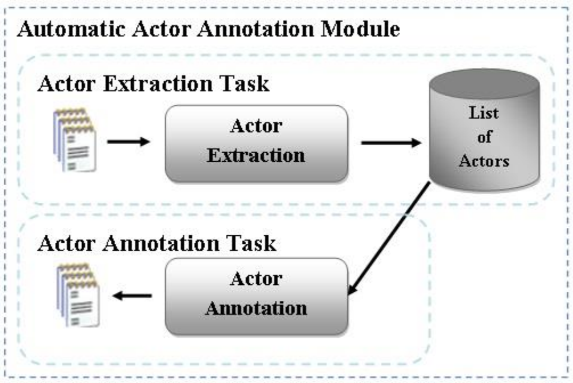
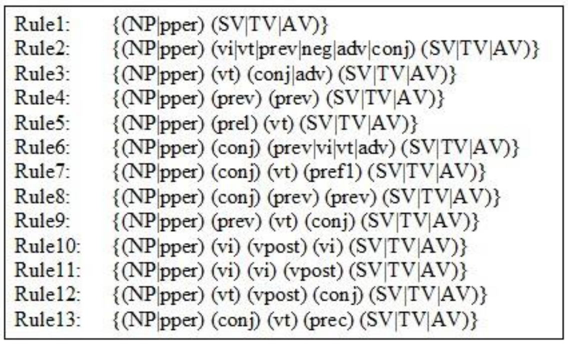


Figure 5 Automatic actor annotation task

Figure 6 are rules used to extract actors from phrases adjacent to a quote text. These rules match (partial) phrases with three verb classes previously described. Note that phrases with these verb classes mostly have only one noun phrase. However, there can be many adjacent phrases to the quote text. Therefore, there can be many actors, which need to be decided which one is most likely to be a speaker in a later module. For now, actors extracted are stored in an actor list to be used in an actor annotation task.



Rule1:	{(NP|pper) (SV|TV|AV)}
Rule2:	{(NP|pper) (vi|vt|prev|neg|adv|conj) (SV|TV|AV)}
Rule3:	{(NP|pper) (vt) (conj|adv) (SV|TV|AV)}
Rule4:	{(NP|pper) (prev) (prev) (SV|TV|AV)}
Rule5:	{(NP|pper) (prel) (vt) (SV|TV|AV)}
Rule6:	{(NP|pper) (conj) (prev|vi|vt|adv) (SV|TV|AV)}
Rule7:	{(NP|pper) (conj) (vt) (pref1) (SV|TV|AV)}
Rule8:	{(NP|pper) (conj) (prev) (prev) (SV|TV|AV)}
Rule9:	{(NP|pper) (prev) (vt) (conj) (SV|TV|AV)}
Rule10:	{(NP|pper) (vi) (vpost) (vi) (SV|TV|AV)}
Rule11:	{(NP|pper) (vi) (vi) (vpost) (SV|TV|AV)}
Rule12:	{(NP|pper) (vt) (vpost) (conj) (SV|TV|AV)}
Rule13:	{(NP|pper) (conj) (vt) (prec) (SV|TV|AV)}

Figure 6 Actor extraction rules

Sometimes there is no narrative phrase adjacent to a quote or those phrases do not matched rules above. Hence, the extracted actor list is used to annotate actors from other parts of text in the story. This is called actor annotation task. This step increases a chance of finding a right speaker. Note that a speaker of a particular quote may have been mentioned in a previous paragraph far from a quote or even in the beginning of the story.

Another task during actor annotation is to assign an action to each actor. This task is important because its verb class information will be used to priority actors for a speaker assignment sub-module. That is SV has a highest priority, then TV and AV.

Assigning an action to an actor is straightforward if there is only one actor and only one verb in the phrase. Then the action of the actor is the type of that verb class, such as SV, TV, AV or other verb. The task is more complicate for phrases with many actors or actions. For example, “มากินพายแอปเปิ้ลกัน” หนูบ้านพูดให้หนูนาคลายความกลัว, the action of “หนูบ้าน” is “พูด” and the action of “หนูนา” is “คลายความกลัว”.

The algorithm is shown in Figure 7. It processes a word token from left to right for each phrase. If other types of token except an actor and a verb are found, the algorithm just moves to the next token. If an actor is found with an action already assigned, it also skips to the next token; otherwise, a current actor is kept. Then a next token with of type verb is kept in a variable *best_action*; when a next verb token is found, its verb class priority is compared to *best_action*'s priority. The verb with highest priority replaces the old one and finally is assigned to be an action of the current actor, which happen either when a new actor is found or when a phrase is ended.

```
Actor's Action Assignment
Input: word token of a phrase
Output: association of (actor_i, action_i) for every
actor_i in the phrase

#set a pair of current actor and best action
cur_actor = null
best_action = null
Loop until no word in the phrase
  for each word w in the phrase:
     if w is actor with no action assigned :
       #check if there is already a pair found
       if cur_actor and best_action is not null
         action(cur_actor) = best_action
         clear values of cur_actor and best_action
       # then start with a new current actor
     cur_actor = w
     # if w is of type verb, set value of best_action
     else if  w is a {ST/TV/AV} verb
       if priority(w) > priority(best_action)
       best_action = w
End loop
action(cur_actor) = best_action
clear values of cur_actor and best_action
```

Figure 7 Actor's action assignment algorithm

After actors and their actions are determined in a story, the last step is to assign speaker for each quote text.

Figure 8 shows the speaker assignment algorithm. First, we try to select a speaker from actors (or candidate speakers) in the same paragraph with the quote. The candidates are spilt into a list of candidates found in previous phrases (ListActorBefore) and a list of candidates found in the phrase following the quote (ListActorAfter). Candidates in both lists are sorted by priorities of their verb classes and their proximity to the quote. The priority of verb classes is “SV”, “TV”, “AV” and “Other”, respectively. The highest priority actors from both lists are compared and chosen. However, if the first actors from both lists have the same priorities, the first actor from ListActorAfter is chosen.

In case that there is no candidate speaker found in the same paragraph, the algorithm uses actors from a previous paragraph to find a speaker. Except that if the previous paragraph is also a quote, the algorithm assign a second last speaker found to a current quote. If everything else fails, the last speaker is assigned to the quote.

```
Speaker Assignment Algorithm
Input: tagged text structure and candidate speakers
Output: association of (quote_i, speaker_i) for each quote_i

# set of (quote, speaker) pairs in a story
Q = {(q_1, s_1), (q_2, s_2) …, (q_n, s_n)}

Loop for each quote q_i
 ListActorBefore = list of actors from previous phrase
 ListActorAfter = list of actors from following phrase
 if the paragraph with quote has a candidate(s)
   sort ListActorBefore and ListActorAfter
   s_i = the highest priority actor from both lists
 else #no candidate in the same paragraph
   #get actors from a previous paragraph
   ListActorBefore = previous paragraph list of actors
   sort ListActorBefore
   s_i = the highest priority actor in ListActorBefore
 endif
 if s_i is empty and a previous phrase is a quote
   then #assign second last speaker as a speaker
    s_i = s_i-2 when s_i-2 not same as s_i-1
   else #everything else fails
      # assign last speaker as a speaker
      s_i = s_i-1
 endif
End loop
```

Figure 8 Speaker assignment algorithm

4 Experiments

4.1 Experiment Design

Experiments are designed to evaluate efficiency and correctness of the proposed algorithms. A collection composes of forty short children stories collected from story books. A collection has 1086 paragraphs, 825 quote phrases, with 3078 narrative phrases and 1020 adjacent phrases. There are 24839 word tokens.

In order to evaluation the effects of each actor identification tasks toward to the correctness of a speaker assignment task, the measurement are done in three stages. In stage 1, only an actor extraction module is applied to the collection, and then the actor list resulting from this step is used as candidate speakers directly applied to a speaker assignment task. In stage 2, after an actor extraction task, an actor list is compiled for each story. The list is used to match against adjacent quote phrases to find more candidate speakers, before applying it to a speaker assignment task. Stage 3 is similar to stage 2, except that an extracted actor list is used to scan against the whole story opposed to scan against only adjacent quote phrases.

4.2 Results and Discussion

Table 2 Results of the three actor annotation tasks and their effects to a speaker assignment task

	Actor Annotation Task		Speaker Assignment Task	
			Correct	(%)
Stage 1	Number of Actors	831	683	82.79
	Retrieved	586		
	Correct	546		
	Precision (%)	93.17		
	Recall (%)	65.70		
Stage 2	Number of Actors	831	703	85.21
	Retrieved	858		
	Correct	807		
	Precision (%)	94.06		
	Recall (%)	97.11		
Stage 3	Number of Actors	1512	706	85.58
	Retrieved	1569		
	Correct	1469		
	Precision (%)	93.63		
	Recall (%)	97.16		

Table 2 shows the results of the three stages. Out of 831 actors, stage 1, focusing on actors with the three verb classes, extracts correctly 546 actors and incorrectly 40 actors, resulting in assigning speaker correctly 683 quotes out of 825 quote or about 82.79%. Note that the recall rate, 65.70%, is not very high at this stage.

When the algorithm is improved with an actor annotation task in stage 2, it drastically increases the correctness of actor identification. From 831 actors found in phrases adjacent to quote, the algorithm identified correctly 807 actors or 97.11%, resulting in identify speakers correctly for 703 quotes or about 85.21%. Note that the number of retrieved actors is more than the number of actors because the algorithm also extracts noun phrases which are not actors.

Stage 3 follows the same pattern, from 1512 actors in the whole text of forty stories, 1469 actors are correctly identified and resulting in correctly assign speakers of 706 quotes or about 85.58%.

Note that although the algorithm in stage 3 improves the overall result, its effect is not as much as the improvement from stage 1 to stage 2.

In addition, the precision rates of three actor annotation tasks only remain about the same, from 93.17, 94.06, to 93.63%, although the recall rates improve from 65.70% in stage 1 to 97.11% in stage 2 and 97.16% in stage 3. This shows that increasing the number of actors is not always resulting in increasing the number of correct speaker assignment. Only *true* candidate actor detection could result in improving a chance to find a *true* speaker of the quote.

5 Conclusion and Future Work

This work proposed a methodology to annotate speakers in multiple Thai children stories. The framework works well without avatar list with an average of 85% correctness.

However, future work is still needed to resolve anaphora. In addition, problems of same character with different aliases and different characters with the same title must also be resolved.

Acknowledgment

This research is partially supported by Faculty of Information Technology Research Fund, King Mongkut Institute of Technology Ladkrabang.

References

[1] Netisopakul, P., Woraratpanya, K., Wangsiripitak, S., & Pasupa, K. (2012). *Emotional Speech Synthesis for Visibility Impaired*: *From-Book-to-Speech,* Research Project Funding Application submitted to National Research Council of Thailand.

[2] Zhang, Y, J., Black, W, A., & Sproat, R. (2003). Identifying Speaker in Children's Stories for Speech Synthesis. In *proceedings of EUROSPEECH 2003* (pp. 2041–2044). Geneva, Switzerland.

[3] Glass, K., & Bangay, S. (2006). Hierarchical Rule Generalisation for Speaker Identification in Fiction Books. In *proceedings of SAICSIT'06* (pp. 31–40). South African: South African Institute for Computer Scientists and Information Technologists.

[4] Glass, K., & Bangay, S. (2007). A Naïve, Salience-Based Method for Speaker Identification in Fiction Books. In *proceedings of the 18th International Symposium of the Pattern Recognition Association of South Africa* (pp. 1–6). Pietermaritzburg, South Africa: PRASA.

[5] Tachanaparak, N., Kunlayanakual, S., & Niinsripaiwan, A. (2011). *Semi-automatic Novel Text Classification based on Character* (BEST Working paper 13p33c002). National Electronics and Computer Technology Center. (in Thai) Retrieved August 08, 2013, from http://thailang.nectec.or.th/halloffame/images/stories/best/download/13p33c002.pdf

[6] Tungkualtaveesub, W., Ratmanee, P., & Jindaphitak, T. (2010). *Thai Word Segmentation a Hybrid Approach* (BEST Working paper 34S001). National Electronics and Computer Technology Center. (in Thai) Retrieved August 08, 2013, from http://thailang.nectec.or.th/halloffame/images/stories/best/download/best2010_12p34s001.pdf

[7] Satayamas, V. (2012). *Part-of-speech tagger for Thai Language*. Retrieved August 08, 2013, from http://veer66.wordpress.com/tag/pos/

[8] NAiST Lab, Kasetsart University. (2011). *Jitar model and Jitar 20100224*. Retrieved August 08, 2013, from http://naist.cpe.ku.ac.th/pkg/

[9] Soon, Meng, W., Ng, Tou, H., & Lim, Yong , Chung, D. (2001). A Machine Learning Approach to Coreference Resolution of Noun Phrases. *Computational Linguistics*, 27(4), 521–544.

[10] Kong, F., GuoDong, Z., & Zhu Qiaoming. (2009). Employing the Centering Theory in Pronoun Resolution from the Semantic Perspective. In *proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing* (pp. 987–996). Singapore: Association for Computational Linguistics Stroudsburg, PA, USA.

[11] Sutheebanjard, P., & Premchaiswadi, W. (2009). Thai Personal Named Entity Extraction without using Word Segmentation or POS Tagging. In *8th International Symposium on Natural Language Processing* (pp. 221–226). Bangkok.