Automatic Annotation Inconsistency Detection: an $n$-Gram-Based Approach

Prachya Boonkwan  Thepchai Supnithi
Human Language Technology Laboratory
National Electronics and Computer Technology Center
Pathumthani, Thailand 12120

Abstract
This paper presents a method to detect potential annotation inconsistency in monolingual corpora. It aims at assistance for linguists to verify corpus annotation and to incorporate corpora with different versions of part-of-speech tag sets, by automatically providing list of potential inconsistency. An $n$-gram probabilistic model was utilized to identify words that are surrounded with similar parts of speech but are annotated with different parts of speech. By tri-gram-based detection, Orchid-1, a Thai part-of-speech-tagged corpus, was examined for auto-inconsistency and for cross-inconsistency with referential Orchid-2. In total, 14,040 places were marked auto-inconsistent, while 20,144 places were marked cross-inconsistent regarding to Orchid-2. The preliminary results show that it is feasible to integrate Orchid-1 and Orchid-2.

1 Introduction
Corpus annotation is a labor-intensive, yet indispensable task in NLP. In recent decades, statistical methods have played an important role in many NLP technologies, such as morphological analysis, statistical parsing, statistical machine translation, etc. Large amount of annotated corpora are required as rudimentary training sets for statistical language learning.

To advance Thai language processing technologies, ORCHID (Open Linguistic Resource Channeled toward Inter-Disciplinary Research), a POS-tagged corpus, was initiated in 1996. Up until now, the first generation (Orchid-1) was released, while the second generation (Orchid-2) is under construction. Some annotation inconsistencies were found in Orchid-1 because of several incoherent annotation criteria. This contaminated the statistical language model and made the accuracy of SWATH, a Thai word-segmentation and POS-annotation tool, considerably deteriorate. Meanwhile, as the size of Orchid-2 grew, the POS tag set was revised. Some tags were re-defined while new ones were introduced. This necessitated the corpus verification on Orchid-1 for forward compatibility to the referential Orchid-2.

This paper presents a method to detect potential annotation inconsistency in monolingual corpora using $n$-gram probabilistic model. The system was developed on a database manage-
ment system for ease of data retrieval. Afterward, Orchid-1 was analyzed by this method to de-
tect both auto-inconsistencies and cross-inconsistencies regarding to Orchid-2.

The rest of the paper is organized as follows. §2 elaborates the problems of annotation in-
consistency in corpus construction. §3 describes the inconsistency detection method based on \( n \) -
gram probability. §4 clarifies the implementation of the method, as some SQL queries are
shown. §5 shows some preliminary results and discusses the feasibility to integrate Orchid-1 and
Orchid-2. Finally, §6 concludes the paper and explains the future works.

2 Annotation Inconsistency

Inconsistency of annotation is a major obstruction in corpus verification. Beyond the difficulty of
annotation process that requires linguistic expertise in selecting an appropriate annotation for
each word, annotation verification furthermore requires skillful recognition of annotation accord-
ing to proper surrounding context. Linguists memorize a great deal of annotation patterns as long
as they scrutinize throughout the text.

Detection of annotation inconsistency in human intuition can be depicted in Figure 1. Suppose
the annotation \( x \) is analyzed. Referential annotation patterns \( cdefg \) are recognized by
means of focused surrounding context \( cd_{fg} \). Therefore, \( x \) is marked inconsistent. It is quite
clear that this process can be automated by means of statistical language learning.

The inconsistency is classified into two categories: auto-inconsistency and cross-
inconsistency. An annotation is said to be **auto-inconsistent** if the referential annotation patterns
from the same corpus are used. The auto-inconsistency is used to examine the self-coherency of
a corpus. An annotation is said to be **cross-inconsistent** if the referential annotation patterns from
other corpora are used in place. The cross-inconsistency is used to examine the annotation coher-
ency of a corpus with respect to other corpora. For example, the older Orchid-1 was checked for
cross-inconsistency with the newer Orchid-2 in order that the annotation criteria of Orchid-1
would be upgraded to conform that of Orchid-2.

The next section describes a method to automatically detect annotation inconsistency using \( n \) -gram probabilistic model.
3 Inconsistency Detection

Surrounding context plays a central role in detecting annotation inconsistency. We assumed that words with similar context should be similarly annotated.

To declare the assumption, let us formalize the inconsistency detection task as follows. An annotation is potentially inconsistent if it is not the most possible annotation. For any words \( w_k \) at the \( k \)th position of a given sentence of length \( m \), a class \( c_k \) is said to be the most possible annotation if it maximizes the probability \( P(w_k \mid c_k) \) in Eq (1).

\[
P(w_k \mid c_k) = P(w_k c_1 \ldots c_k \ldots c_m)
\]

The probability in Eq (1) is calculated by counting the cooccurrences of the word \( w_k \) and all classes of the sentence \( c_1, c_2, c_3, \ldots, c_m \). The data sparseness was prevented by smoothing \( P(w_k \mid c_k) \) with an \( n \)-gram probabilistic model. Therefore, Eq (1) is approximated by \( \hat{P}_n(w_k \mid c_k) \) in Eq (2).

\[
\hat{P}_n(w_k \mid c_k) = P(w_k c_{k-n} \ldots c_k \ldots c_{k+n})
\]

The probability in Eq (2) is calculated by counting the cooccurrences of the word \( w_k \), the preceding \( c_{k-n}, \ldots, c_{k-1} \), the class \( c_k \), and the succeeding \( c_{k+1}, \ldots, c_{k+n} \). From Eq (2), the unigram, bigram, and trigram probabilities can be obtained, as shown in Eq (3), (4), and (5), respectively.

\[
\hat{P}_1(w_k \mid c_k) = P(w_k, c_{k-1}, c_k, c_{k+1})
\]

\[
\hat{P}_2(w_k \mid c_k) = P(w_k, c_{k-2}, c_{k-1}, c_k, c_{k+1}, c_{k+2})
\]

\[
\hat{P}_3(w_k \mid c_k) = P(w_k, c_{k-3}, c_{k-2}, c_{k-1}, c_k, c_{k+1}, c_{k+2}, c_{k+3})
\]

Eq (3), (4), and (5) were made use throughout this paper. That is, any words that occupy in similar unigram, bigram, and trigram contexts but do not maximize their corresponding probabilities were detected as potential inconsistencies.

The next section describes the implementation of this model as SQL queries on a relational database management system.

4 Implementation

The inconsistency detection system was implemented on a database management system (DBMS) for ease of data retrieval. Although Microsoft Access 2003 was chosen in this paper, all queries were developed in Standard SQL for universal compatibility with other DBMS’es such as MySQL and OpenOffice’s Base. However, one should choose a DBMS that supports view capability because of query reusability.

4.1 Data Preparation

Corpus importation is the first step of data preparation. Orchid Corpora, namely Orchid-1 and Orchid-2, were imported as individual tables in the database. The tables storing the corpus contents are schematized in Table 1.
Table 1: Schema of corpus tables

<table>
<thead>
<tr>
<th>Fields</th>
<th>Types</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>sent_id</td>
<td>Number</td>
<td>Sentence ID</td>
</tr>
<tr>
<td>word_id</td>
<td>Number</td>
<td>Word ID</td>
</tr>
<tr>
<td>surface</td>
<td>Text</td>
<td>Surface form of word</td>
</tr>
<tr>
<td>pos</td>
<td>Text</td>
<td>Annotated POS</td>
</tr>
</tbody>
</table>

The table consists of four attributes: sent_id, word_id, surface, and pos. Sentence ID and word ID are in fact of type INT in the Standard SQL. Surface and POS are of type VARCHAR, whereas the length can be arbitrarily selected. Suppose that the 365th sentence of a corpus is “MyPR$ walletNN wasVBE stolenVBN.” It will be stored in the table as in Table 2.

Table 2: Example of fragment of corpus table

<table>
<thead>
<tr>
<th>sent_id</th>
<th>word_id</th>
<th>surface</th>
<th>pos</th>
</tr>
</thead>
<tbody>
<tr>
<td>365</td>
<td>1</td>
<td>my PR$</td>
<td></td>
</tr>
<tr>
<td>365</td>
<td>2</td>
<td>wallet</td>
<td>NN</td>
</tr>
<tr>
<td>365</td>
<td>3</td>
<td>was</td>
<td>VBE</td>
</tr>
<tr>
<td>365</td>
<td>4</td>
<td>stolen</td>
<td>VBN</td>
</tr>
</tbody>
</table>

4.2 \(n\)-Gram Probabilistic Models

To retrieve \(n\)-gram probabilities, sliding windows were created by shifting word IDs of each sentence. For example, a bigram sliding window, as shown in Figure 2, can be created by integrating sentence views, shifted word IDs by 2 and 1 to the right and by 1 and 2 to the left.

![Figure 2: Creating bigram sliding windows by shifting word IDs](image-url)

Each \(n\)-word shifted view was constructed by the SQL expression in Figure 3.
SELECT sent_id, word_id - n AS word_id_next_n, surface, pos
FROM CorpusTable

Figure 3: SQL expression for creating word-shifted view of sentence

If right-shifted, the value of $n$ becomes positive. If left-shifted, $n$ becomes negative and the attribute $\text{word_id}_{\text{next}}n$ becomes $\text{word_id}_{\text{prev}}n$. Each word-shifted view is named after its corpus, its direction, and the value $n$. For example, 2-word left-shifted and 3-word right-shifted views of Orchid-1 are named as $\text{Orchid1}_\text{Prev}2$ and $\text{Orchid1}_\text{Next}3$.

A sliding window, an integration of word-shifted views of different sizes, yielded us an $n$-gram statistics. The operator LEFT JOIN was used to associate the corpus table and created word-shifted views. By this operator, if one of surrounding context is not available, NULL will be generated resembling sentence boundary marker. Then all retrieved entries are grouped by surface form, its POS, and their surrounding context and counted the cooccurrence frequency. For example, bigram statistics of Orchid-1 can be retrieved by the SQL expression in Figure 4. The Orchid-1 table is left-joined to 2-word left-shifted, 1-word left-shifted, 1-word right-shifted, and 2-word right-shifted views. The resulted attributes are 2 previous POS’es, surface form, POS, 2 succeeding POS’es, and frequency.

SELECT Orchid1_Prev2.pos AS prev2_pos,
Orchid1_Prev1.pos AS prev1_pos,
Orchid1.surface, Orchid1.pos,
Orchid1_Next1.pos AS next1_pos,
Orchid1_Next2.pos AS next2_pos,
COUNT(*) AS freq
FROM (( ( Orchid1
            LEFT JOIN Orchid1_Prev2
                ON Orchid1.sent_id = Orchid1_Prev2.sent_id
                AND Orchid1.word_id = Orchid1_Prev2.word_id_prev2
            )
            LEFT JOIN Orchid1_Prev1
                ON Orchid1.sent_id = Orchid1_Prev1.sent_id
                AND Orchid1.word_id = Orchid1_Prev1.word_id_prev1
            )
            LEFT JOIN Orchid1_Next1
                ON Orchid1.sent_id = Orchid1_Next1.sent_id
                AND Orchid1.word_id = Orchid1_Next1.word_id_next1
            )
            LEFT JOIN Orchid1_Next2
                ON Orchid1.sent_id = Orchid1_Next2.sent_id
                AND Orchid1.word_id = Orchid1_Next2.word_id_next2
GROUP BY Orchid1_Prev2.pos, Orchid1_Prev1.pos,
Orchid1.surface, Orchid1.pos,
Orchid1_Next1.pos, Orchid1_Next2.pos

Figure 4: SQL expression for creating a bigram statistics of Orchid-1
($\text{Orchid1}_\text{Stat}_\text{Bigram}$)

To identify the most possible annotation by Eq (3), (4), and (5), cooccurrences of surface forms and surrounding POS’es with maximum frequencies were retrieved. For example, the maximum frequencies of bigram statistics were retrieved by the SQL expression in Figure 5.
SELECT prev2_pos, prev1_pos,  
surface,  
next1_pos, next2_pos,  
MAX(Orchid1_Stat_Bigram.freq) AS freq  
FROM Orchid1_Stat_Bigram  
GROUP BY prev2_pos, prev1_pos, surface, next1_pos, next2_pos

Figure 5: SQL expression for retrieving maximum frequencies of bigram statistics  
(Orchid1_MaxStat_Bigram)

Possible annotations can be now retrieved by simply joining n-gram statistics with the maximum frequencies. For example, the possible annotations of Orchid-1 regarding to bigram statistics can be retrieved by the SQL expression in Figure 6.

SELECT Orchid1_Stat_Bigram.pos_prev2,  
Orchid1_Stat_Bigram.pos_prev1,  
Orchid1_Stat_Bigram.surface,  
Orchid1_Stat_Bigram.pos_next1,  
Orchid1_Stat_Bigram.pos_next2,  
Orchid1_Stat_Bigram.freq  
FROM Orchid1_Stat_Bigram  
LEFT JOIN Orchid1_MaxStat_Bigram  
ON Orchid1_Stat_Bigram.pos_prev2  
= Orchid1_MaxStat_Bigram.pos_prev2  
AND Orchid1_Stat_Bigram.pos_prev1  
= Orchid1_MaxStat_Bigram.pos_prev1  
AND Orchid1_Stat_Bigram.surface  
= Orchid1_MaxStat_Bigram.surface  
AND Orchid1_Stat_Bigram.pos_next1  
= Orchid1_MaxStat_Bigram.pos_next1  
AND Orchid1_Stat_Bigram.pos_next2  
= Orchid1_MaxStat_Bigram.pos_next2  
AND Orchid1_Stat_Bigram.freq  
= Orchid1_MaxStat_Bigram.freq

Figure 6: SQL expression for retrieving most possible annotations  
(Orchid1_PossibleAnnotation_Bigram)

4.3 Inconsistency Detection

The n-gram possible annotations were checked with sliding windows of corresponding size n. For example, the bigram sliding windows can be retrieved by the SQL expression in Figure 7.
SELECT Orchid1.sent_id, Orchid1.word_id,
    Orchid1_Prev2.pos AS prev2_pos,
    Orchid1_Prev1.pos AS prev1_pos,
    Orchid1.surface, Orchid1.pos,
    Orchid1_Next1.pos AS next1_pos,
    Orchid1_Next2.pos AS next2_pos,
FROM ( ( ( Orchid1
        LEFT JOIN Orchid1_Prev2
            ON Orchid1.sent_id = Orchid1_Prev2.sent_id
            AND Orchid1.word_id = Orchid1_Prev2.word_id_prev2
        )
        LEFT JOIN Orchid1_Prev1
            ON Orchid1.sent_id = Orchid1_Prev1.sent_id
            AND Orchid1.word_id = Orchid1_Prev1.word_id_prev1
    )
    LEFT JOIN Orchid1_Next1
        ON Orchid1.sent_id = Orchid1_Next1.sent_id
        AND Orchid1.word_id = Orchid1_Next1.word_id_next1
)
LEFT JOIN Orchid1_Next2
    ON Orchid1.sent_id = Orchid1_Next2.sent_id
    AND Orchid1.word_id = Orchid1_Next2.word_id_next2

Figure 7: SQL expression for creating a bigram sliding window
(Orchid1_Context_Bigram)

Finally, the annotation inconsistencies can be detected by joining sliding windows and
n-gram possible annotations. Surface forms, and preceding and succeeding contexts of the for-
mer are joined with those of the latter, while incompatible POS’es are listed. Auto-inconsistency
can be detected by comparing the sliding windows and the n-gram possible annotations of the
same corpus. For example, bigram auto-inconsistencies in Orchid-1 were detected by the SQL
expression in Figure 8. Cross-inconsistency can be detected in much the same way; the sliding
windows of the targeted corpus are compared with the n-gram possible annotation of the refer-
etorial corpus instead. For example, bigram cross-inconsistencies in Orchid-1 regarding to Or-
chid-2, named as Orchid1_CrossInconsistencyOrchid2_Bigram, can be done by com-
paring Orchid1_Context_Bigram with Orchid1_PossibleAnnotation_Bigram.
SELECT Orchid1_Context_Bigram.sent_id, Orchid1_Context_Bigram.word_id, Orchid1_Context_Bigram.surface, Orchid1_Context_Bigram.pos AS annotated_pos, Orchid1_PossibleAnnotation_Bigram.pos AS possible_pos
FROM Orchid1_Context_Bigram
LEFT JOIN Orchid1_PossibleAnnotation_Bigram
ON Orchid1_Context_Bigram.pos_prev2 = Orchid1_PossibleAnnotation_Bigram.pos_prev2
AND Orchid1_Context_Bigram.pos_prev1 = Orchid1_PossibleAnnotation_Bigram.pos_prev1
AND Orchid1_Context_Bigram.surface = Orchid1_PossibleAnnotation_Bigram.surface
AND Orchid1_Context_Bigram.pos_next1 = Orchid1_PossibleAnnotation_Bigram.pos_next1
AND Orchid1_Context_Bigram.pos_next2 = Orchid1_PossibleAnnotation_Bigram.pos_next2
WHERE Orchid1_Context_Bigram.pos <> Orchid1_PossibleAnnotation_Bigram.pos

Figure 8: SQL expression for detecting auto-inconsistencies in Orchid-1 with bigram statistics (Orchid1_AutoInconsistency_Bigram)

All annotation inconsistencies were listed out by Orchid1_AutoInconsistency_Trigram, Orchid1_AutoInconsistency_Bigram, Orchid1_AutoInconsistency_Unigram, Orchid1_CrossInconsistencyOrchid2_Trigram, Orchid1_CrossInconsistencyOrchid2_Bigram, and Orchid1_CrossInconsistencyOrchid2_Unigram. Words at the specified positions of Orchid-1 were marked inconsistent. At this step, linguists verified the inconsistencies by either accepting the suggested annotation or rejecting it. Linguist’s task on verification of corpus annotation was dramatically reduced by this method.

The next section shows some preliminary experiment results and discusses the feasibility to integrate Orchid-1 and Orchid-2.

5 Preliminary Results and Discussion

This method was applied to detect annotation inconsistency in Orchid-1. Auto-inconsistencies and cross-inconsistencies in Orchid-1 were counted by the expression SELECT COUNT(*) in Standard SQL. Union denotes integration of different sizes of \( n \). However, if different \( n \)-gram statistics detect the same place, the largest size is chosen. Preliminary results are shown in Table 3. Percentages are calculated by division by the total number of words in Orchid-1 (342,636 words).
Table 3: Potential annotation inconsistencies detected in Orchid-1

<table>
<thead>
<tr>
<th>Inconsistency</th>
<th>n-Gram Statistics</th>
<th>Occurrences</th>
<th>Percentages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-inconsistency</td>
<td>Trigram</td>
<td>1,121</td>
<td>0.33%</td>
</tr>
<tr>
<td></td>
<td>Bigram</td>
<td>3,410</td>
<td>1.00%</td>
</tr>
<tr>
<td></td>
<td>Unigram</td>
<td>11,362</td>
<td>3.32%</td>
</tr>
<tr>
<td></td>
<td>Union</td>
<td>14,040</td>
<td>4.10%</td>
</tr>
<tr>
<td>Cross-inconsistency</td>
<td>Trigram</td>
<td>445</td>
<td>0.13%</td>
</tr>
<tr>
<td></td>
<td>Bigram</td>
<td>4,252</td>
<td>1.24%</td>
</tr>
<tr>
<td></td>
<td>Unigram</td>
<td>17,449</td>
<td>5.09%</td>
</tr>
<tr>
<td></td>
<td>Union</td>
<td>20,144</td>
<td>5.88%</td>
</tr>
</tbody>
</table>

Auto-inconsistencies and cross-inconsistencies occur only 4.10% and 5.88% in Orchid-1, respectively.

The following instructions were suggested in order to integrate Orchid-1 and Orchid-2. Cross-inconsistencies should be revised as the first priority. Then, apply this method to recalculate the auto-inconsistencies. After auto-inconsistencies are revised, Orchid-1 and Orchid-2 are available for integration.

The next section concludes the paper and describes remained future works.

6 Conclusion

This paper presented a method for detecting annotation inconsistency in monolingual corpora. n-gram probabilistic models were applied to recognize two categories of potential inconsistency: auto-inconsistency and cross-inconsistency, namely. It has been shown that integration of Orchid-1 and Orchid-2 is feasible by this method.

Our future works remain. First, recall and precision of this method will be measured. Second, graphic user interface for Orchid-1 revision will be developed. And finally, other types of annotation inconsistencies will be analyzed.