Quoted Speech Extraction from Indonesian News

Franky
Shanghai Jiao Tong University
800 Dongchuan Road,
Shanghai, 200240
franky.id@gmail.com

ABSTRACT

In this work, we presented an early effort to extract quoted speech from Indonesian news, together with the speaker of the quote and the topic in which the quote was produced. We crawled the specific news website *Kompas.com* to get a collection of news that had been already manually grouped by topic. The quotes were extracted using a simple pattern matching algorithm to detect the double quotes punctuations that is usually used to surround the quoted speech. The names of the speaker were extracted using a list of indicator words, e.g '*kata* (*said*)'. Names that were extracted includes names that were explicitly mentioned e.g. '*kata Bob* (said *Bob*)' and also names that were referred by reference words e.g. '*katanya* ("*said by him/her"*)'. Different names in different news for the same speaker (person) were aggregated by directly matching similar names across the news, performing a simple edit distance, and by taking the longest name representing the speaker. The evaluations showed promising results to further improving and perfecting the method proposed by incorporating a more comprehensive list of indicator words, implementing recursive algorithm to resolve the reference (anaphora), using proper name entity recognition for Indonesian language, and keeping the list of name variants of a person.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing — *Linguistic processing*;
I.2.7 [**Artificial Intelligence**]: Natural Language Processing — *Text analysis*

General Terms

Algorithms, Design, Experimentation, Languages

Keywords

Crawling, tokenization, quoted speech extraction, opinion mining, name entity recognition, name variant matching, Indonesian language

1. INTRODUCTION

Opinions or comments from people for a certain topic are captured in news. These opinions are valuable for people who are interested in knowing the overall opinions about particular matter, e.g. opinions of voters in presidential campaign, or to know the comments of a certain well known person to related topic. These opinions are captured in news in the forms of direct and indirect speeches. In this work, we tried to extract these valuable information that were in the form of quoted speech (direct speech). The reason is that the quoted speech is more likely to contain the actual words produced by the speaker, and the extracted quoted may be represented as it is, without any modification to the structure of the sentences.

We defined the problems that we tried to solve in this work into three main problems. The first problem is to extract people's opinions from news that are in the form of quoted speech i.e. surrounded by double quotation marks. For the purpose of current work, we only extracted quotes that have the speaker's mentioned after the quotes. For instance, we only extracted quote with form:

> *"Kejadian ini preseden bagi hubungan diplomasi kita," kata Mega*
>
> "*This incident set a bad precedent for our diplomatic relations," said Mega*

and not in the form of:

> *Mega mengatakan, "Kejadian ini preseden bagi hubungan diplomasi kita."*
>
> *Mega said,* "*This incident set a bad precedent for our diplomatic relation."*

where the name of speaker is located before the quote. We used this approach to simplify the problem and also our early analysis on the news showed that almost all of the quoted speeches we encountered are in the first form.

The second problem that we wanted to solve is to identify the correct speaker of the extracted quote, by representing the name in its complete or longest form, e.g. '*Mantan Presiden Megawati Sukarnoputri* (*Former President Megawati Sukarnoputri*)', instead of only '*Mega*'. This also included aggregating different names for the same person across different news under the same name. We took the longest name under assumption that the longest name is the best name representing the person, since it contains the title of the person. This was also due to limitation that we only did simple matching and distance algorithms to aggregate the different names and did not store the variants of names of a person. [4]

The third problem is related to anaphora (pronoun) resolution to extract the speaker's name that is referred by word '*nya*', e.g. '*ujarnya* ("*said by him/her"*)'. There are some words in Indonesian that can also be used, e.g. '*dia*', '*ia*' to refer to a person. But, for simplicity, we would like to be able to solve this problem first for word '*nya*'.

Our original goal in performing this work was to build a quote search engine that given a certain name of a person will return all comments of the person together with the topic. The application should also be able to return all comments under certain topic. We argue that the topic is important for a quote, since without a topic we will not know why a person produced such comments. However, in this work we mainly focused on the algorithm that can extract quotes and names of the speakers. As for the search application, currently, we only built a simple command line tool that given a name can return the quotes for that person together with the topics.

2. RELATED WORK

The similar work on quote extraction was in [4] and [1]. The works were performed in multilingual environment and on a large scale system. The work was part of the project to build a multilingual news explorer, which can be accessed through http://press.jrc.it/NewsExplorer/. It contains news collected from many news websites from different countries.

Other work that dealt with quote extraction was the extraction of Portuguese news from several news sources in [5]. In this work, the authors also automatically classified the news by topic using *Rocchio Classification* and *Support Vector Machine*. They also built a nice website that can be visited to show the resulting quotes in http://voxx.sapo.pt/.

For anaphora resolution, we found the work in this problem in [3]. The work utilized probability and statistical method to resolve the pronoun.

We differentiated our work by performing the quote extraction specifically to news in Indonesian language only, that already grouped by topic. We believed that the challenges will be different with the works presented above. In our work, we used simple algorithms to do this task and also attempt to perform an anaphora resolution that based on indicator words, that as far as we know, was not performed in the quote extraction works mentioned above.

3. ALGORITHM

3.1 Crawling the News

We crawled *Kompas.com* to get the news we needed. In this website, we can find some news that are already grouped under certain topic that they called *topik pilihan* (*selected topic*). The address for each topic is in the format of http://lipsus.kompas.com/topikpilihanlist/<id-of-topic>, where <id-of-topic> is an integer. We crafted a crawler that given a topic id, returned all the news inside the topic and saved each news in the form of Figure 1. We stored all the news in plain text format and put the news that has the same topic under the same folder, to easily differentiating and analyzing individual news.

```
<DOC-ID>1000-7</DOC-ID>
<TITLE>Djohar Sosok Murah Senyum</TITLE>
<TIMESTAMP>Minggu, 10 Juli 2011 | 14:15
PM</TIMESTAMP>
<CONTENT>Djohar Arifin Husin terpilih
sebagai…</CONTENT>
```

Figure 1. Format of the news crawled from the website.

3.2 Tokenization and Sentence Construction

The quoted speech extraction algorithm worked by detecting the occurrence of quote and name in a sentence level of a news. We performed such a task by iterating through the tokens inside the sentence, instead of using the regular expression. For each news file, first we tokenized the <CONTENT> part into a series of tokens. The algorithm is presented in Figure 2. The `title` is a list of known salutation (title) for a person in Indonesian language, e.g. Dr., Bpk., Yth., S.Pd., etc.

```
strings[] = split content by space
for i = 0 to length of strings
    if strings[i] is title then
        add strings[i] to tokens
    else
        char[] = split strings[i] to char

        j = 0
        while char[j] is punctuation
            add char[j] to tokens
            j = j + 1

        k = length of char - 1
        while char[k] is punctuation
            add char[k] to tokens
            k = k - 1

        token t
        for m = j to k
            add char[m] to t

        add t to tokens
```

Figure 2. Pseudo code for tokenization

After the tokenization, we constructed a list of sentences for each news. The algorithm worked by detecting the occurrence of a sentence boundary. In this work, we used "." and "–" as sentence boundaries, based on the analysis that we performed on the crawled news. A sentence was simply constructed by collecting the tokens sequentially until a sentence boundary was met. The exception was for a sentence boundary that occurred inside the quote (single or double quote). In this case, we did not consider it as a sentence boundary.

3.3 Named Entity Extraction

The next step was to identify the named entity mentioned in the news. This can be in the form of name, location, institution, time, etc. We performed a simple extraction by extracting word that started with capital letter. Words that started with capital letter and appeared consecutively was considered as a single named entity.

We permitted word '*dan* (*and*)', '*selaku* (*as*)', and '*untuk* (*for*)' to be appeared within a named entity. It was used to detect named entity such as '*Menteri Pendidikan dan Kebudayaan Mohammad Nuh* (*Minister of Education and Culture*)'. The other things we permitted were the use of parenthesis, that appeared in the news with the form, such as '*Wakil Ketua Komisi Pemberantasan Korupsi (KPK) Bibit Samad Rianto* (*Vice Chairman of Corruption Eradication Commissions (KPK) Bibit Samad Rianto*)'.

In order to minimize the false identification of named entity, we omitted the name of the days and the named entity that appeared after word '*di* (*at/on*)' and '*ke* (*to*)', that usually followed by a location name.

3.4 Quoted Speech Extraction

The quoted speech extraction worked by simply processing each sentence from sentence construction process. If the sentence contained double quote and one of the indicator words after the quote, then we tried to parse the sentence to get the name of the speaker. If the name of the speaker successfully identified, we stored the quote together with the speaker name. Table 1 shows indicator words that we used in extracting the quote. All the words in Table 1 have the same meaning as word '*said*' in English.

Table 1. List of indicator words used in extracting quote

Indicator Words			
kata	tanda	lanjut	ujar
papar	pinta	tutur	imbuh
ucap	ungkap	pungkas	jelas
tegas	tambah		

3.5 Name Identification

The task of identifying the speaker of the quote can be divided into two different categories, explicit name identification and reference resolution (anaphora resolution).

3.5.1 Explicit Name Identification

For the name that was explicitly mentioned, such as the example mentioned in section 1, we merely detected the named entity that occurred after the indicator words (Table 1). So, for the example shown in section 1, we took the name '*Mega*' as the speaker of the quote.

3.5.2 Anaphora Resolution

We tried to resolve a reference to the speaker of the quote in the form '*nya* (*him/her*)' only, for example:

"*Untuk penerbangan tadi pagi ke Makassar berangkat sesuai jadwal,"katanya*

"*The flight this morning to Makassar will depart as schedule," said him*

As the first step, we collected the candidate names for the reference. We added the names identified from explicit name identification process (section 3.5.1) into the candidate names. We also extracted other candidate names, which were the first occurrence of named entity that occurred before or after the indicator words. The indicator words that we used were different from the words used in quote extraction. The complete lists are shown in Table 2.

In Table 2, '*Before Name*' means that the indicator word should appear before the candidate name, e.g. '*Dikatakan Megawati* (*Megawati said*)' and '*After Name*' means the opposite, the indicator word should appear after the name, e.g. '*Djoko mengatakan* (*Djoko said*)'. The candidate name must be in the same sentence as the indicator word.

Table 2. Indicator Words for Candidate Names Extraction

Indicator Words	
Before Name	After Name
menurut	mengatakan
dikatakan	menyatakan
dipaparkan	memaparkan
kata	mengemukakan
ditambahkan	menjawab
lanjut	melanjutkan
	menegaskan
	berpendapat

We also gave number to the sentences in the news. The first sentence in the news were given id 0, the next sentence was given id 1, and so on. This information was added to the candidate names that we had extracted, indicating in what number of sentence that a candidate name appeared.

In order to resolve the reference '*nya*', we simply took the closest candidate name that appeared before '*nya*'. For example, if the sentence number 5 contains the reference that we wanted to resolve, then we looked for the candidate name that appeared in sentence number 4. If no candidate name appeared in sentence number 4, we kept searching to sentence number 3, 2, 1 and so on. The process will fail if no candidate names appeared before the reference word.

3.5.3 Name Merging

Every time a name was identified by the process described in section 3.5.1 or 3.5.2, we tried to look for the *best* name to replace the name. This was necessary because most of the time, the name of the speaker were in its short form, e.g. '*Mega*' instead of her full name *'Megawati Sukarnoputri*'. We called it the name merging process and it was performed in two steps, first at news level and then at the news collection level.

At news level means that we searched for the name replacement inside the same news in which the name appeared. We took a name from named entities that we extracted in section 3.3 as a replacement name if the name contains the name that we wanted to replace. We took the longest name representation as replacement. For example, '*Mega*' can be replaced by *'Megawati Sukarnoputri*' or '*Mantan Presiden Megawati Sukarnoputri*'. We took the second one as replacement, since it contains '*Mega*' and it is longer than the first one. If we could not find any replacement then we used the extracted name as the speaker's name.

The name merging process continued at the news collection level. For every successful extracted quote in the current news, we tried to replace the speaker's name with the better name representation that already appeared in other news processed before. If the current name was found to better representing the speaker, we replaced the name in every quote of this speaker with this current name. In this name merging process, we implemented edit distance to detect the same name that has slightly different representation, e.g. '*Juru Bicara Komisi Pemberantasan Korupsi Djoko*' and *'Juru Bicara Komisi Pemberantasan Korupsi (KPK) Djoko*'. The algorithm is shown in Figure 3.

4. IMPLEMENTATION

We implemented all the algorithms in Java. For dataset, we crawled *Kompas.com* and took all the news under topic number 1000 to 1505. The number of topics that we successfully extracted was 477 topics with the total of 22,867 news files. We performed the quote extraction for all the news files and able to extract 40,784 quotes. The resulting extractions where saved in a single file with the format of *name###quote##topic* for each quote.

A simple command line program was built to read the file. The program took a name as input and looked for similar names in all quotes using the algorithm as in Figure 3. But, instead of returning one replacement name, it will return a list of names that contain or contained by the input name, and also names that have distance smaller than the specified threshold. The program will then return all the quotes for all the names in list, together with the information of topic in which the quote appeared.

```
if length of current-name < 2
    replacement-name = current-name
else
    for name in previously-extracted-names
        if name contains current-name or
        current-name contains name
           replacement-name = name
           exit for
        else
           distance = editdistance(name,
           current-name)

           if distance < threshold and
           distance < min-distance
              replacement-name = name
              min-distance = distance

if length of replacement-name < length of
current-name
    replacement-name = current-name

    for quote in quote-collections
       if speaker-name = replacement-name
            speaker-name = current-name
```

Figure 3. Pseudo code for name merging process at news collection level

5. EVALUATION

The evaluation was performed by selecting 8 topics randomly. We only took the first 10 news for each topic, resulting in 80 news files. Then, we performed the quote extraction on this 80 news. The results were then compared with the quotes that were manually extracted to evaluate the performance.

5.1 Quoted Speech Extraction Results

The evaluation of quoted speech extraction results is shown in Table 3. Explicit means the number of quotes with name of the speaker explicitly mentioned. Ref means the number of quotes using reference (pronoun), that included all type of reference words and not only '*nya*'.

The evaluation shows that the method to perform quote extraction using double quote detection and indicator words achieved high recall. From this result, we can also see that most of the indicator words or speaker names are put after the quote. As for the reference, we can conclude that the word '*nya*' are mostly used.

Table 3. Quoted Speech Extraction Results

Number of Quotes		Extracted Quotes		Recall (%)	
Explicit	**Ref**	**Explicit**	**Ref**	**Explicit**	**Ref**
129	88	119	71	92.25%	80.68%
217		190		87.56%	

We analyzed the result to get the causes of failed extraction. The first one was the speaker name that appeared before the quote, which we already expected. This was due to our method did not detect such kind of quote. The second one was the typing error, in which the writer put wrong number of double quotes. Because our algorithm detected the opening and enclosing double quote, if they could not find the enclosing double quote then the algorithm would not extract the sentence as quote. The third one was that the use of different kind of double quote. Currently, we detected only two types of double quote character. The last one was the use of unrecognized indicator words, such as '*urai*' and '*terang*'.

5.2 Name Identification Results

In evaluating the name identification results, first we evaluated for successful quote extraction, whether the name identification return the correct name, without looking at the quality of extracted name as shown in Table 4. The results showed that for explicit name extraction that the success rate is 89.91% which is quite high, while for the task of anaphora resolution only achieves 57.75%.

Table 4. Name Identification Results

Names		Extracted Names		Success Rate (%)	
Explicit	**Ref**	**Explicit**	**Ref**	**Explicit**	**Ref**
119	71	107	41	89.91%	57.75%
190		148		77.89%	

The reason for failure in explicit name identification was mainly due to incomplete name detection, for example '*Sekjen Kementerian Perdagangan* (*Secretary General of Minister of Trade*)' instead of '*Sekjen Kementerian Perdagangan Ardiansyah Parman* (*Secretary General of Minister of Trade Ardiansyah Parman*)'. This is due to our simple named entity extraction algorithm.

As for the anaphora resolution, the low success rate was caused by several reasons. The first one is the same with the explicit name identification problem, which is incomplete name detection. The second is the incomplete indicator words used that did not cover words, such as '*mengingatkan* (*remind*)' and '*mengungkapkan* (*utter*)'. The other problem that we identified is the use of time entity, such as '*Senin* (*Monday*)', as the speaker's name. This is due to our candidate names extraction that did not filter the type of the named entity.

We further evaluated the results in terms of the extracted name quality. The result is shown in Table 5. We defined the name as having a bad quality with several criteria. First, if the name contains unnecessary information, e.g. '*Djohar Arifin Husein (Kanan)* (*Djohar Arifin Husein (Right Side)*)' and the second one, if it is incomplete, e.g. '*TNGGP Agus Wahyudi*' instead of

‘*Kepala Balai besar TNGGP Agus Wahyudi*’. From the result, the quality of extracted names was quite high for both types of names, which is around 88%.

Despite the high quality rate, there are some other problems that we haven’t able to tackle with our current approach, which is the merging of the name of a person that is completely different, such as ‘*Foke*’ and ‘*Fauzi Bowo*’. And also, to present the name in the form that is desired by the user, such as ‘*Gubernur Fauzi Bowo* (*Governor Fauzi Bowo*)’, instead of ‘*Foke*’. Currently, we only took the longest name representation.

The other problem that was not captured in evaluation news collection is that the use of word ‘*dan* (*and*)’ in named entity extraction. It caused a problem of extracting two speakers like ‘*SBY dan Megawati* (*SBY and Megawati*)’, which is sometime not the true representation of the speaker’s name.

Table 5. Extracted Name Quality

Extracted Names		**Good Quality Names**		**Quality Rate (%)**	
Explicit	**Ref**	**Explicit**	**Ref**	**Explicit**	**Ref**
107	41	95	36	88.78%	87.80%
148		131		88.51%	

6. CONCLUSION

The evaluation showed that the method used in this work to perform quote extraction gave a high recall. Given a more comprehensive indicator words and adding the detection of quote with the name of speaker’s occurred before the quote, the result should be further improved.

As for the speaker’s name extraction, we believe that given proper named entity recognizer, that can differentiate name, location, time, etc, the accuracy will be better, as it will not assign location or time as the speaker. Another idea to improve the results is to keep a list of name variants for a person, which can be used to properly detect any name variations of a single person and also to present the speaker’s name in desired format. Currently, we do not have the resource and have not been able to found the method to do this task automatically.

In attempt to perform an anaphora resolution, we argue that the method proposed here, by using indicator words, is promising. Our idea to get a better result is to combine the use of named entity recognizer and recursive algorithm that keep searching the speaker’s name when it encounters another reference. The use of recursive algorithm is considered since sometimes the other reference word such as ‘*ia*’ is used in the sentence before the indicator word, e.g. ‘*Ia mengatakan…* (*He said*)’, instead of the true speaker’s name. We considered in implementing these ideas that we had in our future works.

7. REFERENCES

[1] Balahur, Alexandra, Ralf Steinberger, Erik van der Goot, Bruno Pouliquen, and Mijail Kabadjov. "Opinion Mining on Newspaper Quotations." *IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies.* 2009.

[2] *EMM NewsBrief.* n.d. http://press.jrc.it/ (accessed 12 26, 2011).

[3] Ge, Niyu Ge, John Hale, and Eugene Charniak. "A Statistical Approach to Anaphora Resolution." *In Proceedings of the Sixth Workshop on Very Large Corpora.* 1998.

[4] Pouliquen, Bruno, Ralf Steinberger, and Clive Best. "Automatic Detection of Quotations in Multilingual News." *In Proceedings of Recent Advances in Natural Language Processing 2007.* Borovets, Bulgaria, 2007.

[5] Sarmento, Luis, and Sergio Nunes. "Automatic Extraction of Quotes and Topics." *4th Doctoral Symposium on Informatics Engineering.* 2009.

[6] *VOXX.* n.d. http://voxx.sapo.pt/ (accessed 12 26, 2011).

.