Selective Annotation of Sentence Parts: Identification of Relevant Sub-sentential Units

Ge Xu[1,2] **, Xiaoyan Yang**[1,2] **and Chu-Ren Huang**[3]
[1]Department of Computer Science, Minjiang University, China
[2]Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, China
[3]Department of Chinese and Bilingual Studies,
The Hong Kong Polytechnic University, Hong Kong
xuge@pku.edu.cn, 349622662@qq.com, churenhuang@gmail.com

Abstract

Many NLP tasks involve sentence-level annotation yet the relevant information is not encoded at sentence level but at some relevant parts of the sentence. Such tasks include but are not limited to: sentiment expression annotation, product feature annotation, and template annotation for Q&A systems. However, annotation of the full corpus sentence by sentence is resource intensive. In this paper, we propose an approach that iteratively extracts frequent parts of sentences for annotating, and compresses the set of sentences after each round of annotation. Our approach can also be used in preparing training sentences for binary classification (domain-related vs. noise, subjectivity vs. objectivity, etc.), assuming that sentence-type annotation can be predicted by annotation of the most relevant sub-sentences. Two experiments are performed to test our proposal and evaluated in terms of time saved and agreement of annotation.

1 Introduction

High quality resources are essential to the performance of NLP systems and in recent information content related NLP tasks, such resources are typically constructed through annotation at sentence level. For instance, to implement opinion mining systems, sentiment/polarity/emotion expressions or lexicons are often built from a corpus of movie or product reviews; and question templates are extracted from sentence corpora for Q&A systems. The construction of these two types of resources are quite similar to construction of resources for other information content related NLP tasks, such as information quality tasks (detection of best/most help answers, hyperbole/embellishment, or lying), and speaker attitude/intention tasks (detection of metaphor/metonymy/irony/sarcasm), etc. They typically involve extraction of a set of specific expressions (words, word sequences, long-distance collocation etc.) from sentence corpora. They share the following characteristics:

- Manually annotating the corpus sentence by sentence is time-consuming, and often impractical;
- There is no prior knowledge of the number and location of expected expressions, hence the full corpus needs to be annotated;
- Information content properties must be determined at sentence (or higher) level, yet expressions marking such properties are often subsidiary units of a sentence;
- Many sentences have repeated or similar subsidiary units;

Based on the above observations, we propose in this paper a selective semi-automatic annotation schema, which aims to reduce resource requirements and improve annotation quality by utilizing human experience. Our selective annotation approach relies on a sequence mining algorithm to generate frequent patterns in sentences, and annotate patterns (as sub-sentences) instead of full sentences. Our proposed approach can find most important expressions statistically, and will annotate a pattern only once to avoid repeated annotation. Beside extracting specific expressions from the set of sentences, we can also use

This work is licensed under a Creative Commons Attribution 4.0 International License. License details: http://creativecommons.org/licenses/by/4.0/

Proceedings of the 12th Workshop on Asian Language Resources,
pages 86–94, Osaka, Japan, December 12 2016.

Input	lines=('caabc', 'abcb', 'cabc', 'abbca'), minSup=3
Output	('a'), 4;('a', 'b'), 4;('a', 'b', 'c'), 4;('a', 'c'), 4;('b',), 4;('b', 'c'), 4;('c',), 4;('c', 'a'), 3;('c', 'b'), 3

Table 1: Sequence mining

our approach to help construct a training corpus for a binary classifier, an related experiment in section 3 is described.

In the following sections, we will describe our approach, and two experiments are provided to show that our approach is effective in specific tasks.

2 Our Approach

2.1 Frequent Sequence Mining

Sequential pattern mining is a topic of data mining concerned with finding statistically relevant patterns between data examples where the values are delivered in a sequence (Pei et al., 2001; Yan, 2003). "Frequent" means that a sequence occurs in no less than minimum support (predefined minimum frequency) times. An example in table 1 shows the input and output of such algorithms.

In table 1, $('a', 'c'), 4$ means the sequence $('a', 'c')$ occurs 4 times in lines, and 4 is larger than minSup (minimum support), which is set to 3 in table 1. Furthermore, skip is allowed between elements of the sequence.

2.2 Algorithm

Algorithm 1 Our approach

Input:
UC(Unlabeled corpus)
MinSup(minimum support)
Initialize: YesSet,NoSet,DoubtSet,MinSupRatio
Output: YesSet,NoSet,DoubtSet
1.Breaking UC into sentences, SENS
while $True$ **do**
  2.Runing a sequence mining algorithm with MinSupRatio on SENS, the result is PATLIST.
  3.If PATLIST is empty, decrease MinSupRatio.
  4.If MinSupRatio is lower than given threshold, **break**; else goto step 2.
  5.Pruning PATLIST using YesSet, NoSet, DoubtSet.
  6.Annotating each pattern in PATLIST by the tag of Yes,No, or Doubt.
  7.Compressing SENS using YesSet, NoSet.
end while
8.return YesSet,NoSet,DoubtSet

Here are some explanations for the approach:

1. MinSup: minimum support, used for terminated sequence mining algorithm. If no patterns occurs MinSup times or more, we think manual annotation should be applied on sentences directly.

2. UC: Unlabeled corpus. For English, the corpus is naturally segmented by space; for other languages such as Chinese, word segmentation should be applied before the corpus is input.

3. MinSupRatio: It will decrease gradually to help sequence mining algorithms to find more less frequent patterns, until the $MinSupRatio * size(SENS)$ is lower than MinSup.

4. YesSet,NoSet,DoubtSet are empty initially.

2.3 Annotating Patterns

Ideally, we wish to meet only two types of patterns, namely YES patterns and NO patterns. A YES pattern is normally the thing we want to extract from a corpus. In different tasks, YES means a sentiment expression, a question template or a dish name etc. NO patterns are what we do not need.

However, the possible types of a pattern are more complicated, which are listed as follows:

1. YES means the pattern is a YES pattern.

2. YES+ means the pattern contains a YES pattern.

3. YES- means the pattern possibly is a part of a YES pattern.

4. NO means the pattern is not a YES pattern. None of its elements have relation with a YES pattern.

5. NO+ means the pattern is not a YES pattern, but some of its elements may have relation with a YES pattern.

To simplify annotation, we merge five types of patterns into three annotation options:

1. Yes(YES, YES+). Meaning that the pattern **contains** is a YES pattern or contains a YES pattern.[1].

2. No(NO). Meaning that the pattern **has nothing to do** with a YES pattern. Any word in the pattern will not be a part of a YES pattern.

3. Doubt(YES-,NO+). Meaning that the current pattern (or its part) maybe a part of a YES pattern, and want to see **longer** patterns containing the current pattern (or its part) in next rounds, then make a decision.

If a pattern is annotated, it will not be annotated again. For a No pattern, all its subsequence (skip allowed) are also No patterns; for a Yes pattern, all sequences containing it are also Yes patterns. All patterns that is annotated as "Doubt" will not be seen again by the annotator, but a longer pattern containing the "Doubt" pattern may be presented to the annotator in the later annotating.

2.4 Compressing the Set of Sentences

After finishing a round of annotation, we need to use the result of annotation to compress the set of sentences, making the set of sentences used for next round smaller. Three points should be noted.

2.4.1 Difference between YesSet and NoSet

When compressing sentences, the patterns in YesSet should be seen as a whole, which is not required for patterns in NoSet.

For example, the current YesSet is {cf} and NoSet is {ba,deh}. For the sentence "abcde"[2], after compressing by YesSet, we still have "abcdeh" since "cf" as a whole does not occur in "abcde". However, after compressing by NoSet the result is "c". The "ab" is removed by using "ba" at first, and then "de" is removed by using "deh".

Such processing is derived from how we define Yes and No for a pattern. A **Yes** pattern can not guarantee that its part is also a **Yes** pattern, but a No pattern can guarantee that its elements have no relation with YES patterns.

[1]Sometimes, some annotators would see YES+ patterns as more confident YES patterns. The annotation criteria is not discussed here, and we just annotate both cases Yes.

[2]One letter denotes one word.

2.4.2 Determine Valid Sentences

In compressing sentences, we must have a **predicate** to determine if a sentence is still valid for the next round of sequence mining. If a sentence is not valid after compressing, it will be discarded, thus reducing the set of sentences.

By default, if the sentence is empty after compressing, the sentence is invalid. However, in specific domains, we must define specific predicates for a valid sentence. For example, we required that length of a valid pattern should be in a specified range, or some words must occur in a valid pattern. It is also not difficult to figure out that if we are extracting relationship templates, such as a comparison between two products or part-whole relationship between two entities, a valid sentence should contain at least two members.

We leave the task-related **predicates** to be defined by users with a function prototype: *bool IsValid-Sentence(sentence)*, which can be called in compressing sentences.

2.5 The order of Annotation

After using a sequence mining algorithm to generate patterns, we have three choices on the order of annotation according to different requirements.

1. Annotating patterns in the descending order of their frequencies. This strategy guarantee will meet YES patterns before YES+ patterns, so the elements in the set of Yes patterns are of higher quality.

2. Annotating patterns in the descending order of the length of patterns. This strategy is suitable for mining long patterns. It is highly possible that we meet YES+ patterns before YES patterns, so many unrelated words are stored in YES+ patterns. However, in compressing sentences by a YES+ pattern, we require that the whole YES+ pattern occur, so these unrelated words still have a chance to be annotated as No patterns, and then remove the unrelated words in YES+ in later stage.

3. Annotating patterns in the descending order of $frequency * length$ of patterns. This strategy will compress the sentences most rapidly. As with the second strategy, it is possible that we meet YES+ patterns before.

2.6 Similarity measurement

When annotating a pattern (we call it the main pattern), to further facilitate annotating, we train a Word2Vec[3] to help finding similar patterns. Normally, most of similar patterns have the same annotating tag with the main pattern, so **one click or keystroke can annotate many patterns**. We use the default sentence similarity from the trained Word2Vec model. If the similarity between a pattern and the main pattern is larger than a given threshold, the pattern is displayed below the main pattern. In our experiments, we find that listing similar patterns when annotating the main pattern can greatly improve efficiency.

2.7 Instructions for Use

Our approach is suitable for two typical tasks:

- Extracting specific expressions from corpora. If a corpus is rich in such expression [4], our approach can help to extract such expressions efficiently. For example, given proper corpora, our approach can extract sentiment expressions, polarity shifters, dish names, or question templates etc.

- Building sentence-level training corpus for binary classification. The basic assumption is that a Yes pattern can infer that the sentence containing the pattern is also Yes. Therefore, we can annotate patterns instead of sentences, but still tag the sentences correctly.

We should pay attention when applying our approach in building the training corpus for binary classification:

[3]https://pypi.python.org/pypi/gensim

[4]We will not discuss how to build a corpus that is rich in a specific expression, it is highly task-related.

- In design our approach, we find that if more than two classes are considered, the annotation would become much more complicated for annotators. For classification tasks with more than two classes, we recommend to use a one-versus-rest schema or other annotation tools.

- If a Yes pattern can not infer that the sentence containing it is Yes, our approach will fail. For example, "happy" is a positive expression, but we can not infer the sentence containing "happy" is a positive sentence, because polarity shifters such as "not X", "if X"[5] may occur in the sentence, then revert or cancel the polarity.

3 Experiments

In our paper, we give two experiments for two types of tasks respectively. They are **Extracting Chinese Sentiment Expressions** and **Annotating Sentences for Binary Classification**. Because Chinese is our native language, both experiments use Chinese corpora to reduce annotation uncertainty.

3.1 Extracting Chinese Sentiment Expressions

When building a sentiment-related system, we have some general Chinese sentiment lexicons to choose, such as 1)NTU Sentiment Dictionary[6]; 2)Chinese affective lexicon ontology[7]. Furthermore, there are some work on automatically constructing task-oriented sentiment resources, such as (Choi and Cardie, 2009; Lu et al., 2011; Jijkoun et al., 2010; Cruz et al., 2011), which still needs human annotation to improve quality, and limits the coverage of the constructed resources due to the restriction of automation.

However, when constructing sentiment resources in a specified domain, the best choice should be to construct directly from the corpora, which can guarantees coverage and exploits human sentiment knowledge.

We report here how we extract Chinese sentiment expressions from a corpus. Normally, it is supposed that sentiment expressions are mainly adjectives, some of verbs (like, hate etc.) and nouns (fool, idiot etc.). However, for creating a practical system, we also must consider multi-words expressions. In Chinese, multi-words expressions may also be the words that are wrongly segmented. In this experiment, we show how to construct a list of sentiment expressions for a specific domain.

3.1.1 Experimental Setting

We set MinSupRatio=0.05 and MinSup=3. At first, we extract frequent patterns using MinSupRatio, then gradually decrease the MinSupRatio to find less frequent patterns in next rounds. When the support equals MinSup and no patterns are mined, we stop our algorithm.

3.1.2 Corpus

The domain corpus is a 88MB file about product reviews, and contain 1280000+ lines of reviews. We use the Jieba package[8] to perform word segmentation and POS tagging on the corpus.

Since we know that degree adverbs have a strong relationship with polarity expressions, and in Chinese sentiment expressions often occur after the degree adverbs, we used degree adverbs[9] to extract the word sequences after them, and we require that the length of a sequence is less than 4 or a sequence meets a punctuation. Each word sequence is stored in one line.

By using this method, we extract 76000+ word sequences from 1280000+ lines of reviews, the 76000+ word sequences is what we input into our approach, from which we want to find (frequent) sentiment expressions.

3.1.3 Experimental Results

The main statistics of annotating process is shown in table 2.

[5]The X in a polarity shifter denotes a sentiment expression.
[6]http://nlg18.csie.ntu.edu.tw
[7]http://ir.dlut.edu.cn/EmotionOntologyDownload.aspx
[8]github.com/fxsjy/jieba
[9]We have 17 ,10, 14 Chinese degree adverbs for adjectives, verbs, nouns respectively.

Round	MinSupRatio	SEN	PAT	ANN	time
1	0.1	76235	1	1	6
2	0.05	76232	4	5	15
3	0.025	76190	4	4	8
4	1.3×10^{-2}	76102	8	7	15
5	1.3×10^{-2}	75938	1	2	9
6	6.3×10^{-3}	75869	32	24	83
7	6.3×10^{-3}	74480	1	2	23
8	3.1×10^{-3}	74374	67	36	181
9	3.1×10^{-3}	70906	6	7	54
10	1.6×10^{-3}	70513	111	49	397
11	1.6×10^{-3}	65565	13	12	136
12	1.6×10^{-3}	65137	3	4	14
13	7.8×10^{-4}	65021	213	73	542
14	7.8×10^{-4}	58365	34	24	111
15	3.9×10^{-4}	57606	482	108	1128
16	3.9×10^{-4}	49742	123	49	361
17	3.9×10^{-4}	46957	53	29	247
18	2.0×10^{-4}	46377	847	138	2003
19	2.0×10^{-4}	38434	397	79	621
20	2.0×10^{-4}	35192	280	59	474
21	9.8×10^{-5}	32569	1526	176	3699
22	9.8×10^{-5}	25861	1182	149	2359
23		14395			

Table 2: Process of Annotating. SEN means the number of sentences; PAT means the number of patterns; ANN means the number of main patterns that are manually annotated in this round; time means how many seconds this round takes.

From table 2, we can see that the MinSupRatio is gradually decreasing for mining less frequent patterns. In 23rd round, there are 14395 sentences after iterative compressing, which mainly contain low frequent words (lower than 3) and words from "Doubt" patterns. Since the annotation already covers the frequent patterns, annotating manually the 14395 sentences becomes less urgent, and will be chosen according to practical requirement.

We totally annotated 1037 main patterns. Since at most 10 similar patterns are listed when a main pattern is displayed, we need to annotated at most 10370 patterns theoretically. However, by our experience, we only need to manually annotate 2$\sim$4 similar patterns of the 10 similar ones, since many similar patterns have the same annotation tag (Yes, No or Doubt) with the main pattern. Therefore, the number of patterns annotated manually is about 3000, which is acceptable to build a domain lexicon.

The total annotation time is 14632 seconds, which include chatting time with colleagues and some time for personal affairs. Roughly, the time used for annotating is about 12000 seconds. If a person annotates about 61000+ (76000-14395) word sequences manually, and we assume that a word sequence take 2 seconds , the total time is about 120000+ seconds, 10 times longer than our approach. So, in this experiment, our approach can **save 90% time** for extracting frequent sentiment expressions.

3.2 Annotating Sentences for Binary Classification

Let's describe the application scenario briefly.

People may want to build a Q&A robot for an online game, which take a user's question as input and return the most possible answer to the user. After analyzing the records of dialogues, we find that many questions (sentences) have no relationship with the game, such as greetings and dirty words etc., which make the Q&A robot less efficient. So we prefer to annotate a training corpus and then train a binary classifier to filter such unrelated questions.

Three annotator (X,S,W) are required to annotate sentences both **manually** and **by our approach**. We mainly check the agreement between different annotators, as well as between fully manual annotation and our approach.

Annotator	X	S	W
Manual (min.)	25	28	35
Our Approach (min.)	13	15	18
Sentences covered	879	854	867
Time saved (%)	≈40%	≈40%	≈40%

Table 3: statistics of annotating 1000 sentences

	S	S^*	X	X^*	W	W^*
S	1.0	0.99	0.956	0.965	0.971	0.972
S^*	0.99	1.0	0.956	0.965	0.967	0.968
X	0.956	0.956	1.0	0.989	0.951	0.97
X^*	0.965	0.965	0.989	1.0	0.96	0.981
W	0.971	0.967	0.951	0.96	1.0	0.969
W^*	0.972	0.968	0.97	0.981	0.969	1.0

Table 4: Agreement of annotation. X,S,W denotes annotators X,S,W annotating sentences manually; X^*,S^*,W^* denote annotators X,S,W annotating sentences by our approach

3.2.1 Data

The data are users' questions collected from the game dialogues. To control the time used for manually annotating, we randomly selected 1000 sentences from the database of the user's questions.

3.2.2 Annotators

Annotator X is the first author, with NLP background, who designed the experiment and is familiar with domain of the corpus.**Annotator S** is a programmer who is quite familiar with the corpus, and read a lot of related corpus before this annotating. **Annotator W** is a person with medical science background and never plays online games or uses Q&A systems for games, who is introduced about the corpus and given some instructions on how to annotate by the first author for several minutes.

3.2.3 Experimental Results

Some statistics of the annotating process are shown in table 3.

Using our approach, we can cover 850∼880 sentences by annotating frequent patterns, and the rest 120∼150 sentences mainly contain words of low-frequency and still need manual annotation.

Averagely, 40% time are saved for those sentences annotated by our approach. If there are more sentences, the time saved by our approach is supposed to increase.

During the annotating, the significant difference between W and other two annotators is that W took a long time to annotate short patterns which are often single Chinese characters due to wrong word segmentation. Annotator W tended to annotate more "Doubt" patterns because she lacked domain knowledge, which causes more long patterns in the later rounds annotated. From such observation, it can be seen that our approach has a higher requirement for domain knowledge, which is the cost for high speed.

3.2.4 Agreement Analysis

At first, for each annotator, the agreement between manual annotation and our approach is 0.99,0.989 and 0.969 for S, X, W respectively in table 4. As introduced in section 3.2.2, annotators S and X both have domain knowledge for the annotating, so when patterns are short, they can use the domain knowledge to help annotating, and make their annotation using our approach more consistent with the fully manual annotation. For annotator W, due to lack of domain knowledge, many short patterns are annotated as No.

(a) Manually Annotation

	S	X	W
S	1.0	0.956	0.971
X	0.956	1.0	0.951
W	0.971	0.951	1.0

(b) Our Approach

	S^*	X^*	W^*
S^*	1.0	0.965	0.968
X^*	0.965	1.0	0.981
W^*	0.968	0.981	1.0

Table 5: Agreement of annotation

In fact, such No patterns help to infer a sentence as Yes when the sentence is shown to annotator W as a whole.

Agreement between different annotators using manual annotation are shown in table 4(a), and agreement between different annotators using our approach are shown in table 4(b). We are happy to see that the average agreement using our approach is a bit higher than agreement using manual annotation, which may suggest that our approach can provide more consistent annotation among different annotators. Furthermore, the agreement between manual annotation and our approach is averagely higher than the agreement between different annotators, which also suggests that our approach will not harm the annotation quality while accelerating the annotation.

4 Conclusion and Future work

In this paper, to extract a set of specific expressions from corpora, we propose an approach that iteratively uses sequence mining algorithms to extract frequent parts of sentences for annotating. The approach can also be used in constructing training corpus for a binary classification under specific condition.

The approach has the following merits:

1. Our approach can greatly save human labor when extracting specific expressions from corpora. In an experiment on extracting sentiment expressions (see section 3.1), our approach can save 90% time.

2. In constructing training corpus for a binary classification, our approach can also save human labor. Furthermore, our approach annotate a pattern only once and can reduce the inconsistent annotation, especially when the annotator is tired after long time annotating. Agreement statistics support our analysis.

We have used our approach to extracting sentiment expressions from a domain reviews, Chinese polarity shifters from the corpus of product reviews, dish names from the corpus of food reviews; we also have used our approach to construct training corpus for a classifier to detect if a text has relationship with Internet games, and removed noise texts for a recognition system of flowers.

In the future, we would use our approach in extracting patterns of some binary relationship such as part-whole, hyponymy etc., and also relationship across sentences may be considered.

Acknowledgements

This research is supported by National Natural Science Foundation of China (No.61300156, No.61300152) and Fujian Provincial Department of Education (JAT160387).

5

References

Y. Choi and C. Cardie. 2009. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Fermín L. Cruz, José A. Troyano, F. Javier Ortega, and Fernando Enríquez. 2011. Automatic expansion of feature-level opinion lexicons. In *In Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2.011) (June 2011),*.

Valentin Jijkoun, Maarten de Rijke, and Wouter Weerkamp. 2010. Generating focused topic-specific sentiment lexicons. In *In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Yue Lu, Malu Castellanos, and Umeshwar Dayal Chengxiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon : An optimization approach. In *www2011*.

J. Pei, J. Han, Mortazavi-Asl B, H. Pinto, Q. Chen, U. Dayal, and Hsu M-C. 2001. Prefixspan: mining sequential patterns efficiently by prefix-projected pattern growth. In *In: Proceeding of the 2001 international conference on data engineering (ICDE?01), Heidelberg, Germany, pp 215?224*.

J.and Afshar R. Yan, X.and Han. 2003. Clospan: Mining closed sequential patterns in large databases. In *Proceedings of the 3rd SIAM International Conference on Data Mining. San Francisco, CA*.