Generate Compressed Sentences with Stanford Typed Dependencies towards Abstractive Summarization

Peng Li and **Yinglin Wang**
Department of Computer Science and Engineering, Shanghai Jiao Tong University
{lipeng, ylwang@sjtu.edu.cn}

Abstract

In this paper, we implement sentence generation process towards generate abstractive summarization which is proposed by (Genest and Lapalme, 2010). We simply use Stanford Typed Dependencies[1] to extract information items and generate multiple compressed sentences via Natural Language Generation engine. Then we follow LexRank based sentence ranking combined with greedy sentence selection to build final summary. Although the quantitative evaluation based on Rouge metric demonstrates poor performances, we believe that this sentence generation process make important role towards generate abstractive summarization.

1 Introduction

TAC2011 guided summarization task is to write a 100 word summary of a set of 10 newswire articles for a given topic, where the topic falls into a predefined category. A summary should cover all the aspects relevant to its category. Referring to accidents and natural disasters category, 7 aspects should be covered by the automatically generated summary. These aspects are what happened; date; location; reasons for accident/disaster; casualties; damages; rescue efforts/countermeasures. Additionally, an "update" component of the guided summarization task is to write a 100-word "update" summary of a subsequent 10 newswire articles for the topic, under the assumption that the user has already read the earlier articles.

The work proposed by (Genest and Lapalme, 2010; Genest et al., 2009) demonstrate many reasons why the next generation summarization system architecture should move from extractive way to abstractive way. Here we give more implement details about how to generate multiple compressed sentences.

2 Main Approach

2.1 Information Item Extraction

In (Genest and Lapalme, 2010), they define information items as subject-verb-object triple. Instead of defining some pruning rules, we use English grammatical relations defined by Stanford Typed Dependency Manual. Algorithm 1 present how to recognize possible information items. We found that possible information items contains many wrong triples, so we need Algorithm 2 to filter out correct information items.

2.2 Sentence Generation

We leverage information items to generate new sentences. For each information item, we have predicate, then need generate Noun Phrase (See Algorithm 4) to build subject and generate Verb Phrase(See Algorithm 5) to build object. Then call NLG to combine subject, predicate and object to become a new sentence(See Algorithm 3). We use Simplenlg[2] which is a simple Java API designed to facilitate the generation of Natural Language.

2.3 Summary Generation

Inspired by (Li et al., 2011), we want to order the compressed sentences so that the representative sentences can be ranked higher, then select top ranked sentences as long as the redundancy score (similarity) between a candidate sentence

[1] http://nlp.stanford.edu/software/dependencies_manual.pdf

[2] http://code.google.com/p/simplenlg/

Algorithm 1 Recognize Information Items

```
EnglishGrammaticalStructure eg
SET predicates
Collection typedDependencyCollection
for all td in typedDependencyCollection do
    TreeGraphNode gov = td.gov()
    GrammaticalRelation gr = td.reln()
    if gr = “nsubj″ or gr = “dobj or gr = “xcomp″
    or gr = “agent″ then
        predicates.add(gov)
    end if
end for
List tmpItems
Set subjects, objects
for all n in predicates do
    subjects = getSubjects(eg, n)
    if subjects.size() == 0 then
        continue
    end if
    for all s in subjects do
        objects = getObjects(eg, s)
        if objects.size()! = 0 then
            for all o in objects do
                item = new InformationItem(s, n, o)
            end for
        else
            item = new InformationItem(s, n, null)
        end if
        tmpItems.add(item)
    end for
end for
return tmpItems
```

Algorithm 2 Filter Information Items

```
Require: Run Algorithm 1 get tmpItems
List filteredItems, objs
TreeGraphNode subj, obj
for all item in tmpItems do
    subj = item.getSubject()
    obj = item.getObject()
    if !predicates.contains(subj) and obj! = null
    then
        objs.addobj
        items.add(item)
    end if
end for
for all item in tmpItems do
    TreeGraphNode s = item.getSubject(), p, o
    InformationItem newItem
    if predicates.contains(s) then
        for all TreeGraphNode obj in objs do
            p = item.getPredicate()
            o = item.getObject()
            newItem = new
            InformationItem(obj, p, o)
            items.add(newItem)
        end for
    end if
end for
return filteredItems
```

Algorithm 3 Generate compressed sentence

```
Require: build DependencyGraph
SPhraseSpec newSent
NPPhraseSpec subjectNp
VPPhraseSpec vp
TreeGraphNode s ,p
for all item in filteredItems do
    s = item.getSubject()
    subjectNp = generateNP(graph, s)
    newSent.setSubject(subjectNp)
    p = item.getPredicate()
    vp = generateVP(graph, p)
    newSent.setVerbPhrase(vp)
end for
```

Algorithm 4 Generate Noun Phrase

```
NPPhraseSpec np, tmpNp;
PPPhraseSpec pp;
Stack stack
stack.add(head)
while !stack.isEmpty() do
    children = grpah.adj(head);
    for all td in children do
        GrammaticalRelation gr = td.reln()
        if gr = “prep″ then
            pp = generatePrepP(td)
            np.setPostModifier(pp)
        else if gr = “nn″ or gr = “conj″ then
            tmpNp = generateNP(graph, td.dep())
            np.setPostModifier(tmpNp)
        else if gr = “det″ or gr = “num″ or gr =
        “amod″ then
            np.setPostModifier(td.dep())
        else
            continue
        end if
    end for
end while
```

Algorithm 5 Generate Verb Phrase

```
VPPhraseSpec vp
NPPhraseSpec dirobjNp, indirObjNp
vp.sertVerb(verb)
if object! = null then
    dirobjNp = generateNP(graph, object)
    vp.setObject(dirobjNp)
    children = grpah.adj(verb);
    for all td in children do
        GrammaticalRelation gr = td.reln()
        if gr = "iobj" then
            indirObjNp =
            generateNP(graph, td.dep())
            vp.IndirectObject(indirObjNp)
            break
        end if
    end for
else
    for all td in children do
        GrammaticalRelation gr = td.reln()
        if gr = "ccomp" then
            vp.setPostModifier(complement)
            break
        end if
    end for
end if
```

and current summary is under 0.5. This is repeated until the summary reaches a 100 word length limit. We use an LexRank algorithm to obtain top ranked sentences. LexRank (Erkan and Radev, 2004) algorithm defines a random walk model on top of a graph that represents sentences to be summarized as nodes and their similarities as edges. The LexRank score of a sentence gives the expected probability that a random walk will visit that sentence in the long run. A variant is called continuous LexRank improved LexRank by making use of the strength of the similarity links. The continuous LexRank score can be computed using the following formula:

$$L(u) = \frac{d}{N} + (1-d) \sum_{v \in adj[u]} p(u|v)L(v)$$

where $L(u)$ is the LexRank value of sentence u, N is the total number of nodes in the graph, d is a damping factor for the convergence of the method, and $p(u|v)$ is the jumping probability between sentence u and its neighboring sentence v. $p(u|v)$ is defined using content similarity function $sim(u, v)$ between two sentences:

	average ROUGE-2 recall		average ROUGE-SU recall	
	A	B	A	B
Run-1	0.04069	0.04436	0.07923	0.08217
Run-2	0.04200	0.03579	0.07476	0.07234

Table 2: Rouge Results

	average BE recall	
	A	B
Run-1	0.02207	0.02407
Run-2	0.02391	0.01877

Table 3: BE Results

$$p(u|v) = \frac{sim(u,v)}{\sum_{z \in adj[v]} sim(z,v)}$$

We use Jaccard similarity, Longest Common Substring, Levenshtein Distance to define $sim(u, v)$ as the similarity between two sentences.

3 Evaluation

TAC 2011 provides 44 topics for evaluation. Each topic includes a topic statement and 20 relevant documents which have been divided into 2 sets: Document Set A and Document Set B. Each document set has 10 documents, and all the documents in Set A chronologically precede the documents in Set B. Eight NIST assessors selected and wrote summaries for the 44 topics in the TAC 2011 guided summarization task, and assessors wrote 4 model summaries for each docset. All summaries were also automatically evaluated using ROUGE/BE.

In Run-1(summarizer ID is 36), we set the ranking threshold is 0.002, we set this value is 0.004 in Run-2(summarizer ID is 50). Table 1 is manual evaluation results, Table 2 is the ROUGE results, Table 3 is the BE results.

3.1 Analysis

The pyramid score is better than our TAC 2010 summarization system, but the linguistic quality

	average modified (pyramid) score		average numSCUs		average numrepetitions		macroaverage modified score with 3 models		average linguistic quality		average overall responsiveness	
	A	B	A	B	A	B	A	B	A	B	A	B
Run-1	0.215	0.172	2.955	2.068	0.295	0.136	0.213	0.169	1.364	1.477	1.773	1.727
Run-2	0.223	0.156	3.000	1.909	0.295	0.227	0.221	0.154	1.841	1.864	1.977	1.750

Table 1: Manual Evaluation

is lower then before, The average ROUGE and BE scores get very lower performance. The main reason lies in the sentence generation process. In the future, we will explore the English grammar deeply to find new algorithm to get better results.

4 Conclusions

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC No. 60773088), the National High-tech R&D Program of China (863 Program No. 2009AA04Z106), and the Key Program of Basic Research of Shanghai Municipal S&T Commission (No. 08JC1411700).

References

Erkan, Günes. and Dragomir Radev. 2004. LexRank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479.

Genest, Pierre-Etienne and Guy Lapalme. 2010. Text generation for abstractive summarization. In *Proceedings of the Second Text Analysis Conference, Gaithersburg, Maryland, USA: National Institute of Standards and Technology*.

Genest, Pierre-Etienne, Guy Lapalme, and Mehdi Yousfi-Monod. 2009. Hextac: the creation of a manual extractive run. In *TAC 2009 Workshop, Gaithersburg, Maryland, USA: National Institute of Standards and Technology*.

Li, Peng, Yinglin Wang, Wei Gao, and Jing Jiang. 2011. Generating aspect-oriented multi-document summarization with event-aspect model. In *Proceedings of Empirical Methods in Natural Language Processing*.

SJTU_CIT at TAC 2011: RTE Track

Guangxin Wang and **Peng Li** and **Yinglin Wang**
Department of Computer Science and Engineering, Shanghai Jiao Tong University
{wgxin, lipeng, ylwang@sjtu.edu.cn}

Abstract

In this paper, we present a system that uses machine learning algorithms combining with various knowledge for the task of recognizing textual entailment. The features chosen quantify lexical, syntactic and semantic level matching between text and hypothesis sentences. We analyze how different knowledge resources and classifiers could impact on the final overall performance of the RTE classification of two-way task. The evaluation results are not as good as we hope, but encourage us to make improvement in next version.

1 Introduction

In recent years, the task of recognizing textural entailment (RTE) became a hot topic in natural language processing community. Given two text fragments, the system can determine whether the meaning of one text can be inferred from the other text. More specifically, textual entailment is defined as a directional relational ship between two text fragments, termed Text (T) and Hypothesis (H). For examples,

- T: He bought a pen in the store.
- H: He owes a pen.

Obviously, T infers H. This means that H maybe entailed by incorporating more prior knowledge that would enable its inference from T, but it should not be entailed by that knowledge alone. In other words, it is not allowed to validate H's truth regard-less of T.

Such kind of RTE system can be very useful in many applications. Recent application can be found in Twitter which is used for remove redundant information when generate summarizations from tweets.

The challenge of this task is that it needs in-depth inference instead of just comparing the word similarity between two sentences. For example, from "owe" happens after buy we know that these two sentences has a chronological relation. Word "bought" align with word "owe". However, finding correct word alignment is very difficult due to the fact that possible matches could be exponential in the number of words (Zhang et al., 2010).

In this paper, we extends the work proposed by (Ren et al., 2009), focus on exploring diverse lexical, syntactic and semantic know-ledge in feature-based text entailment using mixture of classifiers. Our study illustrates that the semantic resources contributes to most of the performance improvement. We also demonstrate how semantic information such as Wikipedia, WordNet, ConceptNet and VerbOcean can be used in the feature-based framework.

2 Main Approach

2.1 Classification Approach

We use Support Vector Machine(SVM), Multilayer Perceptron(MLP), Decision Trees(DT) and AdaBoost(AB). Support Vector Machine (SVM) is a supervised ma-chine learning technique motivated by the statistical learning theory (Vapnik, 1998). Based on the structural risk minimization of the statistical learning theory, SVMs seek an optimal separating hyper-plane to divide the training examples into two classes and make decisions based on support vectors which are selected as the only effective instances in the training set. In this paper, we use the binary-class LibSVM developed by (Chang and Lin, 2011). The Decision Trees are interesting because we can see what features were selected from the top levels of the trees. AdaBoost were selected because it is known for achieving high performance, and MLP was used because it has achieved high performance in others NLP tasks.

2.2 Features

2.2.1 Lexical Distance

We use Jaccard Similarity, Longest Common Substring and Levenshtein distance as the lexical distance features. Jaccard Similarity is a similarity measure that compares the similarity between two sentences. When applying to compute similarity between T and H, it is defined as the size of the intersection of the words in T and H compared to the size of the union of the words in T and H. The Longest Common Sub-string (LCS) of T and H will find the longest string that is a substring of both T and H. It is found by dynamic programming. The standard Levenshtein distance is motivated by the good results obtained as a measure of similarity between two strings. This distance quantifies the number of changes (character based) to generate one text string (T) from the other (H). Using stems, this measure improves the Levenshtein over words. The lexical distance feature based on Levenshtein distance is interesting because works to a sentence level.

2.2.2 Dependency Tree

The dependency features includes information about the words, part-of-speeches and phrase labels of the words on which the mentions are dependent in the dependency tree derived from the syntactic full parse tree. We use Stanford Typed Dependencies to build Dependency Tree of the sentence. The dependency features are expressed as a dependency pair (Head, Modifier) (Nielsen et al., 2006), to cover the cases when the same word lemmas appear in different grammatical relations, possibly also with different parts of speech.

2.2.3 WordNet, ConceptNet, VerbOcean

Word similarity based on WordNet has been widely used in RTE. Its strength is that it contains a large amount of words, by calculating the distance between two words it's easily to get the similarity. However, in alignment in RTE is usually between words that have different POS or between single word and phrase. For example, Hunter has some relation to phrase Killing a prey. Therefore, we introduce a knowledge base called ConceptNet. Verb and verb similarity is also important in decision making. The knowledge bases listed above are relatively week in comparing verbs. Thus a few works had put effort in extracting verb pair through plain text. Here we use VerbOcean, it is a broad-coverage semantic network of verbs, it defines several relations between verb.

2.2.4 Wikipedia

Wikipedia is a vast, constantly evolving resource of interlinked articles providing a giant multilingual database of concepts and semantic relations. It serves as a promising resource for natural language processing and many other research areas. (Gabrilovich and Markovitch, 2007) computes two phrase relation by representing them with two vector of wiki entry and calculating the similarity between them. Wikipedia Miner (Milne and Witten, 2009) is a freely available toolkit for navigating and making use of content of Wikipedia. It provides simplified, object-oriented access to Wikipedia's structure and content and offers several services to help users to search for entities, comparing the relation between entities and wikifying snippets of texts. Assuming that each Wikipedia topic serves as a semantic concept, we make use of Wikify service of the toolkit to annotate document collection with links to relevant Wikipedia concepts. Wikiminer also provides a semantic relatedness measure between two concepts using category hierarchy and textual content of respective concepts. After the Wikification of document collection, we use the relatedness measure (RM) of a concept with all other concepts in the document collection as its importance measure. We obtain sentence similarity by combining Wiki and Word similarity linearly.

Run ID	Micro Average Precision	Recall	F measure
Run-1	18.52	27.60	22.17
Run-2	16.50	38.30	23.07
Run-3	17.92	33.33	23.31

Table 1: Main Test Results

Resource	Micro Average Precision	Recall	F measure
Verbocean	15.30	19.50	17.14
Wikipedia	15.49	11.62	13.28

Table 2: Ablation Test Results

3 Evaluation

The experimental data set we use came from RTE 6. After getting all features, we use Support Vector Machine(SVM), Multilayer Perceptron(MLP), Decision Trees(DT) and AdaBoost(AB) to train a classification model.

3.1 Main Test

We submitted three runs, executed with different thresholds. The results are shown in table 1. We can see that we still have many works to do to improve the performance of our system. Compared to past RTE result, these results are relatively poor. We believe that it is mainly due to the sentences' incomplete information and bad feature generation procedure.

3.2 Ablation Test

In this test we submitted two runs with different knowledge resources as shown in table 2. They all have negative impact of Micro Average Precision, Recall and F measure.

4 Conclusions

In this paper, We propose a textual entailment recognition framework and implement a system of classification which takes lexical, syntactic and semantic features as considered. Official results show that our system gets worse performance of all participating systems. That means there are still many part of the system needed to be improved in the future.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (NSFC No. 60773088), the National High-tech R&D Program of China (863 Program No. 2009AA04Z106), and the Key Program of Basic Research of Shanghai Municipal S&T Commission (No. 08JC1411700).

References

Chang, C.C. and C.J. Lin. 2011. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Gabrilovich, E. and S. Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 1606–1611.

Milne, D. and I.H. Witten. 2009. An open-source toolkit for mining wikipedia. In *Proc. New Zealand Computer Science Research Student Conf*, volume 9.

Nielsen, R., W. Ward, and J.H. Martin. 2006. Toward dependency path based entailment. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, pages 44–49.

Ren, H., D. Ji, and J. Wan. 2009. Whu at tac 2009: A tri-categorization approach to textual entailment recognition. In *Preproceedings of the Text Analysis Conference (TAC)*.

Vapnik, V.N. 1998. Statistical learning theory.

Zhang, Xinhua, Peng Li, and Yinglin Wang. 2010. Sjtu_cit at tac 2010 : Rte track. In *TAC 2010 Workshop, Gaithersburg, Maryland, USA: National Institute of Standards and Technology*.