An extractive supervised two-stage method for sentence compression

Dimitris Galanis and Ion Androutsopoulos

Natural Language Processing Group
Department of Informatics
Athens University of Economics and Business
Introduction

- **Sentence compression**: produce a shorter form of a sentence, which is grammatical and retains the most important information.

- **Example**:
  - **source**: *Then last week a second note, in the same handwriting, informed Mrs Allan that the search was on the wrong side of the bridge.*
  - **compression**: *Last week a second note informed Mrs Allan the search was on the wrong side of the bridge.*

- **Examples of applications of sentence compression**:
  - text summarization
  - displaying texts on small screens

- **Extractive compression**: Only word deletions are permitted.
Our Approach

- Our algorithm compresses sentences in two stages.

1. **Generate candidate compressions**
   - Source sentence

2. **Ranking candidate compressions**
   - Candidate compressions

3. **Compressed sentence**
Generating candidate compressions

- **Generate candidates** by deleting edges of the dependency tree of the source sentence.
  - For every edge there are **3 possible actions** leading to 3 different candidates:
    - **Retain** the edge (not_del).
    - **Delete** it along with the **subtree** (del_l).
    - **Delete** it along with the **uptree** (del_u).
  - Sentence with m words → at most $3^{m-1}$ possible candidate compressions.
- In practice we generate fewer candidates:
  - If we delete an edge along with its subtree, then there are no separate actions for the subtree’s edges.
  - If an action has **low probability** (as judged by a MaxEnt classifier, next slide), we don’t use in any of the candidates.
Generating candidate compressions

- We consider the edges in a top-down DFS manner.

We ignore actions that are considered unlikely by the MaxEnt classifier.
Training the MaxEnt classifier

- Learning probabilities for actions:
  - We use a MaxEnt (ME) classifier trained on pairs of source and compressed (gold) dependency trees.

Examples of features:
- label of the dependency edge
- POS tags of head and modifier
- etc

Training vectors for dependency edges:
- `<ccomp, ..., del_u>`
- `<nsubj, ..., not_del>`
- `<prep, ..., del_l>`
Ranking with linear combination

- We need a function $F(c_i | s)$ that will rank the candidate compressions.
- **1st ranker we tried:** Linear combination of **grammaticality** and **importance rate** (LM-Imp model)
  - A compression rate penalty factor $\alpha$ is included, to bias our method towards generating shorter or longer compressions.

$$F(c_i | s) = \lambda \cdot \text{Gramm}(c_i) + (1 - \lambda) \cdot \text{ImpRate}(c_i | s) - \alpha \cdot CR(c_i | s)$$

$$\text{Gramm}(c_i) = \log P_{LM}(c_i)^{1/m} = \frac{1}{m} \cdot \log \left( \prod_{j=1}^{m} P(w_j | w_{j-1}, w_{j-2}) \right)$$

$$\text{ImpRate}(c_i | s) = \frac{\text{Imp}(c_i)}{\text{Imp}(s)}$$

$$\text{Imp}(\xi) = \sum_{w_i \in \xi} tf(w_i) \cdot idf(w_i)$$

$$CR(c_i | s) = |c_i|/|s|$$
Ranking with SVR

- **2nd ranker we tried**: Support Vector Regression (SVR) model
- SVR models are trained using $l$ training vectors and learn a function $f : \mathbb{R}^n \to \mathbb{R}$

### Features ($x_i$):
- grammaticality
- importance rate
- average depth of deleted words
- which POS tags were deleted

### Score ($y_i$): similarity between candidate and gold

#### Training vectors
- Candidate 1 of source sentence 1 $\rightarrow$ <0.1, 0.34, ..., 0.47, 0.8>
- Candidate 2 of source sentence 1 $\rightarrow$ <0.2, 0.31, ..., 0.42, 0.8>
- ...
- Candidate n of source sentence k $\rightarrow$ <1.0, 0.44, ..., 0.41, 0.5>

#### Testing vectors
- <0.1, 0.36, ..., 0.42, ?>
- ...

$n$ source sentences + their candidates + gold compressions (one gold per source)

source sentences + their candidates

A score for each candidate
SVR’s similarity measures

- **Two versions of similarity** between gold (g) and candidate (c_i):
  - **Grammatical relations overlap:**
    - \( d \) denotes the dependencies of a sentence
    - \( F_1 \) is the F-score (\( \beta = 1 \))
    - **SVR-F1 model**
  - **Tokens accuracy and grammaticality:**
    - is the percentage of tokens of \( s \) that were correctly retained or removed in \( c_i \)
    - **SVR-TokAcc-LM model**

\[
y_i = F_1(d(c_i), d(g)) - \alpha \cdot CR(c_i|s)
\]

\[
y_i = \lambda \cdot TokAcc(c_i|s, g) + (1 - \lambda) \cdot Gramm(c_i) - \alpha \cdot CR(c_i|s)
\]
Experiments

- We used Edinburgh's “written” sentence compression corpus ([http://homepages.inf.ed.ac.uk/s0460084/data/](http://homepages.inf.ed.ac.uk/s0460084/data/))
- 3 parts:
  - training, development, and test.
- Training part used to:
  - train the MaxEnt model of Stage 1
  - train the SVR model of Stage 2.
- With $a = 0$, we varied $\lambda$ and selected the value that gives compression rate approximately equal to human compression.
- Then we varied parameter $a$ (compression rate penalty factor), which is available in all models.
- ME threshold $t = 0.2$
  - Limits the number of candidates (< 10,000) for almost every source.
  - Tuned in preliminary experiments.
Selecting our best configuration (with automatic evaluation)

- **F1** is the avg F1-score of the **dependencies** of system compressions against gold compressions on the **development set**.
  - F1 has been shown that correlates well with human judges
- **SVR-TokAcc-LM** is the **best configuration of our system** for most compression rates.
Comparing to state-of-the-art (with human judges)

- We compared **SVR-TokAcc-LM against T3** (Cohn & Lapata 2009)
- **T3** is a *state-of-the-art* sentence compression system.
  - Best reported results on Edinburgh's “written” corpus.
- 80 source test sentences.
- **4 judges** were asked to rate 240 compressions.
  - 80 compressions of T3, 80 compressions of our system, and 80 gold compressions.

| system  | G    | M    | Ov    | F1 (%) | CR (%) |
|---------|------|------|-------|--------|--------|
| T3      | 3.83 | 3.28 | 3.23  | 47.34  | 59.16  |
| SVR     | 4.20 | 3.43 | 3.57  | 52.09  | 59.85  |
| gold    | 4.73 | 4.27 | 4.43  | 100.00 | 78.80  |

Table 2: Results on 80 test sentences. G: grammaticality, M: meaning preservation, Ov: overall score, CR: compression rate, SVR: SVR-TokAcc-LM.
Conclusions

- A new sentence compression method.
  - **Candidate compressions** generated by considering **three actions per dependency edge** (retain, delete subtree, delete uptree).
  - A **MaxEnt** classifier rejects unlikely actions.
  - An **SVR** model ranks the candidate compressions.
  - Our method has **comparable (or better)** results to a **state-of-the-art** sentence compression system.

- Future plans:
  - Use more complex dependency tree transformations.
  - Experiment with different sizes of training data.
  - Add more features.

- Questions?