Imperial College
London

Individual Project

Department of Computing

Imperial College of Science, Technology and Medicine

Using Answer Set Grammars For Text Summarization

Author:
Julien Amblard

Supervisor:
Alessandra Russo

Helper:
David Tuckey

Second Marker:
Krysia Broda

ASG Author:
Mark Law

Thursday 21st May, 2020

Submitted in partial fulfillment of the requirements for the Computing MEng of
Imperial College London

Contents

Chapter 1 Introduction

1 General Problem

In general, the task of summarization in NLP (natural language processing) is to produce a shortened text which covers the main points expressed in a longer text (given as input). In order to do this, a system performing such a task must analyze/process the input to be able to extract from it the most important information.

2 Specific Problem

There are three different overlapping problems we may wish to pursue with the use of answer set programming (ASG):

1. Given a short text of about a page long, for example a short story aimed at young children, to provide a summary in multiple sentences. The goal here would be to identify which sentences in the passage are important, extract only these, and then possibly apply some post-processing to link them a bit better.

2. Given a very short text less than a page long comprised of very brief sentences, for example reading comprehension exercises for Key Stage 1 students, to provide a summary of just one or two sentences in length. What is important here is to establish which *chunks* (i.e. subparts of sentences) are important, thereby semantically learning the main descriptions and actions occurring in the text. From this, we can construct a meaningful *abstract* (see Chapter **??**).

3. Given a short text of about a page long (as in the first problem) as well as a summary of a few sentences, to establish whether the latter is a summary of the passage. The goal here would be to understand the semantics of both inputs, use a metric to determine how well they match, and then apply a decision criterion to give a boolean answer.

3 Objectives

TODO

Chapter 2 **Background**

- Learning answer sets -¿ learning ASG
  - Encoder-decoder overview

Chapter 3 **Contributions**

1 Architecture Overview

The main pipeline, which performs story summarisation, is made of three main steps: the Preprocessor, multiple calls to ASG, and finally post-processing and scoring (as seen in Figure 3.1. A description of each step can be found in the following chapters.

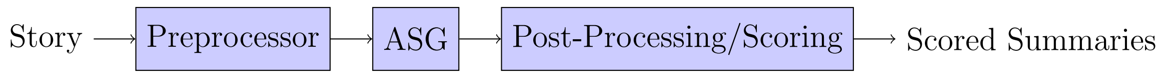

Figure 3.1: Main Pipeline

- Do same as project (main core with acronym like SumASG, then SumASG* to fix/build on top of foundation) - Use Peter Little examples - Appendix with summary generation rules - Talk about expandability (talk about mechanisms) - Very nice to not be able to generate grammatically incorrect with ASG - Describe action predicates as high level semantic descriptor of all possible actions that can happen in sentences - Think about initial motivation (have definition of what is summary, NN does not) 1. NNs need lots of data and time to summarize 2. ASG can give results with short list of rules, pre/post-processing and carefully constructed structure - Maybe formalize mathematically task of summarization (with CFG, BK, E+, E-) 1. CFG is language, BK is leaf nodes, result is actions 2. CFG is language, BK is leaf nodes, E is actions, result is summaries - For report think about how to formalize task of summarization in ASG (how thought evolve) 1. Originally just create summaries from actions 2. Now preprocess to prune and homogenize, produce and score summaries, take best and fix grammar 3. Use example of Peter Little to showcase pipeline - Compare with NN 1. Randomize action(...) to generate summary(...) on trained ASG 2. Train NN to generate same summary(...) 3. Show framework is sane and expandable (computationally tractable) 4. Compute Rouge score (PyRouge, must clone repo into project) on ASG and NN - Final goal: take story-specific ASG and general rules to generate summaries, then use top 5/10

Chapter 4 **Preprocessor**

1 Overview

In order to... as shown in Figure 4.1.

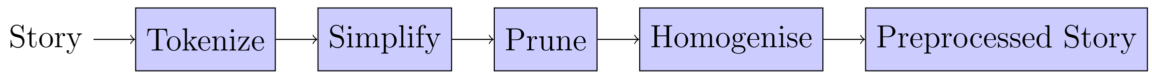

Figure 4.1: Preprocessor Steps

2 Tokenization and Simplification

3 Sentence Pruning and Homogenisation

Choice: 1. Simplify using Python script (faster) 2. Make ASG format more complex (new information probably lost in summary anyway)

- Punctuation - [?]: remove clause (helps avoid negation for rhetorical questions) - [!—,—;—:]: transform into '.' - [———]: delete inner part - Multiple clauses (conjunctive or main+auxiliary): split into multiple sentences - Adverbs: move to end - Possessive pronouns and interjections: remove - Prepositions: remove when at start of sentence - Contractions: expand - Acronyms: remove punctuation - Dependant clauses: split into separate sentence (remove if there is no punctuation) - Verbless sentences: remove - Subordinating conjunctions: split into separate sentence - Preposition clauses: remove if after object - Complex proper nouns: collapse into CamelCase - Proper nouns: replace occurrences of pronouns with relevant proper noun (idea: if they are used in the story then there should be little ambiguity) - Conjunction of common nouns from same lexical field: replace with hypernym (pluralize if items are plural and hypernym plural is used in English)

Chapter 5 **ASG**

1 Overview

Our use ASG is two-fold. Firstly, we pass in each sentence from the story to ASG to obtain its semantic representation in ASP. Secondly, we take these actions and use ASG rules to generate possible summary components. These will later be post-processed and turned into actual valid summaries. A diagram of the two ASG steps is shown below in Figure 5.1.

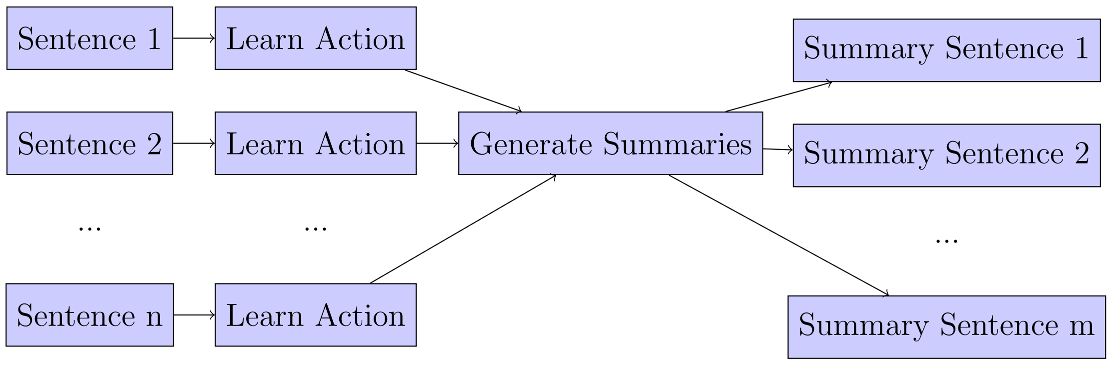

Figure 5.1: ASG Steps

2 Learning Actions

3 Generating Summary Sentences

- Learning is not really learning (ASG never learns how to summarize, we build in rules of feature extraction)

http://universalteacher.org.uk/lang/engstruct.htm

Ideas: - final fix-up using language_checker.fix - hard-code determiners into derivations (https://www.ef.com/wwen/english-resources/english-grammar/determiners/) - use lots of simple/precise rules rather than complicated/general ones to minimize ss - keep rules as restricted as possible, when concept implemented over time add missing rules - to avoid having to add grammar constraints try and rely on grammar of input

- reduce search space using mode bias (simple example: 396-¿16, very complicated example: 9477-¿1044) - for learning actions do one sentence at a time to minimize ss - pick best summary according to TTR*

action(INDEX, VERB, SUBJECT, OBJECT) summary(VERB, SUBJECT, OBJECT)

verb(INDICATIVE_FORM, TENSE) subject(NOUN, DET, ADJ_OR_ADV) object(NOUN, DET, ADJ_OR_ADV) noun(NAME) adj_or_adv(NAME) det(...) compound(FIRST, SECOND) for verbs conjunct(FIRST, SECOND) learn both

* Type-Token Ratio (TTR): The basic idea behind that measure is that if the text is more complex, the author uses a more varied vocabulary so there's a larger number of types (unique words). This logic is explicit in the TTR's formula, which calculates the number of types divided by the number of tokens. As a result, the higher the TTR, the higher the lexical complexity.

Chapter 6 **Post-Processing / Scoring**

1 Overview

Once we have obtained potential sentences from ASG to be used in a summary, we can now post-process these as explained in Section 2. By combining them in different ways, we are able to form summaries. From these, we will retain the highest scoring ones, according to the metric detailed in Section 3. A diagram illustrating these steps is shown below in Figure 6.1.

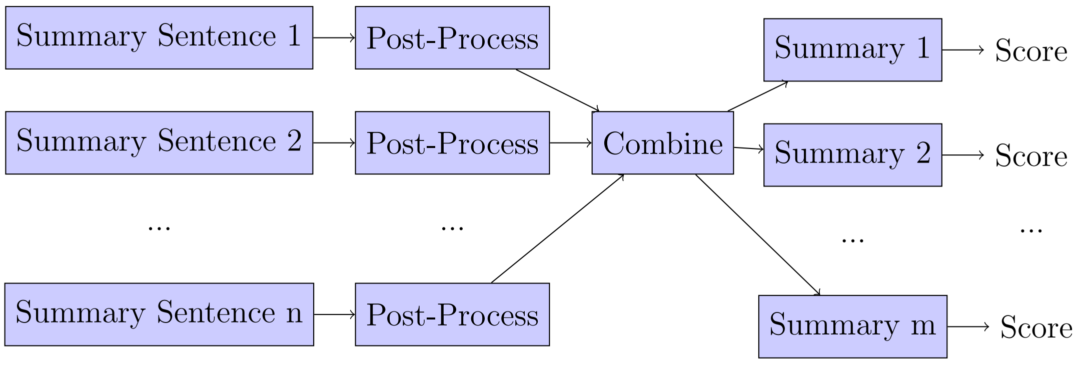

Figure 6.1: Post-Processing / Scoring Steps

2 Summary Creation

3 Scoring

- Sub diagrams

Chapter 7 Evaluation

1 General Idea

2 Dataset

3 Results

For for each story: 1. Pick predefined lexical field (topic) 2. Pick a single pronoun (p) 3. Pick a single proper noun (pn) 4. For each sentence: - Subject: p, pn, or synonym/hyponym/hypernym of topic with optional common adjective for it - Verb: verb from same lexical field as topic if possible, otherwise random - Object: p, pn, or holonym/meronym of subject with lexical field of currently used common nouns

Chapter 8 **Literature Review**

- Reread to refer
    - Compare approaches

Appendix A. **POS Tags**

Tag	Description
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

Table A.1: [**?**] List of position of speech (POS) tags

Appendix B. **ASG**

TODO