makes a long story short.

summarization

CSC401/2511 – Natural Language Computing – Spring 2014
Lecture 8-1 Gerald Penn, slides by Frank Rudzicz
University of Toronto
Summarization

- **Summarization**: *n.* the act of producing a shortened version of a text or collection of texts (i.e., a *summary*) that preserves the most important points.
Examples of summaries

- Russia fights Napoleon and a Natalia likes Boris.
- Gregor turns into a bug.
- Don’t sit on a wall if you’re an egg.
- Girl kills a woman, forms a gang, and kills again.
Headline news

- News articles are often shortened to one or two sentences. These summaries:
  - convey the most important aspects of their articles,
  - are often collected together in a group of summaries for easy scanning by the reader.
Abstracts

• Abstracts are often author-generated and time-saving.

An Incremental Interpreter for High-Level Programs with Sensing

Giuseppe De Giacomo
Dipartimento di Informatica e Sistemistica
Università di Roma “La Sapienza”
Viale S. Asia 113, 00198 Rome, Italy
degiacomo@dis.uniroma1.it

Hector Levesque
Department of Computer Science
University of Toronto
Toronto, Canada M5S 3H5
hector@cs.toronto.edu

Abstract

Like classical planning, the execution of high-level agent programs requires a sequence to look all the way to a final
goal state before even a single action can be taken in the world. This is a serious problem in practice for large
programs. Furthermore, the problem is compounded in the presence of sensing actions which provide necessary
information, but only after they are executed in the world. To deal with this, we propose (characteristic formally in the
situation calculus, and implemented in logic) new mechanisms for interpreting high-level programs and a new high-level
language construct, which together, and with actions of generality, allow much more control to be exerted over when
actions can be executed. We argue that such a scheme is the only practical way to deal with large agent programs
containing both nondeterminism and sensing.

Introduction

In [4] it was argued that when it comes to providing high
level control to autonomous agents or robots, the notion of
high-level program execution offers an alternative to classical
planning that may be more practical in many applications.
Briefly, instead of looking for a sequence of actions a
such that

\[
\text{Action}(a) \perp \text{Legal}(a, S_0) \land \psi(a, S_0)
\]

where \( \psi \) is the goal being planned for, we look for a se-
quence \( \delta \) such that

\[
\text{Action}(\delta) \perp \text{Do}(\delta, S_0, a_1, S_1)
\]

to find a sequence with the right properties. This can involve
considerable search when \( \delta \) is very nondeterministic,
but much less search when \( \delta \) is more deterministic. The fea-
sibility of this approach for AI purposes clearly depends on
the expressiveness of the programming language in ques-
tion. In [4], a language called ConsLog is presented, which
in addition to non-determinism, contains features for se-
quence, iteration, conditionals, concurrency, and prior-
itized interrupts. In this paper, we extend the expressive
power of this language by providing much finer control over
the nondeterminism, and by making provision for sensing
actions. To do so in a way that will be practical even for
very large programs requires introducing a different style
of on-line program execution.

In the rest of this section, we discuss on-line and off-line
execution informally, and show why sensing actions and
nondeterminism together can be problematic. In the follow-
ing section, we formally characterize program execution in
the language of the situation calculus. Next, we describe an
incremental interpreter in Prolog that is correct with respect
to this specification. The final section contains discussion
and conclusions.

Off-line and On-line execution

To be compatible with planning, the ConsLog interpreter
presented in [4] executes in an off-line manner, in the
sense that it must find a sequence of actions constituting an
entire legal execution of a program before actually execut-
ing any of them in the world. Consider, for example, the
following program:
Kinds of summaries

- Summaries can be produced according to several features.
  - **Perspective**: whether the summary is **informative** on its own or if it is merely meant to be **indicative**.
  - **Composition**: whether the summary is **extracted** directly from the source or **synthesized** from scratch.
  - **Orientation**: whether the **author’s view** is preserved or if the summary reflects the **user’s interest**.
  - **Source**: whether we summarize a **single** document or **multiple** documents.
  - **Background**: whether we can assume that the reader has **prior knowledge** or not.
Summarization by extraction

• **Extractive summarization** involves identifying **important sections** in the original text, and **copying** those sections into the summary.

• e.g., The review next week will be *mostly* an extractive summary of the course.
  • i.e., important slides will be extracted from other lectures and agglutinated together.
Summarization by extraction

How do we determine which sentences are relevant?

From Jurafsky & Martin
Determining relevance

• The relevance of sentences and phrases within the text can be approximated by:

  • **Position:** The location of the phrase in the document.
  • **Cues:** The presence of certain words that indicate relevance (e.g., “crucially”, “in conclusion”).
  • **Cohesion:** The distribution of words and their co-occurrences across the document.
  • ...
Position-based method

- Important sentences tend to occur in predictable positions within paragraphs and sentences.
  - Baxendale (1958) found that in 85% of 200 paragraphs, the **most important** sentence was also the **first**.
  - In the **news domain**, early paragraphs are often very important.
Optimum position policy (OOP)

- **Claim**: Important sentences are located at positions that are **genre-dependent**; these positions can be determined automatically through training (Lin and Hovy, 1997).

- **Corpora**: 13,000 newspaper articles (ZIFF corpus) and the Wall Street Journal.

- **Step 1**: For each article, determine the overlap between its **index terms** and its sentences.

- **Step 2**: Determine a partial ordering over the locations where sentences contain important words.
Optimum position policy (OPP)

• The OPP for the ZIFF corpus is:

\[ T \succeq P_2 S_1 \succeq P_3 S_1 \succeq P_2 S_2 \succeq \{P_4 S_1, P_5 S_1, P_3 S_2\} \]

where \( T = \) title, \( P = \) paragraph, and \( S = \) sentence.

• The OPP for the Wall Street Journal is:

\[ T \succeq P_1 S_1 \succeq P_1 S_2 \succeq \cdots \succeq P_2 S_1 \succeq \cdots \]

• By taking the most important 10% of a document according to these orderings, we can cover 91% of the salient words in that document.
Cue-phrase method

• Important sentences often contain ‘bonus phrases’, e.g., ‘significantly’, ‘in this paper, we show’, and ‘in conclusion’.
• Unimportant sentences contain ‘stigma phrases’ such as ‘hardly’, ‘incidentally’, and ‘ongoing’.

• These cue phrases can be detected automatically (Kupiec et al., 1995; Teufel and Moens, 1997).

• We increment a sentence score for each bonus phrase, and decrement its score for each stigma phrase.
Which words are cues?

• Liu and Hovy (1998) scored words \( w \) in ‘high-yield’ sentences from newspaper and scientific publications according to two measures:

  • \( S_1(w) \) is the **count** of \( w \) in the summary (e.g., abstract) **over** its total occurrence in a document.

  • \( S_2(w) = S_1(w) \cdot \frac{df_w}{D} \) where \( D \) is the total number of training documents, and \( df_w \) is the number of documents in which the word \( w \) appears.
Cue-phrase method

- Results: the following phrases are good ‘bonus’ cue phrases:

<table>
<thead>
<tr>
<th></th>
<th>Method 1</th>
<th>Method 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>phrase</td>
<td>$S_2$</td>
</tr>
<tr>
<td>7.666</td>
<td><em>this paper presents</em></td>
<td>3.432</td>
</tr>
<tr>
<td>7.666</td>
<td><em>machine learning algorithm</em></td>
<td>2.889</td>
</tr>
<tr>
<td>6.909</td>
<td><em>present the result</em></td>
<td>2.266</td>
</tr>
<tr>
<td>6.888</td>
<td><em>paper we have</em></td>
<td>2.279</td>
</tr>
<tr>
<td>6.340</td>
<td><em>this paper we</em></td>
<td>2.044</td>
</tr>
</tbody>
</table>
Cohesion-based methods

• Important sentences and paragraphs are highly ‘connected’ entities in larger semantic structures.
  • Important concepts occur more frequently in important sentences.

• How to determine cohesion?
  • Word co-occurrence,
  • Local grammatical relations,
  • Co-reference,
  • Lexical similarity.
Cohesion: word co-occurrence

• Apply IR methods at the document level.
  • Where IR has **documents within collections**, now we have **paragraphs within documents**.
  • Use traditional IR-based word-similarity measures to determine which paragraph is related to the most other paragraphs – that paragraph is the most salient.
Cohesion: lexical chains

- **Lexical chain**: *n.* a sequence of related words that can span an entire text. Lexical chains can provide context for the resolution of ambiguous terms.
  e.g., *Toronto* → capital → city

- The lexical chains method uses the assumption that important sentences are traversed by strong chains, $C$.
  - $\text{Strength}(C) = \text{length}(C) - \text{numDistinctWords}(C)$ (Barzilay and Elhadad, 97).
  - Take the strongest chains, and copy the first sentences spanned by these chains into the summary.
Cohesion: lexical chains example

• Based on Morris and Hirst (1991)

But Mr. Kenny’s move speeded up work on a machine which uses micro-computers to control the rate at which an anaesthetic is pumped into the blood of patients undergoing surgery. Such machines are nothing new. But Mr. Kenny’s device uses two personal-computers to achieve much closer monitoring of the pump feeding the anaesthetic into the patient. Extensive testing of the equipment has sufficiently impressed the authorities which regulate medical equipment in Britain, and, so far, four other countries, to make this the first such machine to be licensed for commercial sale to hospitals.
Cohesion: co-reference

• **Co-reference**: *n.* when multiple expressions refer to the same entity or event.
  e.g., *Stephen Harper* ate a cat that *he* stole from a *small child*. *The prime minister* laughed at *the child*.

• Important sentences are those that are traversed by a large number of **co-reference chains**.
  • A preference is placed on those chains that traverse the title or first paragraph of a document.
Naïve Bayes classification

• We can treat summarization as a sequence of binary classification problems:
  • Every sentence is either in or out of the summary.

• Bayes’s decision rule is to select the outcome that is most probable, given contextual features, $f_i$:

$$\max \left( P(s \in \text{summary} | f_1 \ldots f_k), P(s \notin \text{summary} | f_1 \ldots f_k) \right)$$

• **Question**: How can we rewrite $P(\sigma | f_1 \ldots f_k)$ to make it easier to measure? *Hint*: consider the title of this slide.
Naïve Bayes classification

• Recall **Bayes’s Theorem** (aka Bayes’s Rule):

\[ P(\sigma | f_1 \ldots f_k) = \frac{P(f_1 \ldots f_k | \sigma)P(\sigma)}{P(f_1 \ldots f_k)} \]

• We assume that all features of relevance are **conditionally independent**,

\[ P(f_1 \ldots f_k | \sigma) = \prod_{i \leq j \leq k} P(f_j | \sigma) \]

• And we can use relative frequency in annotated corpora for learning \( P(f_j | \sigma) \) using **maximum likelihood estimation**.
Controlling summary length

• Alternatively, we can control the length of our summary by simply computing

\[ \arg\max_{s_i} P(s_i \in \text{summary} | f_1 \ldots f_k) \]

over all sentences \( s_i \) in the original document and repeating until either

• we’ve extracted a predefined number of sentences, or
• the best probability of a new sentence is below some threshold.
Aspects of extractive summarization

- Summaries produced by extraction can be **hard to read**, **misleading**, or **incoherent**.

- If a pronoun in an extracted sentence (e.g., ‘he’) refers to an entity in a previous sentence that is **not** extracted, that pronoun may be impossible to resolve by the reader.

- **Discourse** or **argument connectives** can become inappropriate in summaries. For example, it would be very odd if the first sentence in a summary began with ‘Finally’.

- Parts of extracted sentences can be unimportant.
Aspects of extractive summarization

• Extractive summarization can be improved.
  • Argument structure can be used to determine relevance.
  • **Cut-and-paste** summarization extracts **phrases** rather than sentences and **combines** those phrases to **synthesize** new, natural-sounding sentences.
  • We can **compare** summaries generated from multiple **related** texts to see which sentences/phrases seem most important.
  • We can tweak the features used in extraction by evaluating our summaries within human-based **tasks**.
Evaluation of summarization

• As in other domains, we can evaluate a summarizer extrinsically (within a task) or intrinsically (independent of task).

• E.g., we might ask subjects to perform time-constrained fact-gathering tasks given documents and:
  • Human-generated summaries,
  • Automatically-generated summaries,
  • No summaries.
  • The speed and correctness of this task constitutes an extrinsic evaluation.
ROUGE

• A commonly used automatic intrinsic evaluation in summarization is ROUGE (Recall-Oriented Understudy for Gisting Evaluation).

• ROUGE is named after and based upon the BLEU metric we saw in statistical machine translation.

• ROUGE automatically scores a machine-generated candidate summary by measuring the degree of its $n$-gram overlap with human-generated summaries (references).
ROUGE-2

- ROUGE-2 fixes the length of $n$-gram overlap at $n = 2$.
- Given $\text{Count}_{\text{match}}(\text{bigram})$ that counts the number of distinct bigrams that occur in both the candidate summary and a given reference $S$ from among all references, $\text{RefSumm}$,

$$\text{ROUGE2} = \frac{\sum_{S \in \{\text{RefSumm}\}} \sum_{\text{bigram} \in S} \text{Count}_{\text{match}}(\text{bigram})}{\sum_{S \in \{\text{RefSumm}\}} \sum_{\text{bigram} \in S} \text{Count}(\text{bigram})}$$

- ROUGE-1 is identical, except it counts unigrams.
ROUGE-2 example

• **Candidate**: An egg falls off a wall.

\[
ROUGE2 = \frac{\sum_{S \in \{RefSumm\}} \sum_{\text{bigram} \in S} \text{Count}_{match}(\text{bigram})}{\sum_{S \in \{RefSumm\}} \sum_{\text{bigram} \in S} \text{Count}(\text{bigram})}
\]

\[
ROUGE2 = \frac{2 + 1 + 0}{8 + 7 + 5} = \frac{3}{20}
\]

Don’t sit on a wall if you’re an egg.

Horses fail to perform surgery upon an egg.

Humpty Dumpty had a great fall.

Ref 1

Ref 2

Ref 3
Aspects of ROUGE

• ROUGE is measured relative to the number of $n$-grams in the references, whereas BLEU was measured relative to the number of $n$-grams in the candidate.
  • ROUGE is based on a desire to cover the same content as in human summaries.

• Unfortunately, human summarizers often disagree about which sentences to include in a summary.
  • Overlap between human summaries can be very low.
  • Although it is a useful baseline, ROUGE is often supplemented with other assessment techniques.
Summarizing summarization

• **Extractive** summarizers produce summaries by selecting important/representative sentences from a text.

• The relevance of sentences can be determined by:
  - **Position:** for instance, news articles tend to begin with their most relevant sentences.
  - **Cue words:** words that indicate relevance such as ‘crucially’ can be determined automatically
  - **Cohesion:** Sentences that contain many strong lexical chains or many co-reference chains ought to be similar to the rest of the document.
Miscellany

• Reading (optional):  Jurafsky & Martin, sections 23.3, 23.4, 23.5, 23.7

• Some of these slides are based on those of Eduard Hovy, Daniel Marcu, and Gerald Penn.