GOFAISUM: A Symbolic Summarizer for DUC

Fabrizio Gotti, Guy Lapalme
RALI-DIRO
Université de Montréal
P.O. Box 6128, Succ. Centre-Ville
Montréal, Québec
Canada, H3C 3J7

Luka Nerima, Eric Wehrli
Laboratoire d'Analyse et de
Technologie du Langage
Université de Genève
2, rue de Candolle
CH-1211 Genève 4, Switzerland

{gottif,lapalme}@iro.umontreal.ca {Luka.Nerima,Eric.Wehrli}@lettres.unige.ch

Abstract

This article presents GOFAISUM, a topic-answering and summarizing system developed for the main task of DUC 2007 by the Université de Montréal and the Université de Genève. We chose to use an all-symbolic approach, the only source of linguistic knowledge being FIPS, a multilingual syntactic parser. We further attempted to innovate by using XML and XSLT to both represent FIPS's analysis trees and to manipulate them to create summaries. We relied on $tf{\cdot}idf$-like scores to ensure relevance to the topic and on syntactic pruning to achieve conciseness and grammaticality. NIST evaluation metrics show that our system performs well with respect to summary responsiveness and linguistic quality, but could be improved to reduce redundancy.

1 Introduction

The main task for DUC 2007 is a two-fold process: answering a complex, "real-world" question given a cluster of 25 documents (news articles) related to the topic and condense the resulting material to produce a 250 word-or-less summary. In previous years, most competing systems considered the problem as an excerpt retrieval task, attempting to find sentence-like text units most likely to answer each question (Dang, 2005). Usually, where the teams differ the most is in the way they compute each sentence's relevance to the question and in the resources they exploit. These resources may be linguistic tools (e.g. Wordnet, Google) and mathematical toolkits (EM, SVM, lexical chains, etc.).

This year, the RALI proposed GOFAISUM, a question-answering and summarization system built around passage retrievals like most of its predecessors, but whose main originality is in its implementation and its use of a syntactic parser for English.

Indeed, we decided to "go back" to the roots of NLP and use a "good old-fashioned artificial intelligence" approach for summarization (hence the acronym GOFAISUM). In order to better pinpoint the strengths and limitations of this strategy, we deliberately chose not to use linguistic resources other than that of a symbolic English parser. No Wordnet, no gazetteer, no Wikipedia information have been used. It is not that we consider them useless for this task: we merely want to limit the number of parameters to study. We did use the ROUGE package to tune our system, but not to augment our system with the resources embedded in the package.

As will be shown in Section 2.1, FIPS incorporates all the handcrafted rules and corpus-derived linguistic knowledge we hope will be sufficient to tackle the task at hand.

With this symbolic approach in mind, we decided to rely entirely on XML for both data representation and manipulation. XML is the ideal language for the task, because it considers information as trees and allows their easy manipulation.

Our system starts by submitting each text source (article cluster and topic) to FIPS. Each analysis is scored by relevance to the topic (using $tf \cdot idf$-like scores) and by other information-content metrics.

The highest scored sentences are then selected, filtered, and post-processed to produce a summary, up to 250 words long. All the computations performed by GOFAISUM are done by tree manipulations, implemented as XSL stylesheets.

Section 2 discusses FIPS and XML, the tools at the heart of our system. Section 3 describes in more detail the approach we developed. In Section 4, we present and analyze the results obtained from the evaluations carried out by NIST. The last two sections highlight possible future work and conclude the paper.

2 Resources Used

We limited the resources we used for DUC to FIPS, a syntactic parser as a source of linguistic information and to XML-derived technologies to process this information.

2.1 FIPS: a syntactic parser

FIPS (Wehrli, 2007) is a robust multilingual symbolic parser based on generative grammar. It is the cornerstone of a long-term project at the LATL (Language Technology Laboratory) at the Université de Genève and is used in several NLP applications: text-to-speech synthesis, automatic collocation extraction, translation of words in context, etc.[1] Although we used the English configuration of FIPS for DUC 2007, FIPS also parses French, German, Italian, Spanish and Greek.

2.1.1 The principles of FIPS

The syntactic structures built by FIPS are all of the same pattern, that is `[XP L X R]` where `XP` stands for the label of the structure, `L` stands for the possibly empty list of left constituents, `X` for the possibly empty head of phrase and `R` for the possibly empty list of right constituents. The possible categories for `X` are the usual parts of speech. The overall resulting structure is a tree where the node labels are the `XP`s and the leaves, the `X`s. Figure 1 shows the parse tree constructed by FIPS for a sentence extracted from the DUC 2007 test data for the topic D0707B, "Turkey and the European Union".

The parser makes use of 3 fundamental mechanisms: projection, merge and move.

[1] See `http://www.latl.unige.ch/english/latl_e.html` for a complete list of references.

The **projection** mechanism assigns a fully developed structure to each input word, based on its category and other inherent properties. Thus, a common noun is directly projected to an `NP` structure (with the noun as its head), a preposition to a `PP` structure, etc. The occurrence of a tensed verb triggers a more elaborate projection: a whole `TP-VP` structure is assigned. For instance, Figure 1 shows that the modal *should* occurs in the `T` position while *accept* is projected to a `VP`.

The **merge** mechanism combines two adjacent constituents, `A` and `B`, either by attaching constituent `A` as a left constituent of `B`, or by attaching `B` as a right constituent of any active node of `A`. In Figure 1, the complementizer phrase *that the European Union...* is right attached to the `VP` *said* while the `DP` *The U.S. government* is left-attached to the `TP` *said....* Merge operations are constrained by mostly language-specific conditions which can be described by means of syntactic rules. For instance, in English, a rule states that a `DP` can be left-attached to a `TP` if (1) the `DP` agrees in number and person with the `TP` and (2) the `DP` can be interpreted as the subject of the `TP`. For English, FIPS has about 30 rules for left attachment and 90 rules for right attachment.

In order to handle extraposition, the **move** operation creates chains by linking each of the extraposed elements to an abstract (empty) element in the canonical position. For instance, when parsing the sentence "Whom did they invite?", FIPS creates the following chains: "Whom$_i$ did$_j$ they e$_j$ invite e$_i$?"

2.1.2 Lexical resources for FIPS

The parser uses (1) a word lexicon associating with each word all its inflected forms along with their morphological descriptions and (2) a lexeme lexicon, containing the possible syntactic configurations for the word (about 55,000 entries for English). FIPS handles unknown words by guessing their lexical category according to their position in the sentence and the applicable syntactic rules.

2.1.3 Implementation

As the syntactic structures produced by FIPS are trees, the transformation into XML trees was rather straightforward. We also added to the output the base representation of each word (or lemma).

The parser uses a mixed bottom-up and top-down

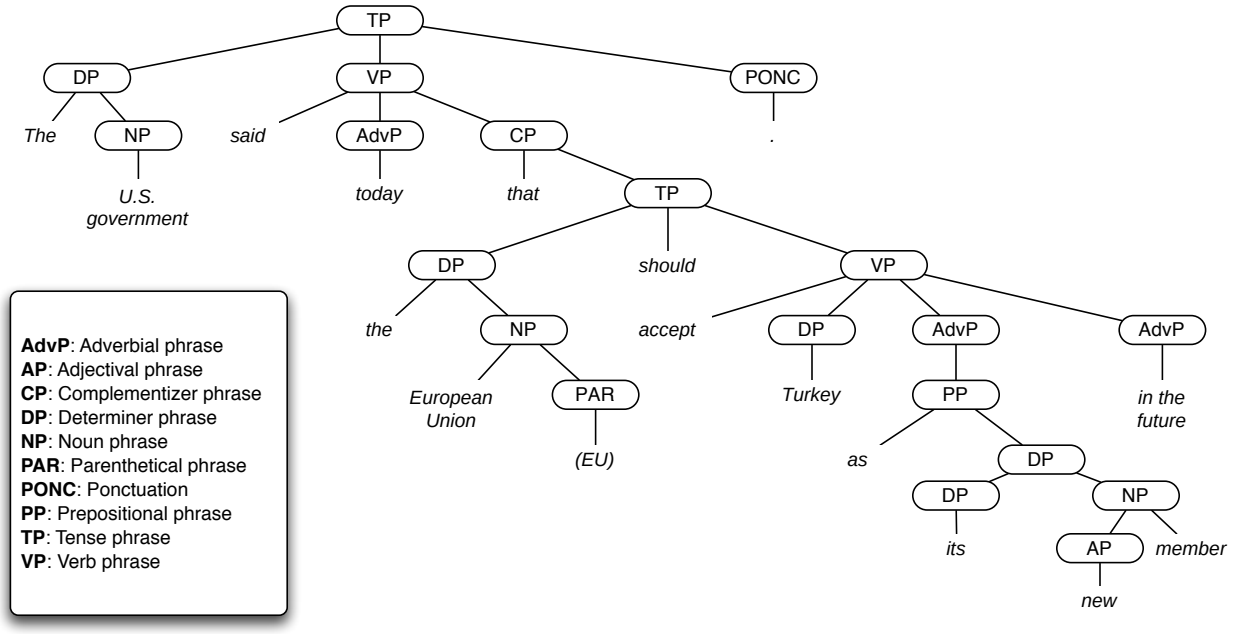

Figure 1: FIPS parse tree for the sentence *The U.S. government said today that the European Union (EU) should accept Turkey as its new member in the future.* Nodes indicating lemmas for each node are omitted for clarity. The corresponding topic question was "What positive and negative developments have there been in Turkey's efforts to become a formal member of the European Union?".

filtering algorithm with parallel treatment of alternatives, as well as heuristics to rank alternatives. When the number of candidate analyses exceeds a threshold (35 for this work), the parser prunes the lowest scored alternatives in excess.

2.2 XML and XSLT

XML (eXtensible Markup Language) is a formalism designed to describe and share structured information through text-based trees. Some of the information in an XML document describes its structures (these are markup elements) and the rest constitutes the data per se. It is becoming the standard for information exchange between computer programs and was used in this work to describe the output of FIPS as well as all other intermediate outputs.

According to (Kay, 2007), XSLT 2.0 is a language for transforming the structure and content of an XML document. It stands for eXtensible Stylesheet Language: Transformations. It's a declarative language used to build *stylesheets*, consisting of template rules each describing how a particular element of the XML document it is fed should be processed. XSLT stylesheets contain XPATH 2.0 expressions. XPATH acts a sublanguage of XSLT, primarily to make it easy to access data in XML documents (Clark and DeRose, 1999), although it can do much more: it handles arithmetic, regular expressions, control structures, etc.

For instance, in the XML document whose schematic representation is shown in Figure 1, to prune out all subtrees rooted in a parenthetical node (`PAR` in the schema), one only has to declare a template that matches `PAR` nodes in the input tree and outputs nothing. This way, all other source nodes are output unchanged, while the parenthetical node and its whole subtree is deleted. This is the kind of elegant and powerful strategies from which we seek to benefit in GOFAISUM.

3 Our Approach

GOFAISUM firsts parses the English text of any given cluster/topic pair with FIPS, after some light preprocessing. FIPS yields an XML (tree-like) representation of the analysis, which is manipulated and given a score with XSLT 2.0 stylesheets. Figure 2 displays the various steps involved in the workflow.

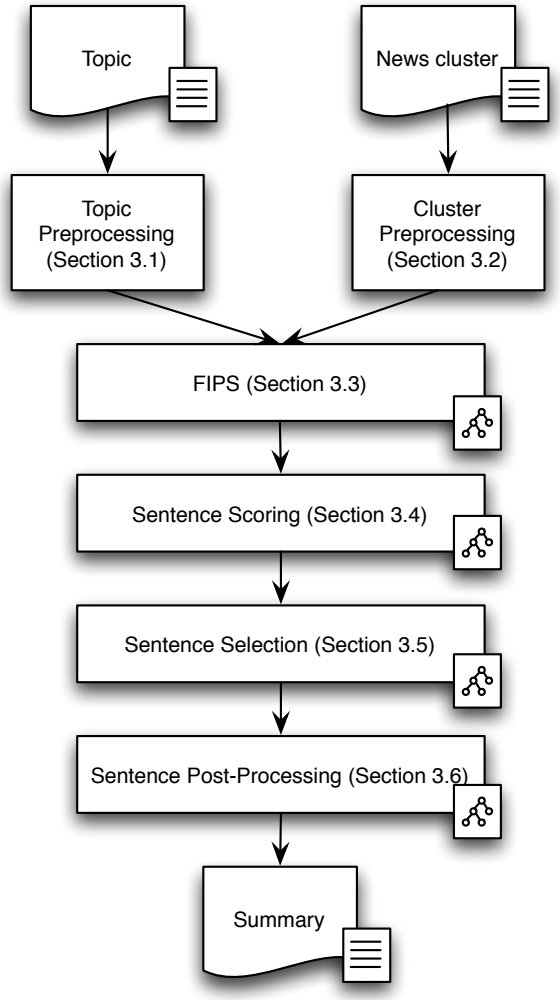

Figure 2: Workflow used by GOFAISUM for a single topic. The section labels refer to the section in this paper where the labeled operation is discussed.

We discuss these steps in the present section.

3.1 Topic processing

Processing the topic is straightforward, since the source file is already in XML format: for each topic, we create a smaller XML file containing only the `<title>` and `<narr>` tag text, for the given topic.

3.2 Cluster processing

We start by converting each article file from the SGML format to XML. Then we filter out everything but the article text, contained within the `<text>` tags, ignoring the paragraph marks. We maintain a link to the original article date in order to resolve relative temporal expressions later on (see Section 3.6.1). We exclude the headline of the article, because it rarely constitutes a valid excerpt in the final summary: it is often an incomplete sentence (without a verb), designed primarily to capture the reader's attention rather than to convey information. We furthermore remove the datelines from the articles and simplify quotations marks.

Table 1: Values and weights of the 5 different scores used to evaluate the relevance of the example sentence shown in Figure 1. The column "Section" indicates which section of the paper discusses each score.

Score name	Section	Value	Weight
Word $tf \cdot idf$	3.4.1	0.30	15%
Lemma $tf \cdot idf$	3.4.2	0.35	50%
Lemma $tf \cdot idf$ (+ depth)	3.4.3	0.26	5%
Sentence weight	3.4.4	0.29	20%
Absolute position	3.4.5	1.0	10%
Final score	3.4.6	0.39	100%

3.3 FIPS

All the text isolated from steps 3.1 and 3.2 is submitted for analysis to FIPS, described in Section 2.1 From this point on, we only rely on the analysis tree produced by FIPS to evaluate the relevance of each sentence/analysis to the topic.

3.4 Sentence scoring

To evaluate the relevance score of each sentence analyzed by FIPS, we used a combination of five different scores, each one based on the information derived from the syntactic analysis trees. The final score for every sentence is a linear combination of these five scores, each one ranging from 0 to 1.

Table 1 shows the different components of the final score for the running example, from Figure 1.

3.4.1 Word-based $tf \cdot idf$ similarity score

Since FIPS performs word segmentation as part of its analysis, we implemented a $tf \cdot idf$ similarity score based on the word vectors delimited by FIPS. We use the cosine similarity between two vectors $\vec{s}$ and $\vec{q}$, where s_i and q_i are the weights of the word i

in, respectively, the sentence s and the corresponding topic q. The similarity between s and q is then simply $\vec{s} \cdot \vec{q} / \|\vec{s}\| \|\vec{q}\|$.

The weight $p(w)$ of a word w in a sentence (or topic) is $p(w) = tf(w) \cdot idf(w)$, where $tf(w)$ is the term frequency in the sentence and $idf(w)$ is the inverse document frequency. The latter was looked up in a table (in XML format) built offline from the collection of all sentences from all article clusters and topics in the DUC 2007 test material. Stopwords were filtered out from the word vectors.

It is noteworthy that a "document" here is in fact a sentence, because it is the primary retrieval unit for this task (assembling relevant sentences to form a summary).

A manual inspection of the topics led us to conclude that the topic title is of the utmost importance: it almost always include the keywords that a good candidate sentence must contain in a summary. For this reason, the weight of topic words is boosted by a factor of 2 (an arbitrary value) before calculating the cosine similarity measure.

3.4.2 Lemma-based $tf \cdot idf$ similarity score

During each analysis of an English sentence, FIPS identified the lemmas for each noun or noun phrase. We exploited this information by computing a lemma-based $tf \cdot idf$ similarity score. We proceeded as explained in subsection 3.4.1, but considered the lemmas here, instead of the words. Extracting the lemmas was simple and elegant in XSLT: we only had to explore the content of the <LEX> nodes within the analysis tree. Another inverse document frequency table had to be built for lemmas only.

3.4.3 Lemma-based $tf \cdot idf$ score, with node depth

One advantage of considering analysis trees rather than mere character strings for a given candidate sentence is that we can augment the weight $p(w)$ of a word or lemma with its depth in the analysis tree. Indeed, intuitively, the higher a word is in the analysis tree, the more dependent words it is likely to have and the greater its imporance must be. We therefore modified the $tf \cdot idf$ score for lemmas presented in subsection 3.4.2 to account for the lemma's depth. The weight $p_d(w)$ of a lemma w becomes:

$$p_d(w) = tf(w) \cdot idf(w) \cdot \frac{1}{min\_depth(w)}$$

where $min\_depth(w)$ is the depth of the highest-positioned occurrence of lemma w.

3.4.4 Sentence weight

We also used the sentence weight, the sum of the weights of each of the n words it contains (i.e. $\sum_{i=1}^{n} p(w_i)$) as an element of the final score. This score favors information-rich sentences, regardless of their relevance to the topic. This score is normalized, to ensure that the final score remain between 0 and 1. We therefore divided each non-normalized sentence weight by the highest non-normalized sentence weight for the sentences of the same cluster.

3.4.5 Absolute sentence position

Typically, the sentences near the beginning of an article are more concise. We took this into account with an absolute position score, equal to $1/pos_s$, where $pos_s = 1 \ldots S$ is the position of the sentence s relative to the beginning of an article containing S sentences. We did not have time to implement a relative position score, one that would indicate the relative position of a sentence within its paragraph. This would have been possible, thanks to the <P> tags or the empty lines delimiting paragraphs.

3.4.6 Final Score

Each sentence in a cluster is given a score that is a linear combination of the 5 scores described previously. The weights used to combine these scores were determined by trial and error, on the DUC 2006 data set, available to us because we participated in DUC last year as well. We sought to optimize the scores NIST will use to evaluate summaries this year, i.e. ROUGE-2, ROUGE-SU4 (with stemming and keeping stopwords) and overlap in Basic Elements (BE). We used the ROUGE & BE package available from ISI[2] to do so. Qualitative visual inspections were also performed to ensure quality.

3.5 Sentence selection

Sentences with the highest scores are used to build the final summary. Regardless of their score, however, we dismiss sentences if they meet *any* of the

[2] http://haydn.isi.edu/BE/

following conditions. After each criteria, we put within parentheses the percentage of the original 26,452 sentences that are filtered out by the step.

1. Sentences that cannot be completely analyzed by FIPS, because they produce partial analyses that are difficult to manipulate, and whose content cannot be trusted entirely. (29%)

2. Duplicate sentences, excerpted from two different articles describing the same event. (4%)

3. Sentences with no verb, as they rarely convey interesting information and would hurt the overall grammaticality of the summary. (5%)

4. Sentences containing the "I" pronoun, which are usually opinions or feelings, rarely adding factual information. Naturally, FIPS's analysis allows an easy distinction between the "I" in, say, Voyager I, and the actual pronoun. (3%)

5. Sentences ending with a colon or a question mark, which usually introduce an element of information, rather than discuss a point. (2%)

6. Sentences whose words are all in uppercase, or that count less than 5 words. (4%)

Note that sentences meeting criteria 3 through 6 are easily detectable when exploring their analysis tree with XSLT and XPATH. We ultimately filter out 47% of the original sentences.

3.6 Sentence post-processing

3.6.1 Referential clarity

Assembling sentences coming from multiple sources creates referential clarity problems. Not only does it hurt the probability that an important sentence is retrieved when it uses pronouns instead of the actual referred entities, but the summary produced by assembling such sentences is unclear.

To try to solve this, we filter out some ambiguous referential expressions. For instance, a sentence like *"Climate is changing," he said.* becomes *Climate is changing.* This suppresses the ambiguous *he*, while retaining the sentence's information (presented differently, we must admit) and its grammaticality.

Ambiguous temporal references are also a problem: a word like *Tuesday* has no meaning when taken out of context. We can resolve relative dates using the date of the container article.

References to the present day in a sentence are replaced with the full date, and references to days of the week are replaced by the month and year only. In Figure 1, the adverb *today* within the tags `AdvP` is pruned and replaced with the date of the article. To avoid repetitions in the summary, we avoided repeating a given absolute temporal reference, preferring to remove it altogether.

3.6.2 Sentence compression

Sentence compression strives to identify and remove non-essential parts of the text while maintaining grammaticality and fluency, something relatively easily achieved when the sentence is represented by a syntactic analysis tree, as in our case. We need only perform syntactic pruning, removing the subtrees that are non-essential (see (Gagnon and Sylva, 2005) and (Chali and Kolla, 2005) for instance).

We pruned out subtrees whose roots are `PAR` elements, which indicates a parenthetical expression. In Figure 1, this removes the unnecessary `(EU)`, the acronym used for the European Union. Unfortunately, most parenthetical expressions could not be deleted without hurting the meaning of the compressed sentence. For instance, the sentence *The SLORC – State Law and Order Restoration Council – which...* cannot be understood without the parenthetical expression within double dashes. So we resorted to prune parenthetical expressions consisting of acronyms in uppercase and other easily identifiable redundant expressions.

3.7 Implementation

Thanks to the highly expressive power of the XSL and XPATH languages, the complete manipulation code which performs the tasks described in sections 3.4 through 3.6 is contained by a single 750-line XSL stylesheet.

We conducted our experiments on 3 GHz desktop computers. The time required to process a single cluster of articles and its associated topic question is presented in Table 2. It takes about 8 minutes to go from the raw material to a summary ready to submit. FIPS yielded complete analyses for 71% of the article sentences and for 93% of the (shorter) topic sentences.

Table 2: Average wall clock times for each step in processing a *single* article cluster (25 articles) and its associated topic question.

Step	Section(s)	Time (min)
Preprocessing	3.1, 3.2	0.1
FIPS analysis	3.3	4.0
Sentence scoring	3.4	4.0
Selection & Post-processing	3.5, 3.6	0.1
Total time	—	8.2

4 Results

NIST organizers provided every team with a thorough evaluation of the quality of its summaries. Figure 3 presents four evaluation metrics for automatic and model (human) summaries. The figure clearly shows how human summaries outperform automatic ones on all metrics, something that was expected.

The **content** is a human evaluation measuring the responsiveness of the summary, i.e. how it answers the topic question. GOFAISUM is in 11^{th} place (out of 30 participants) with respect to this metric.

The **Linguistic quality** is an average of five distinct linguistic quality metrics (not shown in Figure): grammaticality, non-redundancy, referential clarity, focus and structure and coherence. GOFAISUM performs quite well on this test, with a 5^{th} place. This good performance is essentially attributable to the grammaticality and referential clarity scores. Indeed, a system that selects and outputs sentences from human-made articles is bound to fare well on grammaticality. This also shows that the postprocessing and pruning we perform on sentence is sound. Referential clarity clearly benefits from the simple, yet efficient strategies described in Section 3.6.1.

Our performance is however handicapped by a low score on non-redundancy (23^{rd} rank). A manual inspection of our summaries shows that a non-negligible number of selected sentences repeat the same ideas as previous ones in the same summary. This should be improved in the next iteration of GOFAISUM. Once again, this could be performed with a $tf \cdot idf$-like score measuring similarities between candidate sentences for the same summary.

Overlap in **Basic Elements** is also used, through the `ROUGE-1.5.5` package, to automatically evaluate the summaries. GOFAISUM obtains a 17^{th} rank for this metric, but, when the 95% confidence interval (the error bars on the chart) is taken into account, GOFAISUM actually performs worse than only 9 other teams, which puts it in the 10^{th} position. It is noteworthy that ROUGE-2 and ROUGE-SU4 scores are very similar to the overlap in Basic Elements (all Pearson correlation coefficients are approximately 0.97). We therefore only displayed one of these three scores in Figure 3.

Finally, a pyramid evaluation was performed by human annotators. GOFAISUM placed 8^{th} out of 13 annotated systems.

5 Future Work

Improvements for a future participation in DUC are the following.

Sentence parsing by FIPS could be improved by augmenting the parser with dedicated lexicons (e.g. one for proper nouns). This would translate into a higher proportion of complete analyses and more accurate parse trees.

Moreover, this step would be the ideal moment to perform anaphora resolution. Indeed, this problem appears to be one of the most challenging tasks, both for question-answering and summarizing. We think that a better solution to this difficulty could significantly improve GOFAISUM. A better anaphora-resolution module would not only attempt to resolve most pronouns, but also complete the work initiated here on temporal expressions.

Sentence selection would benefit from the development of additional sentence scores besides those presented in Section 3.4. The weights for these new scores could be trained using more sophisticated strategies, for instance the Nelder and Meade's downhill simplex method (Nelder and Meade, 1964) or a perceptron, to stay within the AI field.

Finally, **sentence post-processing** could integrate a strategy to reduce redundancy within the same summary. We would also like to refine our syntactic pruning techniques to eliminate some subordinate clauses, adjectival and adverbial modifiers, in a principled way.

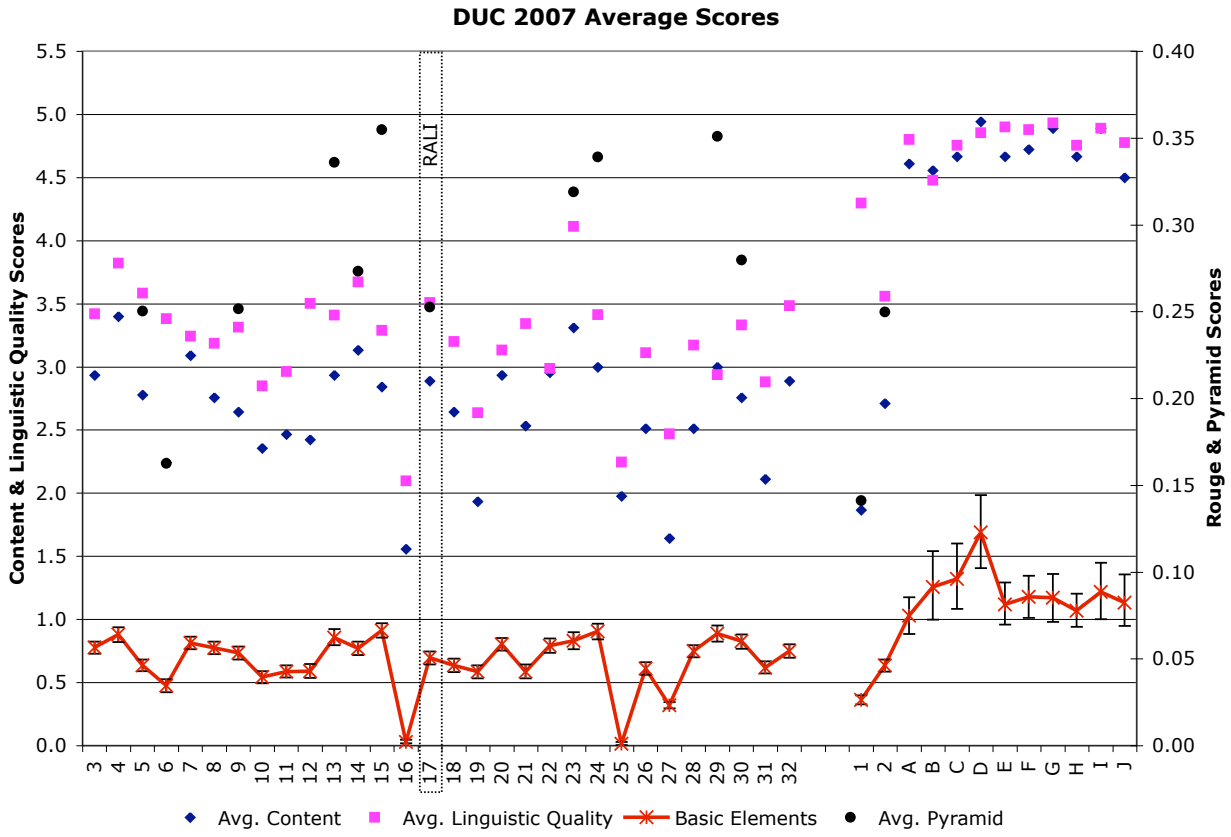

Figure 3: Average evaluation scores for DUC 2007. Scores are provided for all teams (labeled 3 through 32), for the two baselines (labeled 1 and 2) and the ten manual summaries (A through J). Content and linguistic quality scores should be read off the y-axis on the left side while Basic Elements and pyramid scores should be read using the right-hand side y-axis. The RALI team has ID 17, as shown by the vertical box in the chart. Error bars displayed for the Basic Elements scores correspond to the 95% confidence intervals.

6 Conclusion

We presented a simple, yet powerful and promising, approach to topic-answering and summarizing tasks. We went back to the roots of AI by relying entirely on a syntactical parser for linguistic knowledge and on a symbolic approach for data analysis and manipulation. We hope to have shown that modern tools and reliable syntactic parsers open new possibilities for principled summarizations.

References

Yllias Chali and Maheedhar Kolla. 2005. Producing headline summaries for newspaper articles. In *Canadian Conference on AI*, volume 3501 of *Lecture Notes in Computer Science*, pages 422–426. Springer-Verlag.

J. Clark and S. DeRose. 1999. XML Path Language (XPath) Version 1.0. W3C Recommendation, `www.w3.org/TR/xpath`, November.

H. Dang. 2005. Overview of DUC 2005. In *Document Understanding Conference 2005 Workshop*.

Michel Gagnon and Lyne Da Sylva. 2005. Text summarization by sentence extraction and syntactic pruning. In *Proceedings of Computational Linguistics in the North East (CliNE05)*.

Michael Kay. 2007. XSL Transformations (XSLT) Version 2.0. W3C Recommendation, `www.w3.org/TR/xslt20/`, January.

J.A Nelder and R. Meade. 1964. A simplex method for function minimization. *The Computer Journal*, 7:308–313.

Eric Wehrli. 2007. Fips, a "Deep" Linguistic Multilingual Parser. In *"Deep Linguistic Processing" workshop, ACL 2007*, Prague, June.