TeX(t)-Condensation

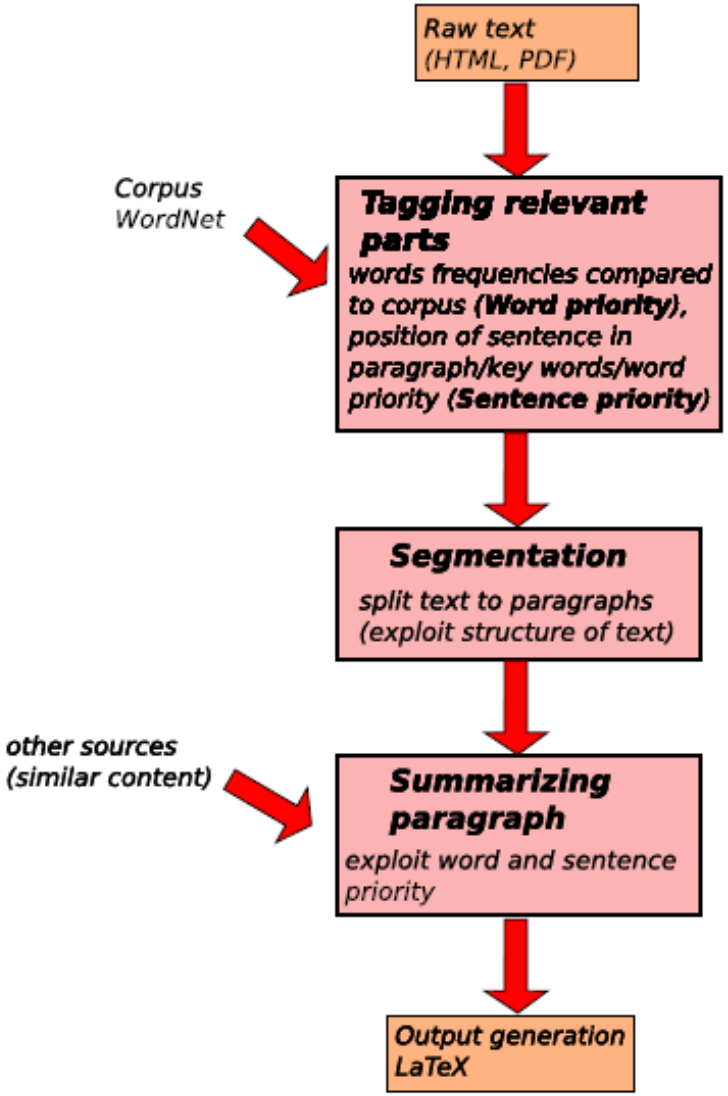
Eduardo Aponte, Kimon Batoulis,
Stefan Depeweg, Steffen Vogel

Institute of Cognitive Science
University of Osnabrueck

July 9, 2010

Outline

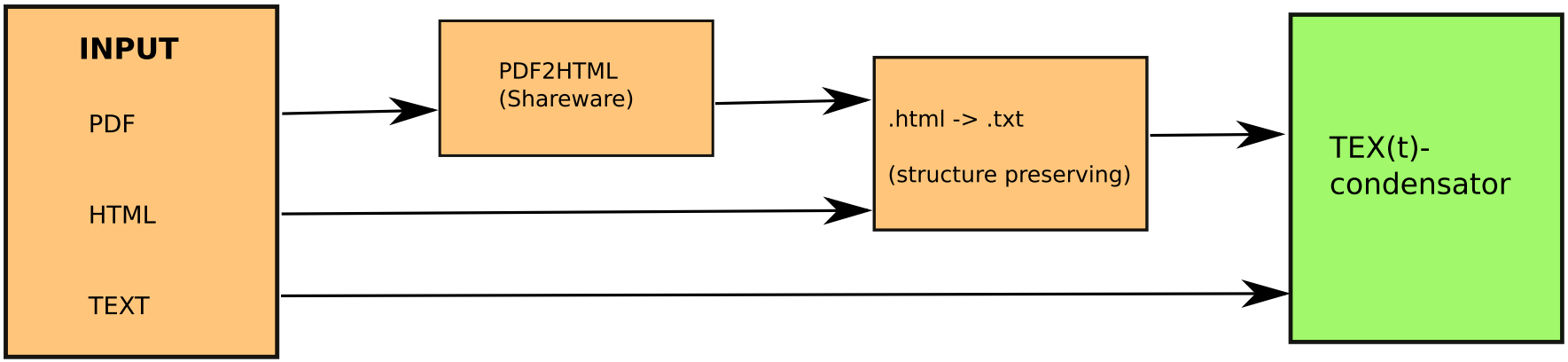
Overview



TeX(t)-Condensator

Steffen Vogel
Kimon Batoulis
Eduardo Aponte
Stefan Depeweg

Input Data


INPUT
PDF
HTML
TEXT
PDF2HTML
(Shareware)
.html -> .txt
(structure preserving)
TEX(t)-
condensator

Calculating Word Priority

- statistical analysis
  - compute frequency of open class words in text
  - compare to frequency of words in corpora (UKWAC)
- morphological analysis
  - stemming and grouping (runs $\rightarrow$ run, ran $\rightarrow$ run)
- semantic analysis
  - find synonyms/hypernyms of nouns, verbs and adjectives/adverbs respectively

Statistical Analysis

- frequently occuring words seem to be important for the meaning of a text

$$freq_{text}(word) = \frac{count(word)}{length(text)}$$

- to rule out frequent words like *like, is* etc., we compare the word frequencies to general corpora

$$freq(word) = \frac{freq_{text}(word)}{freq_{corpus}(word)}$$

- By that we also make words important that are frequent in the text but rare in a corpus (e.g action potential, threshold, Sodium for Neuroscience)

Morphological Analysis

- all words forms for a given lexeme should be handled equally
- use the *Lancaster Stemmer* and a white list to find word stems

$$freq_{stem}(word) = \sum_{\text{wordforms of } word} freq(wordform)$$

$$freq(wordform) = freq_{stem}(word)\ ,\ \forall wordforms(word)$$

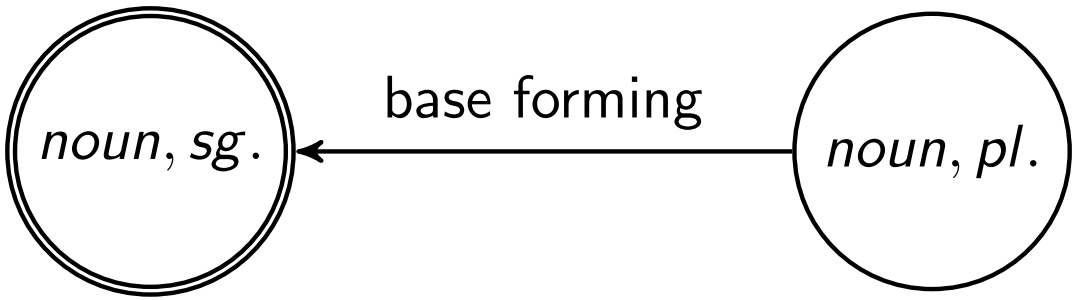
- **Example:**

$$freq_{stem}(run) = freq(run) + freq(runs) + freq(running) + freq(ran)$$

$$freq(run) = freq(runs) = freq(running) = freq(ran) = freq_{stem}(run)$$

Semantic Analysis—Synonyms

- words that are semantically related should have the same frequency
- convert all inflected words to their base forms
- search for synonyms among them using *WordNet*
  (i) nouns
  (ii) verbs
  (iii) adjectives/adverbs
- **Example:**



Semantic Analysis—Hypernyms

- hyponyms increase the values of their hypernyms

$$freq(hypernym) = freq(hypernym) + 0.5 * \sum_{hyponyms} freq(hyponym)$$

- **Example:**

$$freq(car) = freq(car) + 0.5*\mathbf{(}freq(bus) + freq(ambulance) + \ldots\mathbf{)}$$

External Software

TreeTagger, by Helmut Schmid
`www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html`

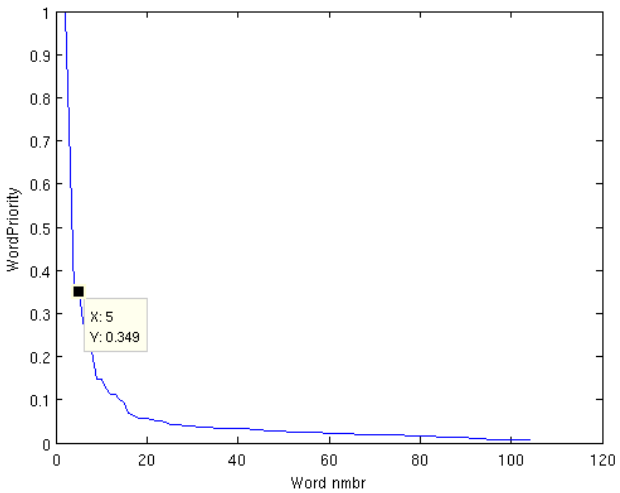
Lancaster Stemmer, NLTK
`nltk.googlecode.com/svn/trunk/doc/api/nltk.stem.lancaster.LancasterStemmer-class.html`

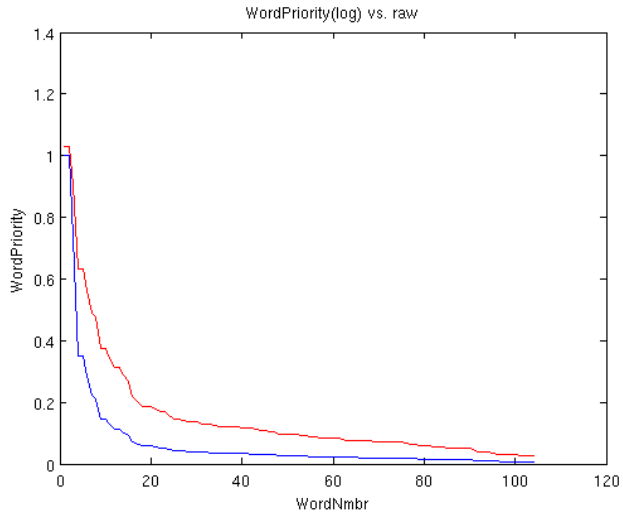
UKWAC, UK Web Archiving Consortium
`wacky.sslmit.unibo.it/doku.php?id=frequency_lists`
1.9G words , 5.9M word forms (including tons of typos), 0.4M word forms (after removing most typos and hardly any real words

Performance: Priority-Space

Problem: Very few words get very high priority: 5 out of 100 words get Priority > 0.35 (max is 1)

Solution: Adjust parameters (e.g reduce influence of synomyns to word priority) and take log over distribution





Sentence Priority

The importance of a sentence is calculated in three steps:

$$if\ \ numberOfWordsInSentence > 14 :$$
$$numberOfWordsInSentence = numberOfWordsInSentence + pValue$$

$$meanWordPriority = \sum priority(word)/numberOfWordsinSentence$$

$$SentencePriority = meanWordPriority * (1 + (keyWordImportance * numberofKeywordsInSentence))$$

Hence, the priority depends on the average WordPriority, the occurence of KeyWords and the penalty for long sentences!

Structure Extraction

From .txt:

- Use newlines to segment text into paragraphs
- Scan text for headlines (Introduction, Conclusion etc.) use this also for Outline-Slide
- Calculate number of slides needed for paragraph by length of paragraph

From .htm

- go through every page of .htm;
- Check for headlines and paragraphs
- Tag positions and transform everything to .txt with tagged headlines and positions

ToDo-List

- further parameter-tuning (both for WordPriority as well as Sentence Priority)
- improved Evaluation methods (up to now: just our impression)
- cross-check corpora
- Sentence-shortening(?)
- solve various Unicode/UTF-8/ASCII problems for data input

Expected Data & Propose

First point: The overall quality of the produced slides is rather poor compared with a human-made summary :/

Reasons:

- No real summary, it just picks sentences that seem to be important
- Measuring word-priority by freq(text)/freq(corpus) is not always good
- The text may not be well structured

However:

- The Condensator in many cases finds important sentences (compared to other sentences)
- For preparing Slides and as a heuristic for important sections in a large text it is quite useful

Demonstration

Task:
1. summarize a short text about Economy: Data input .txt, Difficulty of the text: medium
2. summarize a philosophical essay from Paul Churchland: Data input: .htm Difficulty of the text: hard

Questions to you:

- Does the Condensator recognizes the structure of the text (easy)
- 2. Does the Slides help you to get the concepts out of the text without reading it? (medium)
- 3. Is it possible to hold a presentation just with the generated slides? (ok, just kidding..)