# Extended tokenizer for Polish

Tomasz Bartosiak
Konrad Gołuchowski
Katarzyna Krasnowska

13 marca 2014

## 1 Method description

The tokenization (augmented with simple tagging with token type) implemented in our program consists of 3 main steps:

### 1.1 Basic splitting

At this stage, the most basic splitting operations are performed on the input text. Each sentence is split on spaces. Additionaly, when the resulting tokens begin or end with interpunction characters, the leading and trailing iterpunction is stripped into separate tokens. This allows, e.g., for separating parentheses, colons and semicolons from neighbouring tokens. An exception from this is the treatment of a dot preceded by a non-interpunction character. Such a dot is kept within the same token for later processing of abbreviations.

### 1.2 Filters cascade

In the second stage, the most of the tagging is performed. A series of token-type filters is defined together with an order in which they are applied, forming what we called a *filters cascade*. Each filter may either:

- Recognise a token as belonging to one of the defined types and tag it. In this case, the tagging is done for the given token and the cascade is run on the next token.
- Recognise a token as a concatenation of proper tokens, split them, tag some of them if possible and leave the rest to be recursively passed through the cascade.
- Fail to recognise the token: in this case, the next filter from the cascade is applied.

The simplest filters use regular expressions. In that way, e.g., roman/arabic numerals, e-mail or www addresses can be tagged. A little more complicated ones may split the token based on a regular expression and assign all resulting parts a tag, as is done, e.g., in the case of arabic numerals followed by a dot.

The most complex filter is the one used for recognising abbreviations. It makes use of some predefined list of valid abbreviations of different type, included in `.txt` files (see the files description at the end of this document). *[... more details about abbreviations...]*

### 1.3 Date parsing

At this stage,

## 2 Authors contribution

**Tomasz Bartosiak:** handling XML format of input/output files, abbreviations.

**Konrad Gołuchowski:** filters cascade stage: project and particular filters.

**Katarzyna Krasnowska:** filters cascade stage: particular filters, dates.

Besides the above, each author provided some testing input files, repeatedly tested the method against those files and fixed or reported detected problems.

## 3 Files description

Python source code:

- `main.py` – the main program file, contains high-level code for handling input/output files and code for first-stage tokenization.
- `token.py` – contains filters used in the filters cascade stage.
- `date.py` – contains code for handling dates.
- `tags.py` – tag names defined as constants for convenience.
- `ext_tokenizer_xml_parsing.py` – contains code for parsing and printing XML files.

Other files used by the program:

- `dots_sorted.txt` – list of abbreviations ending with dot.
- `multi_part_dot_abbr.txt` – list of multi-part abbreviations ending with dot.
- `unit_names.txt` – list of abbreviations for physical units.
- `unit_prefixes.txt` – list of prefixes for physical units.
- `uninflected.txt` – list of uninflected abbreviations not ending with dot
- `inflect_base.txt` – list of stems for inflected abbreviations.
- `inflect_ending.txt` – list of possible endings for inflected forms of abbreviations.