# Bitext Linguistic Services Overview

### Lexical services (no grammar)

| | |
|---|---|
| Sentence segmentation | Splits text into sentences, according to language-specific punctuation rules.<br>Applicable to all languages.<br>Example: Hello! How are you doing? → Hello! \| How are you doing? |
| Tokenization | Splits a sentence into words, according to language-specific space and punctuation rules.<br>Applicable to most languages (except Chinese, Japanese, Vietnamese, Thai…)<br>Example: How are you doing? → How \| are \| you \| doing \| ? |
| Word segmentation (no-space tokenization) | Split text into words for languages that do not use spaces to separate them.<br>Applicable to Chinese, Japanese, Vietnamese, Thai…<br>Example: 把音量调低一点→ 把 \| 音量 \| 调低 \| 一点 |
| Decompounding | Split compound words/tokens into its individual component words.<br>Applicable to German, Dutch, Norwegian, Swedish, Korean…<br>Example: Rindfleischetikettierung → Rind \| Fleisch \| Etikettierung |
| Lemmatization (ambiguous) | Return the possible roots for a word form<br>Applicable to most languages (except Chinese, Vietnamese, Thai…)<br>Example: running → run |
| POS Tagging (ambiguous) | Return the possible parts of speech (and optionally other attributes) of a word<br>Applicable to all languages<br>Example: run → verb (infinitive), verb (1st person singular, present tense), noun (singular) |
| Inflection | Return all forms of a root word<br>Applicable to most languages (except Chinese, Vietnamese, Thai…)<br>Example: run → run, runs, ran, running |
| Language identification | Detect the language(s) used in each sentence of a longer input text<br>Applicable to all languages<br>Example: Oui! I love Paris → "Oui!" – French, "I love Paris" – English |
| Spell checking | Check if a word is spelled correctly<br>Applicable to all languages<br>Example: excelent → incorrect |
| Spell suggestions | Suggest corrections for incorrectly spelled words<br>Applicable to all languages<br>Example: excelent → excellent |

**Syntactic services (grammar)**

| | |
|---|---|
| Entity extraction | Detect proper names (people, places…) and other special text (phones, URLs…)<br>Applicable to all languages<br>Example: John lives in New York → “John” – person name, “New York” – place |
| Offensive language detection | Detect offensive or vulgar expressions in text<br>Applicable to all languages<br>Example: tell John to f*ck off → “f*ck off” – offensive |
| Anonymization | Remove sensitive or personal information (PII) from text<br>Applicable to all languages<br>Example: My name is John and my account number is 1234567 → My name is XXXX and my account number is XXXX. |
| POS-Tagging (disambiguated) | Return the parts of speech for each word in a sentence<br>Applicable to all languages<br>Example: John runs back home → “John” – proper noun, “runs” – verb, “back” – preposition, “home” - noun |
| Phrase Extraction | Returns the constituents (noun phrases, verb phrases…) of a sentence<br>Applicable to all languages<br>Example: John’s sister was performing in the theatre → “John’s sister” – NP, “was performing” – VP, “in the theatre” – PP |
| Topic-Based Sentiment Analysis | Returns the sentiment and corresponding topic of opinions in text<br>Applicable to all languages<br>Example: I hate my old phone → opinion: “hate” (negative), topic: “my old phone” |
| Categorization | Returns the categories applicable to a text, based on pre-defined rules<br>Applicable to all languages<br>Example: John is feeling great. → HAPPINESS<br>[RULE: feel + great → HAPPINESS]<br>Example: John was weeping like a willow. → SADNESS<br>[RULE: weep + like + willow → SADNESS] |

**Other (low level)**

| | |
|---|---|
| Parsing | Produce a tree with the hierarchical constituent parts of a sentence (words, phrases, clauses…)<br>Applicable to all languages |