Large-scale lexical analysis

Gr. Thurmair, V. Aleksić, Chr. Schwarz
Linguatec
Gottfried-Keller-Str. 12, 81245 Munich
E-mail: g.thurmair@linguatec.de, v.aleksic@linguatec.de, c.schwarz@linguatec.de

Abstract

The following paper presents a lexical analysis component as implemented in the PANACEA project. The goal is to automatically extract lexicon entries from crawled corpora, in an attempt to use corpus-based methods for high-quality linguistic text processing, and to focus on the quality of data without neglecting quantitative aspects.
Lexical analysis has the task to assign linguistic information (like: part of speech, inflectional class, gender, subcategorisation frame, semantic properties etc.) to all parts of the input text. If tokens are ambiguous, lexical analysis must provide all possible sets of annotation for later (syntactic) disambiguation, be it tagging, or full parsing.
The paper presents an approach for assigning part-of-speech tags for German and English to large input corpora (> 50 mio tokens), providing a workflow which takes as input crawled corpora and provides POS-tagged lemmata ready for lexicon integration. Tools include sentence splitting, lexicon lookup, decomposition, and POS defaulting. Evaluation shows that the overall error rate can be brought down to about 2% if language resources are properly designed.
The complete workflow is implemented as a sequence of web services integrated into the PANACEA platform.

Keywords: lexical analysis, morphology, information extraction

1. Introduction[1]

Lexical Analysis is of central importance in computational language processing. The reason is that anything a machine ‚knows' about a string must be coded in a lexicon. Therefore, mapping of strings (tokens) to the lexicon is a central task in language processing.
The following paper presents a lexical analysis component as implemented in the PANACEA project (Bel, 2010); it covers German and English text, and is intended to support the analysis of large corpora. The goal is to use corpus-based methods for high-quality linguistic text processing, and to focus on the quality of data without neglecting quantitative aspects.
In PANACEA, analysis starts from the result of text crawling: Relevant texts are identified in the internet, and prepared for linguistic analysis. Sentence splitting and tokenisation identify the units which can undergo lexical analysis.
Lexical analysis has the task to assign linguistic information (like: part of speech, inflectional class, gender, subcategorisation frame, semantic properties etc.) to all parts of the input text. If tokens are ambiguous, lexical analysis must provide all possible sets of annotation for later (syntactic) disambiguation, be it tagging, or full parsing.

2. Workflow

The workflow starts from crawled text. In PANACEA, a monolingual focused crawler is used (Pecina et al., 2011), starting from seed terms and seed URLs. It delivers texts in a basic format (called 'Travelling Object', following the XCES standard (Poch, Bel, 2011). All tools of the workflow read and write this format, this way enabling interoperability of the tools.
The crawler is responsible for producing a proper basic format; this includes document and paragraph markups, and all types of information which can be extracted from HTML documents, like titles, list elements, tables etc. This is valuable information, and provided to the workflow as annotations of the <doc> and <p> tags.
A header element keeps track of the document source, and the status of processing.

2.1 Document and paragraph information

The first set of tools is responsible for providing global information, which will help in later analysis. These tools read documents and paragraphs, and write annotations into the <doc> and <p> tags that can later be used. The main tools in the workflow are:

a. Language Identification
This is needed to load the proper language resources. A word-based approach is used here (based on spell-checking dictionaries of different languages), as it is easier to adapt and to maintain than n-gram-based approaches, and more accurate in cases of closely related languages (like Farsi and Dari).

b. Topic Identification
Topics are detected on document and paragraph level (for multi-topic documents). Topic identification is a useful tool in cases where lexica are sensitive for readings belonging to certain topics. In the present workflow, the *taxonomy* is taken from standard Machine Translation domain classifications. Examples are '*art'*, '*technology'*, '*wood processing'* as subtopic of '*material'* etc.
The topic *features* were extracted from training data, and verified manually to reduce noise. Topic features are lemmata, and can be multiwords, which increases precision.

[1] This research has been supported by the EU research program, project PANACEA, FP7-ICT-2009-4-248064

Resources: The system knows about 60 topics, each described by at least 500 features; size of the feature files is around 40000 entries.
Evaluation: Assignment quality is about 80% correctness for unrestricted text (Thurmair, 2005).

c. Casing
A small component determines the global casing of a paragraph (headings tend to be all-uppercase). Casing information influences the further processing steps, it is communicated as a <p> annotation.

2.2 Preprocessing

The task of the preprocessing components is to prepare the text, and its single tokens, for lexical analysis: Preprocessing must provide strings which can be used for lexicon lookup. Preprocessing consists of the following steps:

a. Sentence Splitting
Sentences are the standard unit in language processing, although their embedding into paragraph contexts becomes more and more important.
Experiments with the WACKy corpus[2] have shown that 70% of the documents are affected by different approaches towards sentence splitting, which makes a huge difference for later sentence alignment.
The naïve approach using periods and abbreviation lists fails in cases of German where periods have many more functions (e.g. they describe ordinals). Therefore, in the current workflow, sentence boundaries are detected using much more sophisticated resources.
Resources: The sentence splitter uses the following resources for each language:

- lists of words which indicate a sentence beginning if capitalised (like '*The'*).
- lists of words which frequently occur before a sentence-final punctuation (i.e. they indicate that a following period is really a sentence-end)
- lists of abbreviations. Abbreviations are further subcategorised into such that can only occur in final position (like '*etc.*'), only in non-final position (like '*Dr.*'), and the others.

The startwords and endwords have been collected from a corpus analysis of the WACky corpus, and manually corrected. They comprise about 12,000 entries per language[3] .

b. Tokenisation and Normalisation
It is an open issue what the definition of a token, in relation to words, may be. Challenges are agglutinations like clitics or compounds, multiword tokens like '*parce que*', digits like '*40 million*', units like '*200km/h*'. Moreover, the definition is script-dependent, thinking of Arabic or Chinese scripts.
From the point of view of lexical analysis, a token is anything that can be looked up in the lexicon, or can be given a valid linguistic description (like cardinal numbers). Therefore, tokenisation must be in line with the lexicon lookup; the tokeniser prepares the lexicon lookup. Another relevant aspect here is that the lexicon contains design decision as to what the normal form of a lemma would be: If the lexicon decides to use British Spelling as its base then the tokeniser must deliver British English strings. The same holds for German old vs. new spelling; for hyphenation in English ('*arm rest*' vs. '*arm-rest*' vs. '*armrest'*), and other such phenomena. Normalisation must map an incoming string to a canonical lexical representation. This is the more important the less standardised a language is. It is the task of the lexicon design to decide on the canonical form of entries; depending on this decision the tokeniser / normaliser must be designed.
The output of the tokeniser component is a list of tokens, i.e. strings which can undergo lexical analysis.
Resources: Tokeniser and Normaliser use rules and lists mapping 'uncanonical' to 'canonical' strings, several thousand entries per language.

2.3 Lexical Analysis

The task of lexical analysis is to assign information to tokens, usually as annotations in the form of feature value pairs. This information is the basis of further (syntactic / semantic) processing; strings without annotations are usually not usable in later steps.

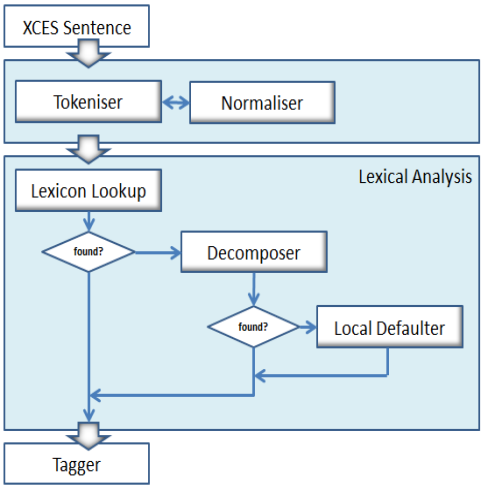

Fig. 1: Lexical Analysis Architecture

a. Lexicon Lookup / Lemmatisation
Tokens first undergo lexicon lookup. The whole lexicon is organised as a two-part resource:

- The first resource is based on *text forms*, and provides links to the lemma, and textform-specific information (like part-of-speech and inflection information). This resource is useful for language model building and similar kind of tasks.

[2] Baroni et al., 2009
[3] The strategy works even for cases in HTML texts where no punctuations are available at sentence ends, with a precision of about 85%.

- The second resource is *lemma-based* and provides lemma-specific information, like subcategorisation frames, semantic features etc., and is required for parsing and deeper processing.

For the lexical analysis workflow, the first resource (Full Word Dictionary) is compiled from the second one (Basic Lemma Dictionary). Each entry in the Full Word Dictionary consists of a triple of <textform, lemma, POS>. The POS information follows three levels of tags:

- a *basic* tagset (BTag) consisting of about 12 tags (noun, verb, pronoun etc.);
- the *standard* tagset (STag), comprising about 80 tags, differentiating according to syntactic distribution;
- and the *extended* tagset (XTag) with several hundred tags, extending the standard tagset by morphological features (gender, case, tense, number etc.).

In the current system, lemmata are described by basic tags while textforms use the standard or extended tagset.

The lexicon lookup accesses a (compiled version of) a full word lexicon, and returns either one reading (with lemma and POS), or several readings (in this case it is the task of later contextual disambiguation components like a Tagger to decide which one is correct in a given context); or it returns no reading, in case the string cannot be found in the lexicon.

So, the treatment of unknowns is an essential part of lexical analysis. Its importance of course depends on the lexicon coverage.

Resources: Full word dictionaries are used for German and English, and Standard Tagset annotations. The German lexicon has 889K entries, representing 208K different lemmata; the English lexicon is smaller, has 167K entries representing 40K lemmata.

Evaluation is given below.

As for speed, the component processes 115,000 words per second on a standard PC; so it can analyse the Europarl / JRC / EMEA corpus in about 10 minutes.

b. Decomposition

An analysis of unknown words in German shows that most of them are compounds. Therefore lexical analysis of unknown words in German continues with a decomposition step.[4]

The task of the decomposer is to provide lemma and POS information to an unknown input token, assuming that this information can be inherited from the compound head.

Resources: The Decomposer uses three resources:

- It has a *lexicon* of compound elements. This lexicon is a full word dictionary, consisting of <textform, lemma, tag> information. The tags are members of a special tagset of about 60 elements reflecting the distribution of elements inside compounds (like: '*only as first element'*, '*only before hyphen'*, *'location prefix'* etc). The dictionary contains about 470 K composition elements; including irregular forms (about 12 K entries), in cases where the standard heuristics would fail.
- It also uses a *transition table* deciding on the legal sequences of tags within a compound. The table is 60x60 fields big.
- As the system will return several decomposition results, a *rule set* is applied for *disambiguation*, based on information such as number of elements, part-of-speech of some elements, availability of certain affixes etc. Some of these rules are:
  - *filterLowCap*: If the word is not capitalized, and we have some readings which are nouns and others which are verbs or adjectives, this rule prefers the non-nouns.
  - *filterANhomo*: If the input token is capitalized, and we have competing nominal and Ad/No homograph readings: Keep them all.
  - *filterNLast*: This rule prefers compounds with nominal head category over other compounds.

  About 20 such rules are used for this.

Decomposition identifies lexically valid substrings in the input word; if it finds one then the transition to the left neighboring entry is validated using the transition table. This way, all linguistically possible decompositions are found. In a second step these decompositions are validated by applying the disambiguation rules, and the best decomposition is identified (one or several).

The speed of the component is about 13,000 words per second, fast enough for large-scale applications.

The output of the decomposer is the lemma and POS of the whole compound (this is the information required for lexical analysis), as well as the compound structure and parts (which can be used e.g. to default translations for such unknown words).

After decomposition, there is still an amount of tokens which is not known to the dictionary, and cannot be decomposed. For these tokens, local defaulting is applied.

c. Local Defaulting

A closer analysis of the left-overs of the decomposition step shows that the non-analyzable strings fall into one of the following classes:

- spelling errors, very frequent esp. in HTML text
- foreign language words, occurring as part of sentences
- acronyms and other mixed-character classes
- unknown regular words and / or proper names

To identify spelling errors, a component like *SmartCorrect* has been developed; it proposes proper words with a precision of about 85%[5]. To identify foreign language words, the same resource as for language identification is used; the challenge is to assign a POS value to these words. For acronyms (like '*320i*' or '*AZ45/1994*') a string-shape-based approach has been taken; it assigns a special noun-like POS to those elements.

[4] For English, words like *counterattack, armrest, battleship,* indicate that a decomposer could also be helpful.

[5] Aleksić/Thurmair 2011

For the last category, unknown regular words, a special component was built which does local defaulting. It is called 'local' as it does not look at contexts (e.g. agreement patterns, POS context or so) but takes just the single token as input.

Local defaulting tries to assign information to unknown strings based only on the shape of such strings. For example, it makes use of the fact that strings ending in '*-ersson*' are person names, or strings ending in '*-isations*' are plural common nouns.

The local defaulter defaults the features expected by later components (esp.: POS, lemma, gender, inflection class etc.) one by one, using heuristics derived from training data. For POS defaulting, the training data consist of (inflected) word forms with attached POS information; from them the system derives heuristics how unknown tokens may have to be analyzed. In cases where several alternatives are possible a score is returned for each reading with the probability of this reading.

Resources: The defaulter uses the following resources:

- Lists of *foreign words*; they are used to check if an unknown word comes from a foreign language. For this purpose, the word lists of the Language Identifier are re-used. Many unknown tokens in the test corpus are foreign language words.
- *Default endings*: These resources are created by a training component which correlates some linguistic information with string endings. Such information can be: Tags (BTag, STag, XTag), lemma formation, gender defaulting, etc. It takes a list of example words, and linguistic annotations of them, and produces the longest common ending strings for this annotation.

  For the defaulting of the part of speech, the training component produces about 470 K correlations of endings and tags assignments; English defaulter is smaller as there is less variance in the language. In case of homographs the resource also gives the relative weights of the different tags against each other, based on the training data.[6]

Defaulting can do about 30.000 words per second.

As the result of lexical analysis, *all* input tokens have a part-of-speech tag which is understood by the following components, delivered as a lattice of readings per token; so that they can be analyzed by taggers etc. with some chance of success.

3. Evaluation

3.1 Evaluation Data

The following corpora were used for test and evaluation:

- For German: dpa news, Süddeutsche Zeitung, WACKy[7], Wikipedia, Europarl[8], JRC/Acquis[9], emea[10]. The overall size of the corpus material is more than 2.5 bn tokens.
- For English: Reuters news, WACKy, Europarl, JRC/Acquis, emea, overall resulting in also more than 2.5 bn tokens.

Random portions of text were taken for the different tasks. In addition, a control corpus was created from the PANACEA corpus[11] for environment (ENV) and labor (LAB); 50 documents were taken from each set, containing 160K (ENV) and 50 K (LAB) tokens respectively.

3.2 Tool Evaluation

This section describes the evaluation of the different single tools included in the workflow.

a. Sentence Segmentation

Sentence segmentation was evaluated in an absolute and in a comparative way. For absolute evaluation, a set of randomly selected sentences was collected from several corpora, 3800 sentences for German and 3000 for English, and manually evaluated. Error rates were 0.44% for German and 0.26% for English texts. For comparative evaluation, a random comparison with a tool like TreeTagger showed a reduction of segmentation errors by 27% on a test file with several thousand sentences.

b. Tokenisation

Tokenisation was evaluated on the basis of tokens which were left over after all analysis steps: Tokens containing strange characters (like non-breakable spaces etc.) were found. Overall, an error rate of about 1.3% was computed for this component. The PANACEA control corpus had even smaller error numbers (0.22%).

c. Lexicon Lookup

There are two factors which determine the quality of a dictionary:

- *Internal quality*, i.e. correctness of annotations. This is usually evaluated by looking at a subset of the entries, and checking its correctness
- *Coverage*, i.e. how well does the dictionary match the intended application domain (a 100K dictionary of spare parts of agricultural machines does not really help in a medical domain).

(For external delivery, additional criteria are required, cf. the ELRA Validation manual for dictionaries).

Quality: No larger lexicon is without errors. So, lexicon lookup can produce errors when lexical entries are incorrectly coded. To have an impression of the lexicon error rate, 1000 random entries were extracted from the XTag German lexicon, and manually inspected.

The error rate was 1.29% for standard tag entries (i.e. errors in lemma or POS), and 2.53% for extended tag entries (i.e. half the errors were due to incorrect

[6] For the current version, only the STag defaulter is used; following versions will default more features if the approach turns out to be viable.

[7] Baroni et al., 2009

[8] Koehn, 2005

[9] Steinberger et al., 2006

[10] Tiedemann, 2009

[11] Pecina et al., 2011, Toral et al., 2011

morphological annotations). In the experiments presented here, only the standard tag annotations were used; so 98.7% of the words resulting from lexical analysis can be assumed to be correct.

Coverage: For German, the test corpus consisted of all tokens of Europarl, EMEA, and JRC corpora, overall 65 mio. After lexical analysis, 7.795 mio were left unknown; this is a coverage of 88.1% for the lexicon. The PANACEA control corpus showed similar results, with a coverage of 84% for German.

For English, the test corpus consisted of the WACky (2.3 bn), the Europarl (27 mio), and the Reuters (233 mio) corpus, overall 2.56 bn tokens. Of these, 185 mio were left unknown, which is a coverage of 92% for the lexicon. Of these, Reuters is a bit worse (89% coverage) than the others as the English lexicon contains very few proper names.

d. Decomposition

Decomposition was only tested for German. Test data were all tokens left unknown by the lexicon lookup (about 7.79 mio tokens, 3.18 mio types). As for lexicon lookup, quality (decomposition accuracy) and coverage were tested.

Accuracy: Decomposition quality / *accuracy* was evaluated by manually inspecting a random subset (14.300 entries) of decompositions. Of these, 2.18% were found to be incorrect. For the PANACEA control corpus, *all* compounds in the set were evaluated (16.700), the error rate is slightly higher (3,4%).

Main error types were proper names identified as compounds (esp. location names like ‚*nieder+au*', ‚*klein+schön+ach*', ‚*renn+steig*'), lexicon gaps (e.g. ‚*günstigkeit*' in ‚*günstigkeits+prinzip*' or ‚*vernässung*' in ‚*wieder+vernässung*' were not in the lexicon), and foreign language words (like ‚*attribut+ion*').

Coverage: In the PANACEA control corpus, the tokens remaining unknown after decomposition were inspected, to find out cases which *should* have been decomposed but in fact were not. It turned out that of all compounds in the corpus, 98.2% were in fact detected by the decomposer; so it missed only very few compounds.

All tokens that cannot be decomposed are sent to the POS defaulter.

e. Local POS Defaulting

The local defaulter assigns lexical information to *all* tokens (so the coverage is 100% by definition). Input of the component were the decomposer output files, i.e. all strings which were neither known to the lexicon, nor decomposable. Only German was evaluated here.

The question to evaluate is how good such an assignment can be. The component was evaluated such that entries of the test file were manually inspected. If *one* of the assigned POS tags was correct, the assignment was considered valid, as the tagger has a chance to find the correct POS. If this was not the case this was counted as an error. Also, tokeniser errors were counted as errors here (as the *final* assignment is incorrect).

It turned out in the tests that the quality of the component depends both on the tags to be defaulted and on the kind of input, in particular how 'regular' the input is (i.e. how close to the training data it is).

Tests with a dictionary (clean data, 700 K tokens) showed an error rate of 5,28%. Tests with the 'leftovers' of the Europarl/EMEA/JRC data (of which 21.000 entries were evaluated) show that

- if only *German* or German-looking words are considered, the error rate is 10.62%
- if *all* words are considered it drops to 20.5%, the reason being that there is a high percentage of foreign language words in this test set.

For the PANACEA control set, the error rate was 20,2%, in line with the results of the large test set.

It could also be noticed that the defaulting quality depends on the parts of speech to be assigned: Some parts of speech are defaulted more reliably than others, esp. nouns have rather low error rates, and so have participles. Verbs, esp. finite ones, have rather high error rates in the corpus data , which may indicate that most verbs are already in the dictionary, so defaulting produces mainly incorrect hypotheses.

Overall, assuming the defaulter works with 'in-language' material, the error rate related to the whole test file would be 10.6%, and for all kinds of material, it would increase to about 20%, which would mean that, after dictionary lookup and decomposition, 80-90% of the remaining material would be assigned a correct part of speech. This is quite superior to an approach where *all* possible POS values are assigned to an unknown token, introducing significant noise for the following components.

3.3 Workflow Evaluation

If the different components of lexical analysis are combined in a workflow, then the question is how the errors behave. Usually errors in a workflow accumulate, and the different components, fed with incorrect input, tend to produce more errors as in single tool evaluation.

For instance, in the overall workflow, the decomposer also must process tokens which are not compounds (as it is given *all* tokens that did not pass the lexicon filter). Its performance in the workflow therefore differs from its performance as a stand-alone tool (where only compound candidates are analysed), as the basis of comparison is not just the compounds but *all* tokens after lexical analysis.

For the workflow analysis, the whole *chain* of tools needs to be inspected, whereby the respective next component operates on the output of the previous one. What is of interest is again:

- Coverage: Although the coverage is 100%, as the defaulter always assigns a POS tag, it is still interesting which component contributes how much to the overall analysis;
- Accuracy: How many errors will each of the participating components produce, and how is the error rate of the whole workflow?

For the error rates, the approach was to determine how many incorrect results a given component would produce,

given the component error rate determined above; this way the overall number of incorrect entries can be determined.
Results for German, and for the main test set (Europarl/EMEA/JRC) and for the PANACEA control set are given in Tab. 1.

	tokens (K)	coverage (%)	error rate (%)	wrong tokens (K)
total tokens	**65,568**			
lexicon lookup found	57,772	88.1%	1,29	745
remaining	7,796			
decomposition found	5,807	74.4%	2,18	126
remaining	1,988			
POS defaulting for	1,988	100%	20,5	397
remaining	0			**1,269**
total correct assignments		**98.06%**		

Table 1: Workflow results for the Europarl/EMEA/JRC test set

It shows that the whole corpus of 65 mio tokens can undergo lexical analysis with an accuracy of 98.06%, taking into account the coverage and the error rates of the single tools. It should be noted that the corpus is not very clean[12], which is reflected in the higher error rates of the POS defaulter.
The same analysis was done for the PANACEA control corpus, and the following data were evaluated, cf. Tab. 2:

	tokens (K)	coverage (%)	error rate (%)	wrong tokens (K)
total tokens	**206.2**			
lexicon lookup found	172.9	83.8%	1,29	2.2
remaining	33.2			
decomposition found	22.4	67.4%	3.4	0.7
remaining	10.8			
POS defaulting for	10.8	100%	20,2	2.1
remaining	0			**5.0**
total correct assignments		**97.48%**		

Table 2: Workflow results for PANACEA control test set

The two test sets show similar behavior, with a slightly worse performance of the decomposer in the PANACEA control set.

[12] It contains very many foreign language words, and lots of unknown medical substances and product names from emea.

Conclusion
As a conclusion, it can be seen that the complete workflow of lexical analysis and POS assignment has an accuracy of around 98%, for large scale data.
The work reported here shows that it is quite possible, especially in the era of corpus linguistics and data driven approaches, to run linguistic analysis with high accuracy and sufficient speed, to serve as the basis for further knowledge-driven as well as improved data-driven applications. Of central importance is the availability of language resources, both in quantity and in quality.

4. Integration

The whole workflow is implemented as a sequence of web services, following the PANACEA specifications, supporting the format of the Travelling Object specified there, and being implemented in the Taverna system as a workflow.

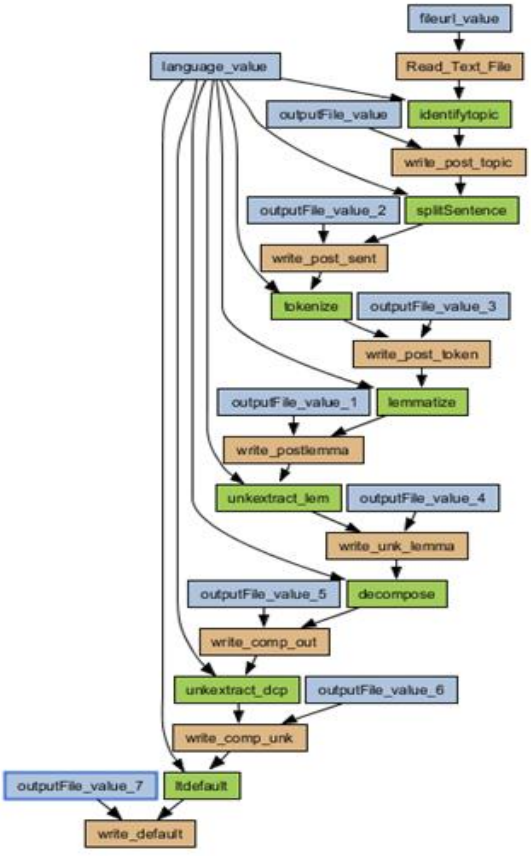

Fig. 2: Lexical analysis workflow in PANACEA

The workflow assumes a crawled file in XCES format as input, and includes the following web services (green boxes):

- identifyTopic: Topic Identification
- splitSentence: Sentence Splitting
- tokenise: Tokeniser and Normaliser
- lemmatise: Lexicon lookup and lemmatisation
- decompose: Decomposer
- default: Local POS defaulting
- an auxiliary service for unknowns extraction

It produces an output list of remaining unknown tokens, with POS proposals by the system.

The services are registered in the PANACEA registry as single web services, and combined into a Taverna workflow.

5. References

Aleksić / Thurmair 2011: Personal Translator at WMT 2011. Proc. WMT 2011, Edinburgh

Baroni, M., Bernardini, S., Ferraresi, A., Zanchetta, E., 2009: The WaCky wide web: a collection of very large linguistically processed web-crawled corpora. in: Language Resources and Evaluation 43,3

Bel, N., 2010: Platform for Automatic, Normalized Annotation and Cost-Effective Acquisition of Language Resources for Human Language Technologies: PANACEA. *Proc.* SEPLN, Valencia

Koehn, Ph., 2005: Europarl: A parallel corpus for Statistical Machine Translation. Proc. MT Summit, Phuket.

Pecina, P., Toral, A., Way, A., Papavassiliou, V., Prokopidis, Pr., Giagkou, M., 2011: Towards using web-crawled data for domain adaptation in statistical machine translation. Proc. 15th EAMT, Leuven

Poch, M.; Bel, N., 2011: "Interoperability and technology for a language resources factory". Proc. Workshop on Language Resources, Technology and Services in the Sharing Paradigm at IJCNLP (Chiang Mai, Thailand).

Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufiş, D., Varga, D., 2006: The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. Proc. 5th LREC'2006, Genova

Thurmair, Gr., 2005: Improving machine translation quality. Proc. MT Summit X, Phuket, Thailand

Tiedemann, J., 2009: News from OPUS – A collection of Multilingual Parallel Corpora with Tools and Interfaces. in: N. Nicolov and K. Bontcheva and G. Angelova and R. Mitkov (eds.) Recent Advances in Natural Language Processing (vol V) (Benjamins)

Toral, A., Pecina, P., Way, A., & Poch, M., 2011: Towards a user-friendly webservice architecture for statistical machine translation in the PANACEA project**.** Proc. EAMT Leuven.