Compilation and Exploitation of Parallel Corpora

Tomaž Erjavec

Dept. of Intelligent Systems, "Jožef Stefan" Institute, Ljubljana, Slovenia

With more and more text being available in electronic form, it is becoming relatively easy to obtain digital texts together with their translations. The paper presents the processing steps necessary to compile such texts into parallel corpora, an extremely useful language resource. Parallel corpora can be used as a translation aid for second-language learners, for translators and lexicographers, or as a data-source for various language technology tools. We present our work in this direction, which is characterised by the use of open standards for text annotation, the use of publicly available third-party tools and wide availability of the produced resources. Explained is the corpus annotation chain involving normalisation, tokenisation, segmentation, alignment, word-class syntactic tagging, and lemmatisation. Two exploitation results over our annotated corpora are also presented, namely a Web concordancer and the extraction of bi-lingual lexica.

Keywords: natural language processing, corpus annotation, multilinguality, lexicon extraction.

1. Introduction

With more and more text being available in electronic form, it is becoming easy to obtain large quantities of digital texts and to process them computationally. If a collection of such texts is chosen according to specific criteria and is consistently and correctly marked-up [16], it is said to be a text corpus. Such corpora can be used for a variety of different purposes [14], from empirically grounded linguistic studies, lexicography and language teaching, to providing datasets for language technology programs for terminology extraction, word-sense disambiguation, etc.

Collected and uniformly encoded collections of texts are already quite useful, but it is the addition of (linguistic) markup that makes corpora a prime resource for language exploration. As will be seen, we view the process of compiling a corpus as one of annotation accrual: starting from plain text, we successively add mark-up, thereby enriching the information contained in the corpus. This markup is typically produced automatically, but can be hand validated and, in a cyclic process, can serve for inductive programs to learn better models of the language with which to annotate subsequent generations of corpora. The added annotation enables the people and software using the corpus to employ extra levels of abstraction, leading to better exploitation results.

If monolingual corpora are already useful for a variety of purposes, it is multilingual corpora that open the way for empirical study of the translation process. Especially valuable are so called parallel corpora, i.e., corpora consisting of texts together with their translation into one or many languages. They can be used directly as translation aids for humans or can provide data for the automatic induction translation resources (lexica) and software (machine translation).

In this paper we explore the process of compilation and exploitation of such parallel corpora, grounding the discussion in our experience with two annotated parallel corpora: the 7-language MULTEXT-East corpus [5, 8], which contains the novel "1984" by G. Orwell (100,000 words per language) and has had its annotation manually validated; and the larger (500,000 words per language) automatically annotated Slovene-English IJS-ELAN corpus [7].

Our work is characterised by the use of open standards for text annotation and publicly available third-party tools. We claim that it is better

to invest labour into producing high-quality annotated corpora than in trying to build, from scratch, tools for such annotation. Unlike local and idiosyncratic software, linguistic resources encoded in a standard manner will be sooner useful to other research groups. Such largess aside, there also exist more and more (statistical or symbolic) machine learning programs that are able to induce language models from pre-annotated corpora. They are typically more robust and, with sufficiently large training sets, might even perform better than hand crafted systems.

The rest of this paper is structured as follows: Section 2 introduces standards for corpus annotation, which are then used in the examples in the remainder of the paper; Section 3 enumerates the basic (pre-linguistic) processing steps involved in the compilation of a corpus; Section 4 details the more complex word-level syntactic annotation, which is performed by a trainable algorithm; Section 5 turns to the exploitation of corpora and gives two examples: an on-line concordancer, and an experiment in bi-lingual lexicon extraction; Section 6 gives conclusions and directions for further work.

2. Encoding Standards

While the question of the encoding format for corpora and other language resources might seem incidental to the main task of producing and exploiting the corpora, it has long been known that the proliferation of data formats and annotation schemes, many of which are proprietary and poorly documented, are a significant bottleneck for resource sharing, re-use and longevity. There have been, therefore, a number of attempts to standardise the encoding of various language resources, corpora among them.

All such standardisation efforts have as their basis the ISO Standard Generalized Markup Language, SGML, or, more recently, the W3C Extensible Markup Language, XML [21], a simplified form of SGML meant primarily for interchange of data on the Web. SGML and XML are metalanguages, that is, a means of formally describing a language, in this case, a markup language. Thus, they do not directly define a particular set of tags but rather enable the mechanisms for defining such sets for particular purposes, e.g., for the encoding of corpora.

The best known and widely used set of conventions for encoding a wide variety of texts, corpora among them, are the SGML-based Text Encoding Initiative Guidelines (TEI), the most recent version of which is also XML compliant [17]. The TEI consist of the formal part, which is a set of SGML/XML Document Type Definition fragments, and the documentation, which gives the semantics of the elements available in these fragments, as well as overall information about the structure of the TEI. The DTD fragments are combined to suit an individual project, and, if necessary, can also be extended or modified. We have used parametrisations of TEI for both the MULTEXT-East corpus as well as for the IJS-ELAN corpus. TEI encoded examples from these corpora will be used in the rest of this paper.

3. Pre-processing the Corpus

In this section we deal with the basic processing steps involved in normalising and marking-up the corpus, in order to make it minimally useful for exploitation and to prepare it for further annotation. The steps we outline below usually proceed in sequence, and can be, for the most part, performed automatically, although — given the unconstrained nature of texts — they are likely to be less than 100% accurate. While further development of tools and associated resources lowers the error rate, manual validation might still be necessary for high-quality corpora.

The texts constituting a corpus can be garnered from a variety of sources and can come in a range of formats. Therefore, the first step in corpus preparation is invariably normalisation of the texts into a common format. Usually custom filters — written in pattern matching languages such as Perl — are employed to, on the one hand, normalise the character sets of the documents and, on the other, to remove and convert the formatting of the originals.

3.1. Character Sets

As far as character sets go, the corpus compliers have a few options at their disposal. One possi-

bility is to use — at least for European languages — an 8-bit encoding in the corpus, preferably a standard one, e.g., ISO 8859-2 for encoding Central and Eastern European language texts. While the advantage is the fact that the corpus texts are immediately readable — given that we have installed the appropriate fonts — the disadvantage is that a number of processing applications do not handle well 8 bit characters and, more importantly, that it is impossible to mix languages that use different character sets; this is, of course, a special concern in multilingual corpora.

Until a few years ago the standard solution involved translating the non-ASCII characters of the original texts into ISO-mandated SGML entities. SGML (and XML) entities are descriptive names, somewhat like macros, which the application then substitutes for their values. In our case, the names are of the characters in question; so, for example, the Slovene letter *č* is written as the entity `&ccaron;` (small c with a caron), where ampersand starts an entity reference and semicolon ends it. Such entities for characters are defined in public entity sets, for the case of `&ccaron;` in `ISO 8879:1986//ENTITIES Added Latin 2//EN`, i.e., the added entity set for encoding the Latin alphabets of Central and Eastern European languages. Similar entity sets exist for Western European languages (Latin 1), for Greek, Russian and non-Russian Cyrillic, as well as for mathematical relations, publishing symbols, etc. The advantage of using entities is their robustness: being encoded in (7 bit) ASCII characters, they are portable, and can be read on any platform. For the application, the SGML processor then translates them into the desired encoding via their definitions.

With the advent of XML, this solution has somewhat lost its currency. While entities are supported in XML, the default character set of XML is Unicode, which, because a character can be encoded in two bytes, is sufficient to accommodate most of the characters of the world's scripts. But while using Unicode makes for a theoretically good solution, practice still lags behind, with many applications not being Unicode-aware yet. In our corpora we have, therefore, chosen to use SGML entities for the representation of non-ASCII characters.

3.2. Markup of Gross Document Structure

Various encodings of input texts also encode the structure of the documents in vastly different and often inconsistent manners. This structure includes such things as divisions of the text, headers and titles, paragraphs, tables, footnotes, emphasised text, etc. In general it is a very hard task to correctly and completely transform this structure into descriptive TEI markup. For many natural language processing applications this might not even be necessary, as the goal here isn't to preserve the layout of the text, but only the information that is relevant to correctly classify the text and to enable linguistic processing. To illustrate a case of relatively detailed structure markup, we give, in Figure 1, a TEI encoded example from the MULTEXT-East corpus.

```
<text lang="en" id="0en.">
 <body>
  <div id="0en.1" type="part">
   <head>First part</head>
   <div id="0en.1.1" type="chapter">
    <head>I</head>
    <p id="0en.1.1.1">
It was a bright cold day in April, and the
clocks were striking thirteen. Winston Smith,
his chin nuzzled into his breast in an effort
to escape the vile wind, slipped quickly
through the glass doors of Victory Mansions,
though not quickly enough to prevent a swirl
of gritty dust from entering along with him.
    </p>
    ...
```

Fig. 1. Structure markup in TEI.

3.3. Header Information

Typically, a corpus is composed of a large number of individual texts. For analysing the corpus or for choosing a particular subset out of it, it is vital to include information about the texts into the corpus. Of course, the corpus as a unit must also be documented. The TEI provides a header element, <`teiHeader`> expressly meant to capture such meta-data. The TEI header contains detailed information about the file itself, the source of its text, its encoding, and revision history.

Depending on the number of texts and the regularity of provenance, the information in a text or corpus header can be inserted either manually or automatically.

3.4. Tokenisation and Segmentation

The next step in corpus preparation already involves basic linguistic analysis, namely isolating the linguistic units of the text, i.e., words and sentences. Identification of words — and punctuation marks — is usually referred to as tokenisation, while determining sentence boundaries goes by the name of segmentation.

On the face of it, determining what is a word and what a sentence might seem trivial. But correctly performing these tasks is fraught with complexities and the rules to perform them are, furthermore, language-dependent. So, while sentences end with full stops, not every full stop ends a sentence, as with, e.g., *Mr.*; and if some abbreviations will never end sentences, e.g., *e.g.*, other almost invariably will, e.g., *etc.* Correct tokenisation is complex as well; punctuation marks can sometimes be part of a word, as is the case with abbreviations and, say, Web addresses. Some domains, for example biomedicine have "words" with an especially complex internal structure, e.g., *Ca(2+)-ATPase*.

In the process of tokenisation various types of words and punctuation symbols must be recognised and this information can be retained in the markup, as it can be potentially useful for further processing. In Figure 2 we give an example of a tokenised segment from the IJS-ELAN corpus.

```
<seg id="ecmr.en.17">
  <w>Euromoney</w><w type="rsplit">'s</w>
  <w>assessment</w> <w>of</w> <w>economic</w>
  <w>changes</w> <w>in</w> <w>Slovenia</w>
  <w>has</w> <w>been</w> <w>downgraded</w>
  <c type="open">(</c><w>page</w>
  <w type="dig">6</w><c type="close">)</c>
  <c>.</c>
</seg>
```

Fig. 2. Segmentation and tokenisation in TEI.

While it is possible to write a tokeniser and segmenter using a general purpose computer language, there also exist freely available tools for this purpose. We have extensively used the MULTEXT tools [3], which, however, no longer seem to be maintained.

Fortunately, there are other good choices, e.g., the text tokeniser tool, LT TTT [10], which is freely distributed for academic purposes as binaries for Sun/Solaris. LT TTT is based on XML and incorporates a general purpose cascaded transducer which processes an input stream deterministically and rewrites it according to a set of rules provided in a grammar file, typically to add mark-up information. With LT TTT come grammars to segment English texts into paragraphs, segment paragraphs into words, recognise numerical expressions, mark-up money, date and time expressions in newspaper texts and bibliographical information in academic texts. These grammars are accompanied by detailed documentation which allows altering the grammars to suit particular needs or develop new rule sets.

3.5. Sentence Alignment

An extremely useful processing step involving parallel corpora is the alignment of their sentences. Such alignment would be trivial if one sentence were always translated into exactly one sentence. But while this may hold for certain legal texts, translators — either due to personal preference, or because different languages tend towards different sentence lengths — often merge or split sentences, or even omit portions of the text. This makes sentence alignment a more challenging task.

High-quality sentence alignment is still an open research question and many methods have been proposed involving the utilisation of e.g., bilingual lexicons and document structure. Still, surprisingly good results can be achieved with a language-independent and knowledge poor method, first discussed in [9]. The alignment algorithm here makes use of the simple assumption that longer sentences tend to be translated into longer sentences, and shorter into shorter. So, if we come across, e.g., two short sentences in the original, but one long one in the translation, chances are, the two have been merged. Hence the input to this aligner are only the lengths of the respective sentences in characters. The program, with an algorithm known as dynamic time warping, finds the best fit for the alignments, assuming the valid possibilities are 1-2, 0-1, 1-1, 1-0, 2-1, and 2-2.

There are several public implementations of this algorithm; we have used the so called Vanilla aligner [4], implemented in C and freely available in source code.

The quality of the automatic alignment is heavily dependent on the manner of translation but, in any case, is seldom perfect. For our corpora we have manually validated the alignments via a cyclic process, with initial errors of alignment corrected and the text then automatically re-aligned. The process is less labour intensive than it might seem, as errors tend to occur at relatively rare non 1-1 alignments.

The end result is the sentence aligned text; the alignment information may then be encoded in one of several ways. One possibility is to encode the alignments in separate documents, where only pairs of references to sentence IDs are stored. Figure 3 gives a hypothetical Slovene-English alignment span illustrating the syntax and types (one, many, zero) of the alignment links. The first link encodes a 1-1 alignment, the second a 2-1 and the third an 1-0 alignment.

```
<link xtargets="Osl.1.1 ; Oen.1.1"/>
<link xtargets="Osl.1.2 Osl.1.3 ; Oenl.1.2"/>
<link xtargets="Osl.1.4 ; "/>
```

Fig. 3. Example of stand-off bilingual alignment.

4. Word-class Syntactic Tagging

It is well known that the addition of word-level syntactic tags significantly adds to the value of a corpus [19]. Knowing the part-of-speech (e.g., that the word is a noun, an adjective, or a verb) and other morphosyntactic features (such as number, gender and case), helps to lexically determine the word and serves as a basis for further syntactic or semantic processing. In a parallel corpus such annotations can also act as guides for automatic bilingual lexicon extraction or example-based machine translation.

The flip side of morphosyntactic tagging is lemmatisation. Here, an (inflected) word in the corpus, say *walks* is annotated with its base form or lemma, i.e., *walk*. Such a normalisation of word-forms is useful in many applications where we wish to abstract away from the surface realisations of the word, for example in concordancing or in identifying translation equivalents. While lemmatisation in English is relatively simple (although *wolves*, *geese*, and *oxen* complicate matters) it is a more difficult task in heavily inflecting languages, such as Slovene.

Manual annotation is extremely expensive, so corpora are typically tagged and lemmatised automatically. Below we explain our work on the IJS-ELAN corpus, where we used the statistical tagger TnT which had been trained on the MULTEXT-East parallel corpus, and the initial results improved in various ways.

4.1. The TnT Tagger

Trainable word-class syntactic taggers have reached the level of maturity where many models and implementations exist, several among them being robust and available free of charge. Prior to committing ourselves to a particular implementation, we conducted an evaluation (on Slovene data) of a number of available taggers [6]. The results show that the trigram-based TnT tagger [1] is the best choice, in terms of accuracy (also on unknown words) as well as efficiency. TnT is freely available under a research license, as an executable for various platforms.

The tagger first needs to be trained on an annotated corpus; the training stage produces a table with tag tri- bi- and uni-grams and a lexicon with the word forms followed by their tag ambiguity classes, i.e., the list of possible tags, together with their frequencies. Using these two resources, and possibly a backup lexicon, tagging is performed on unannotated data.

4.2. The Training Corpus

The greatest bottleneck in the induction of a quality tagging model for Slovene is the lack of training data. The only available hand-validated tagged corpus is the Slovene part of the MULTEXT-East corpus, which is annotated with validated context disambiguated morphosyntactic descriptions and lemmas.

These morphosyntactic descriptions (MSDs) for Slovene — and six other languages, English among them — were developed in the MULTEXT-East project [5, 8]. The MSDs are structured and more detailed than is commonly the case for English part-of-speech tags; they are compact string representations of a simplified kind of feature structures. The first letter of an MSD encodes the part of speech, e.g., `Noun` or

`Adjective`. The letters following the PoS give the values of the position determined attributes. So, for example, the MSD `Ncfpg` expands to `PoS:Noun, Type:common, Gender:feminine, Number:plural, Case:genitive`. In case a certain attribute is not appropriate for the particular combination of features or for the word in question, this is marked by a hyphen in the attribute's position.

To illustrate the properties of the training corpus, as well as the difference between Slovene and English, in Table 1 we give the number of word tokens in the corpus, the number of different word types in the corpus (i.e., of word-forms regardless of capitalisation or their annotation), the number of different context disambiguated lemmas, and the number of different MSDs. Inflectional nature of Slovene is evident from the larger number of distinct word-forms and especially MSDs used in the corpus.

	English	Slovene
Words	104,286	90,792
Forms	9,181	16,401
Lemmas	7,059	7,903
MSDs	134	1,023

Table 1. Inflection in the MULTEXT-East corpus.

4.3. Tagging Slovene

Unsurprisingly, the tagging produced by TnT trained only on the MULTEXT-East corpus had quite a low accuracy; this can be traced both to the inadequate lexicon induced, as more than a quarter of all word tokens in IJS-ELAN were unknown, as well as to n-grams applied to very different text types from those used for training. To offset these shortcomings we employed two methods, one of them meant primarily to augment the n-grams, and the other the lexicon.

It is well known that "seeding" the training set with a validated sample from the texts to be annotated can significantly improve results. We selected a sample comprising 1% of the corpus segments (approx. 5,000 words) evenly distributed throughout the corpus. The sample was then manually validated and corrected, also with the help of Perl scripts, which pointed out certain typical mistakes, e.g., lack of concordance in case, number and gender between adjectives and nouns. The tagger n-grams were then relearnt using the concatenation of the validated ELAN sample with the Slovene MULTEXT-East corpus.

It has also been shown [11] that a good lexicon is much more important for proper of inflective languages than higher-level models, e.g., bi- and tri-grams. A word that is included in a TnT lexicon gains the information on its ambiguity class, i.e., the set of possible context-independent tags, as well as the lexical probabilities of these tags.

The Slovene part of the ELAN corpus was therefore first lexically annotated, by courtesy of the company Amebis, d.o.o., which also produces the spelling checker for Slovene Word. The large lexicon used covers most of the words in the corpus; only 3% of the tokens remain unknown. This lexical annotation includes not only the MSDs but also, paired with the MSDs, the possible lemmas of the word-form.

We first tried using a lexicon derived from these annotations directly as a backup lexicon with TnT. While the results were significantly better than in the first attempt, a number of obvious errors remained and additional new errors were at times introduced. The reason turned out to be that the tagger is often forced to fall back on uni-gram probabilities, but the backup lexicon contains only the ambiguity class, with the probabilities of the competing tags being evenly distributed. So, in fact, TnT often assigned a random tag from the ones available, leading to poor results. To remedy the situation, a heuristic was used to estimate lexical frequencies of unseen words, taking as the basis the known frequencies from similar ambiguity classes taken from the training corpus.

Using this lexicon and the seeded model we then re-tagged the Slovene part of the IJS-ELAN corpus. By manually validating a small sample of the tagged corpus, consisting of around 5,000 words, we showed that the current tagging accuracy was about 93%.

As mentioned earlier, the lexical annotations included lemmas along with the MSDs. Once the MSD disambiguation had been performed it was trivial to annotate the words with their lemmas. But while all the words, both known and unknown, were annotated with an MSD,

we have so far not attempted to lemmatise approximately 3% of the corpus words which are unknown.

The results of the tagging were encoded in the corpus as attribute values of the TEI <w> element. To illustrate this, in Figure 4 we give a sentence as an example from the corpus: *Razlike med metropolitanskimi centri in njihovim zaledjem so ogromne. / The differences between the metropolitan centres and their hinterlands are enormous.*

```
<seg id="ecmr.sl.92">
  <w ana="Ncfpn" lemma="razlika">Razlike</w>
  <w ana="Spsi" lemma="med">med</w>
  <w ana="Aopmpi">metropolitanskimi</w>
  <w ana="Ncmpi" lemma="center">centri</w>
  <w ana="Ccs" lemma="in">in</w>
  <w ana="Ps3fpdp" lemma="njihov">njihovim</w>
  <w ana="Ncnsi" lemma="zaledje">zaledjem</w>
  <w ana="Vcip3p--n" lemma="biti">so</w>
  <w ana="Afpfpn" lemma="ogromen">ogromne</w>
  <c ctag=".">.</c>
</seg>
```

Fig. 4. Linguistic annotation in the corpus.

4.4. Tagging English

Tagging the English part of the corpus with the MULTEXT-East MSDs was also performed with TnT, using the English part of the MULTEXT-East corpus as the training set. However, automatic tagging with this model is bound to contain many errors, although less than for Slovene, given the much smaller tagset.

Rather than try to improve the accuracy of our own tagging, we opted for additional annotations with other, better, models and also with a better known tagset, namely (variants of) the one used in the Brown corpus [12]. For the additional annotation of the English part we combined the output of two taggers.

First, the TnT tagger distribution already includes some English models. We chose the one produced by training on the concatenation of the Brown corpus with the Wall Street Journal corpus; this training set contained approximately 2.5 million tokens and distinguished 38 different word-tags.

Second, we used QTag [13], which is also freely available, probabilistic tri-gram tagger, although the underlying algorithm differs from that employed by TnT. The English model of QTag uses a similar, though not identical, tagset to the TnT English one. QTag is also offered via an email service, which, in addition to tagging the texts, also lemmatises them; we used this lemmatisation to annotate the corpus.

5. Utilising the Corpus

The IJS-ELAN corpus was thus encoded in XML/TEI, segmented, tokenised, aligned and tagged with morphosyntactic descriptions and lemmas. It was now time to turn to exploiting the corpus. Given the fact that the texts included in the corpus do not have copyright restrictions (they are mostly publications of the Slovene government), it was trivial to ensure one type of "exploitation", namely to simply make the complete corpus freely available for downloading: it can be accessed at `http://nl.ijs.si/elan/`.

In this section we discuss two methods of utilising the corpus. The first is geared directly towards human usage, and has to do with making the corpus available for sophisticated online searching. The second employs a statistics-based tool that extracts a bi-lingual lexicon from the corpus.

5.1. Web Concordancing

For our corpora we have developed an on-line concordancing system. Concordancers are programs that, given a (possibly complex) query, return all the words or phrases in the corpus satisfying the query, together with their context; such programs are an invaluable aid in utilising the information contained in corpora.

Our Web concordancer, at `http://nl2.ijs.si/`, comprises a set of HTML pages, a simple Perl CGI script and a corpus processing back-end. The back-end is the CQP system [2], a fast and robust program, freely available for research purposes as binaries for a number of platforms. CQP supports parallel corpora and incorporates a powerful query language that offers extended regular expressions over positional (e.g., word, lemma, MSD) and structural (e.g., <text>, <p>, <seg>) attributes.

The Web page of the concordancer contains various input fields and settings available to the user. The settings and options have associated hyperlinks, and clicking on them gives help on the particular topic. So, for example, the Display setting affects how the search results are presented: the Bilingual Display shows the hits in the target corpus, followed by their aligned segment in the translation; the KWIC Display shows the results in the familiar key-word in context format; and Word List Display gives a list of word types found in the corpus, indicating their frequencies. The last option makes the most sense with fuzzy queries.

The result of the query can also be refined by specifying an additional query on the aligned corpus. This constraint can be either required or forbidden. The latter option is useful when exploring 'unexpected' translations.

The on-line concordancer has been in use at the Department of Translation and Interpreting at the University of Ljubljana, for different purposes such as contrastive analysis, translation evaluation, translation-oriented lexical and terminology studies, discourse analysis, etc. The methodological aims of this work were, on the one hand, to help students gain a deeper understanding of living languages and remember things they discover on their own and, on the other, to enable them to become skilled and critical users of the corpora for translation purposes.

The concordancer is also being used by translators, esp. by the volunteers of LUGOS, the Linux Users' Group of Slovenia, that are localising Linux documentation, e.g., the HOWTOs and the KDE desktop environment. As the IJS-ELAN corpus contains a whole book on Linux and the PO localisation files, it can be a welcome source of terminology translations.

5.2. Lexicon Extraction

We have also performed an initial experiment in automatic bi-lingual lexicon extraction from the corpus. Extracting such lexica is one the prime uses of parallel corpora. As manual construction is an extremely time consuming process, the resource is invaluable for lexicographers, terminologists, translators as well as machine translation systems.

Some preliminary experiments on the IJS-ELAN corpus had already been performed [20], using a variety of different tools. The software we have used here is the PWA system [18], which is a collection of tools for automatically finding translation equivalents in sentence aligned parallel corpora. The output of the system is, inter alia, a list of word token correspondences (i.e., translations in the text), a list of word type correspondences (i.e., a lexicon) and lists of monolingual collocations (i.e., a terminological glossary). The system is freely available under a research license as a binary for various platforms.

For the data, we have used one of the elements of the IJS-ELAN corpus, namely the book "Linux Installation and Getting Started" by Matt Welsh et al., translated into Slovene by Roman Maurer. The book contains $2 \times 5{,}773$ aligned sentence segments, with the English original having 91,526 and the Slovene translation 81,955 word tokens.

For lexicon extraction we have not used the word-forms directly, but rather the lemmas (where defined) of the words in question. This normalises the input and abstracts away from the rich inflections of Slovene, which would cause PWA to treat different forms of the same word as different words. Secondly, we reduced the input to only adjectives, nouns and punctuation symbols. The reasoning behind this is that noun phrases will be the most useful (terminological) lexical correspondences and eliminating other word classes reduces the chance of spurious translation correspondences. We have included punctuation signs in order to break up long stretches of nouns, which otherwise tend to get analysed as collocations.

For this input data the PWA system needed 15 minutes (on a Pentium laptop) to produce the results, i.e., the list of token correspondences totalling 22,880 items, the lexicon containing 2,850 entries, and a list of collocations with 1,329 entries.

To illustrate this, in Figure 5 we show the data for one sentence from the text; first we give the sentence and its translation, then the equivalent input to the system, and finally the computed translation equivalents.

English sentence: *In addition, all of the source code for the Linux system, including the kernel, device drivers, libraries, user programs, and development tools, is freely distributable.*

Slovene sentence: *Dodatno je dostopna in prosto razširljiva še vsa izvorna koda sistema Linux, vključno z jedrom, gonilniki naprav, knjižnicami, uporabniškimi programi in razvojnimi orodji.*

English input: `addition , source code linux system , kernel , device driver , library , user program , development tool , distributable .`

Slovene input: `dostopen razširljiva izvoren koda sistem Linux , jedro , gonilnik naprava , knjižnica , uporabniški program razvojen orodje .`

Output English → Slovene translations:
source code → izvoren
linux → linux
system → sistem
kernel → jedro
device driver → gonilnik
library → knjižnica
user → uporabniški
program → program
development → razvojen
tool → orodje

Fig. 5. Automatically extracted translation equivalents.

Most of the proposed translations are correct, but some are less than perfect. While the system correctly identifies translation equivalents for `linux` and `system`, it misses out on the larger collocation `linux system`, and similarly for `user program` and `development tool`. The main shortcoming of the output for the example sentence is the suggested translation equivalent for `source code`, as it lacks the noun `koda`. But despite these omissions, the result is (already) quite useful.

6. Conclusions and Further Research

The paper presented the processing steps involved in building and exploiting parallel corpora and introduced the tools necessary to accomplish this task. We have tried to show how third-party publicly available software is sufficient to operationalise the complete tool chain, and how language resources can be built in a cyclic manner, with initial annotations enabling the production of language models which, in turn, enable refinement and further annotation.

The text processing model outlined in this article is especially useful in an academic environment: the software to implement it is free, and can thus be easily acquired by cash-strapped university departments. The building as well as the exploitation of the resources can be profitably used for teaching purposes, be it for computer science courses on natural language processing, or for linguistic courses on the use of language technology. As has been shown, the resources can also be used directly, by helping translators or language students make use of bilingual data.

As far as corpus compilation goes, our further work can be divided into two areas. The first is, of course, the acquisition of more texts, and hence the production of larger corpora. The second, and scientifically more challenging, is the addition of further markup. In the paper we have discussed only basic linguistic markup. Useful further annotations include terms (either single or multiword), named entities (proper names, acronyms, dates, etc.), chunks (esp. noun phrases), and phrase structure (i.e., full syntactic analysis). Each of these areas has been the subject of much research but, so far, not yet attempted for Slovene.

On the exploitation side, in addition to carrying further our research on lexicon extraction, we plan to experiment with statistical machine translation, in particular to use the freely available system EGYPT [15] with the IJS-ELAN corpus as the training set.

Acknowledgements

The author would like to thank the company Amebis, d.o.o., for lexically annotating the Slovene part of the IJS-ELAN corpus and Jin-Dong Kim for useful comments on a previous version of this paper. All errors, of course, remain the responsibility of the author.

References

[1] T. BRANTS, (2000), TnT — A Statistical Part-of-Speech Tagger, in *Proceedings of the Sixth Applied Natural Language Processing Conference ANLP-2000*, pages 224–231, Seattle, WA, `http://www.coli.uni-sb.de/~thorsten/tnt/`.

[2] O. CHRIST, (1994), A Modular and Flexible Architecture for an Integrated Corpus Query System, in *Proceedings of COMPLEX '94: 3rd conference on Computational Lexicography and Text Research*, pages 23–32, Budapest, Hungary. http://www.ims.uni-stuttgart.de/projekte/CorpusWorkbench/.

[3] P. DI CRISTO, (1996), MtSeg: The Multext multilingual segmenter tools, MULTEXT Deliverable MSG 1, Version 1.3.1, CNRS, Aix-en-Provence. http://www.lpl.univ-aix.fr/projects/multext/MtSeg/.

[4] P. DANIELSSON, D. RIDINGS, (1997), Practical presentation of a "vanilla" aligner, in *TELRI Newsletter No. 5*. Institute fuer Deutsche Sprache, Mannheim, Mannheim. http://nl.ijs.si/telri/Vanilla/doc/ljubljana/.

[5] L. DIMITROVA, T. ERJAVEC, N. IDE, H.-J. KAALEP, Vladimír Petkevič, and Dan Tufiş, (1998), MultextEast: Parallel and Comparable Corpora and Lexicons for Six Central and Eastern European Languages, in *COLING-ACL '98*, pages 315–319, Montréal, Québec, Canada. http://nl.ijs.si/ME/.

[6] S. DŽEROSKI, T. ERJAVEC, J. ZAVREL, (2000), Morphosyntactic Tagging of Slovene: Evaluating PoS Taggers and Tagsets, in *Second International Conference on Language Resources and Evaluation, LREC'00*, pages 1099–1104, Paris. ELRA. http://nl.ijs.si/et/Bib/LREC00/lrec-tag-www/.

[7] T. ERJAVEC, (1999), The ELAN Slovene-English Aligned Corpus, in *Proceedings of the Machine Translation Summit VII*, pages 349–357, Singapore. http://nl.ijs.si/elan/.

[8] T. ERJAVEC, (2001), Harmonised Morphosyntactic Tagging for Seven Languages and Orwell's 1984, in *6th Natural Language Processing Pacific Rim Symposium, NLPRS'01*, pages 487–492, Tokyo. http://nl.ijs.si/ME/V2/.

[9] W. GALE, K.W. CHURCH, (1993), A program for aligning sentences in bilingual corpora, *Computational Linguistics*, 19(1):75–102.

[10] C. GROVER, C. MATHESON, A. MIKHEEV, M. MOENS, (2000), LT TTT - A Flexible Tokenisation Tool, in *Second International Conference on Language Resources and Evaluation, LREC'00*. http://www.ltg.ed.ac.uk/software/ttt/.

[11] J. HAJIČ, (2000), Morphological Tagging: Data vs. Dictionaries, in *ANLP/NAACL 2000*, pages 94–101, Seattle.

[12] H. KUČERA, W. N. FRANCIS, (1967), *Computational Analysis of Present Day American English*, Brown University Press, Providence, Rhode Island.

[13] O. MASON, (1998), Qtag — a portable probabilistic tagger. http://www-clg.bham.ac.uk/QTAG/.

[14] T. MCENERY, A. WILSON, (1996), *Corpus Linguistics*. Edinburgh University Press.

[15] F.J. OCH, H. NEY, (2001), Statistical Machine Translation, in *Proceedings of the European Association for Machine Translation Workshop, EAMT'00*, pages 39–46, Ljubljana, Slovenia, May. http://nl.ijs.si/eamt00/.

[16] J. SINCLAIR, (1994), Corpus typology, EAGLES DOCUMENT EAG–CSG/IR–T1.1, Commission of the European Communities.

[17] C. M. SPERBERG-MCQUEEN, L. BURNARD, EDITORS, (2002), *Guidelines for Electronic Text Encoding and Interchange, The XML Version of the TEI Guidelines*, The TEI Consortium. http://www.tei-c.org/.

[18] J. TIEDEMANN, (1998), Extraction of translation equivalents from parallel corpora, in *Proceedings of the 11th Nordic Conference on Computational Linguistics*, Center for Sprogteknologi, Copenhagen. http://numerus.ling.uu.se/~corpora/plug/pwa/.

[19] H. VAN HALTEREN, EDITOR, (1999), *Syntactic Wordclass Tagging*. Kluwer Academic Publishers.

[20] Š. VINTAR, (1999), A Lexical Analysis of the ELAN Slovene-English Corpus, in *Proceedings of the Workshop on Language Technologies — Multilingual Aspects*, pages 63–70, Ljubljana, Slovenia. University of Ljubljana.

[21] W3C, (2000), Extensible Markup Language (XML) 1.0 (Second Edition), URL. http://www.w3.org/TR/2000/REC-xml-20001006.

Received: April, 2002
Revised: April, 2003
Accepted: May, 2003

Contact address:
Tomaž Erjavec
Dept. of Intelligent Systems
"Jožef Stefan" Institute
Jamova 39
SI-1000 Ljubljana
Slovenia
e-mail: tomaz.erjavec@ijs.si
www: http://nl.ijs.si/et/

TOMAŽ ERJAVEC is a Scientific Associate at the Dept. of Intelligent Systems, "Jožef Stefan" Institute. His research interests lie in the fields of computational linguistics and language technologies, with a focus on the Slovene language and multilingual applications.