Dictionary Writing System (DWS) + Corpus Query Package (CQP): The Case of *TshwaneLex*

Gilles-Maurice de Schryver, *Department of African Languages and Cultures, Ghent University, Ghent, Belgium; Xhosa Department, University of the Western Cape, Bellville, Republic of South Africa; and TshwaneDJe HLT, Pretoria, Republic of South Africa (gillesmaurice.deschryver@UGent.be),*
and
Guy De Pauw, *CNTS — Language Technology Group, University of Antwerp, Belgium; and School of Computing and Informatics, University of Nairobi, Kenya (guy.depauw@ua.ac.be)*

Abstract: In this article the integrated corpus query functionality of the dictionary compilation software TshwaneLex is analysed. Attention is given to the handling of both raw corpus data and annotated corpus data. With regard to the latter it is shown how, with a minimum of human effort, machine learning techniques can be employed to obtain part-of-speech tagged corpora that can be used for lexicographic purposes. All points are illustrated with data drawn from English and Northern Sotho. The tools and techniques themselves, however, are language-independent, and as such the encouraging outcomes of this study are far-reaching.

Keywords: LEXICOGRAPHY, DICTIONARY, SOFTWARE, DICTIONARY WRITING SYSTEM (DWS), CORPUS QUERY PACKAGE (CQP), TSHWANELEX, CORPUS, CORPUS ANNOTATION, PART-OF-SPEECH TAGGER (POS-TAGGER), MACHINE LEARNING, NORTHERN SOTHO (SESOTHO SA LEBOA)

Samenvatting: **Woordenboekaanmaaksysteem + corpusanalysepakket: een studie van *TshwaneLex*.** In dit artikel wordt het geïntegreerde corpusanalysepakket van het woordenboekaanmaaksysteem TshwaneLex geanalyseerd. Aandacht gaat zowel naar het verwerken van onbewerkte corpusdata als naar geannoteerde corpusdata. Wat het laatste betreft wordt aangetoond hoe, met een minimum aan intellectuele arbeid, automatische leertechnieken met succes kunnen worden ingezet om corpora voor lexicografische doeleinden aan te maken waarin de woordklassen expliciet worden vermeld. Alle stappen van de redenering worden geïllustreerd met gegevens uit het Engels en Noord-Sotho. De instrumenten en technieken zelf zijn echter allemaal taalonafhankelijk, waardoor de veelbelovende resultaten van deze studie verreikend zijn.

Sleutelwoorden: LEXICOGRAFIE, WOORDENBOEK, SOFTWARE, WOORDENBOEK-AANMAAKSYSTEEM, CORPUSANALYSEPAKKET, TSHWANELEX, CORPUS, CORPUSANNO-TATIE, WOORDKLASSETAGGER, AUTOMATISCHE LEERTECHNIEKEN, NOORD-SOTHO

1.   DWS & CQP: What are these?

Metalexicographers, dictionary makers and software engineers are constantly seeking better and faster ways to produce today's dictionaries. While software tools to assist the compilation of reference works are becoming ever more advanced, complete digital solutions are also increasingly put directly into the hands of the lexicographers themselves. Most professional dictionary houses that compile dictionaries in-house use (at least) two sets of tools: a dictionary writing system (DWS) on the one hand, and some sort of corpus query package (CQP) on the other. A team of local IT gurus will then typically ensure the transition of data between these two systems. It is very rare that the two systems are truly integrated, and if they are, a publisher-specific setup was designed. Until recently, no off-the-shelf packages — that anyone could acquire, for the compilation of any type of dictionary, for any (number of) language(s) — combining both a DWS and a CQP, were available. Since the release of the TshwaneLex Suite 3.0 in June 2007, however, this is now a reality.

Being a South African product, TshwaneLex is well-known in South Africa, as it is the system of choice for the eleven PanSALB-sponsored National Lexicography Units (NLUs), is in use at all the major local dictionary publishing houses (including OUP Southern Africa, Pharos and Macmillan SA), and copies of TshwaneLex are also found at virtually all South African universities. Its sister application TshwaneTerm, designed for the management of terminology, is popular with various government departments, including the South African Police Service (SAPS). Worldwide, there are currently over three hundred users of the TshwaneLex Suite (which now bundles TshwaneLex and TshwaneTerm).

In this article, the main issues that are relevant when integrating a CQP with a DWS are analysed. Specific attention goes to the current status quo at the NLUs. The strong and weak points uncovered are then used as a point of departure for the discussion of the built-in corpus functionality of TshwaneLex. A major leap forward is subsequently suggested in the form of the utilisation of integrated part-of-speech tagged corpora, and a method to obtain these for under-resourced languages is worked out for Northern Sotho as an illustration. In conclusion, a fully-working system is presented.

2.   DWS >< CQP: The status quo

Before discussing the setup at the South African NLUs, it is useful to briefly look at two of the most advanced setups in the world, namely that of the 'IMS Textcorpora and Lexicon Group' at the University of Stuttgart (Heid et al. 2007), and that of the users of the 'Sketch Engine' (Kilgarriff et al. 2007). Over the course of many years, the Stuttgart team has developed the following linguistic resources and tools:

—   Lexicons
—   Tools for automatic text analysis and corpus annotation
—   Retrieval and extraction tools
—   Linguistically annotated text corpora
—   Linguistic engineering standards

Whenever they have to compile new dictionaries and/or analyse, revise or update existing dictionaries, they employ these linguistic resources and tools, but none of these is integrated into their DWS, and new, ad hoc bits of code have to be written every time to link the two.

Arguably the most powerful CQP currently available is the Sketch Engine. In the words of Adam Kilgarriff: "The Sketch Engine (SkE, also known as Word Sketch Engine) is a Corpus Query System incorporating word sketches, grammatical relations, and a distributional thesaurus. A word sketch is a one-page, automatic, corpus-derived summary of a word's grammatical and collocational behaviour." The lexicographers thus have these one-page summaries in front of them, but still need to manually copy-and-paste data from the SkE into their DWS.

Less advanced, but still within the same framework, the South African lexicographers currently use WordSmith Tools (WST, Scott 2007) as their CQP.

Figure 1: The corpus query package (CQP) WordSmith Tools, as used at the Northern Sotho National Lexicography Unit (NLU)

Figure 1 shows a screenshot of the use of a corpus at the Northern Sotho NLU, where corpus lines for the verb *hwetša* 'find; get; obtain' are shown, sorted one to the left as primary sort, and one to the right as secondary sort. For each corpus line, the source can be seen in the rightmost column.

In this case, the lexicographers have selected corpus line 84 to illustrate one of the senses of *hwetša*, and a straightforward copy of the highlighted line in WST allows them to paste this data into TshwaneLex, following a toggle between programs. Corpus examples are only adapted minimally, and after clean-up and adaptation, the result is as seen in Figure 2, which is a screenshot of a search for *hwetša* in the current online version of this NLU's dictionary (Mojela et al. 2007).
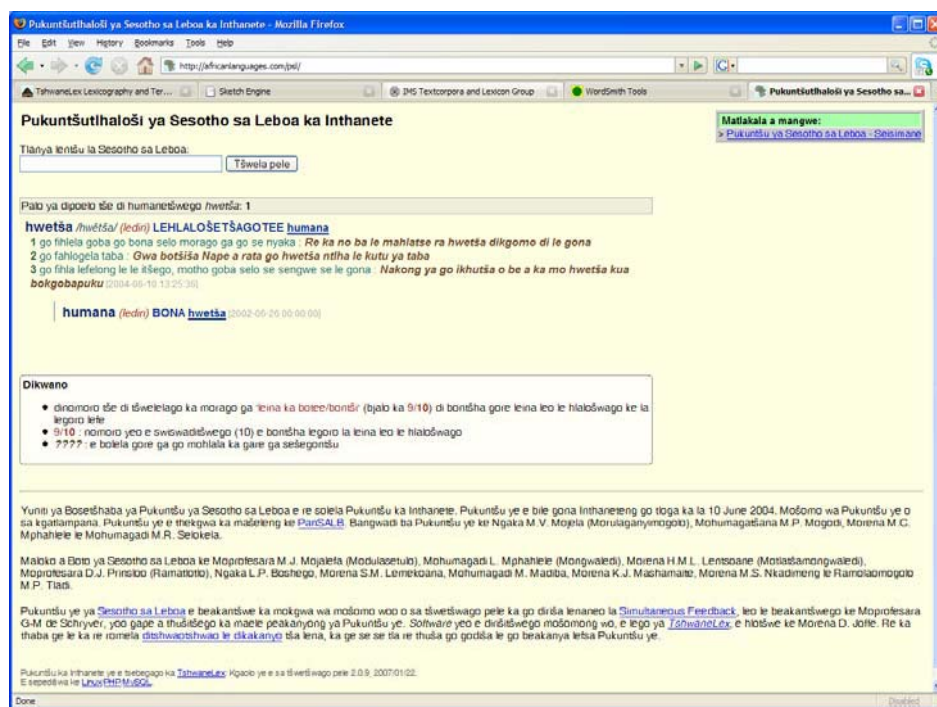
Figure 2: A search for *hwetša* in the online monolingual Northern Sotho dictionary [Note that the example for sense 2 corresponds to line 84 in Figure 1]

This setup whereby two separate pieces of software are used, on the one hand TshwaneLex and on the other WST, works well, but the fact that data has to be manually selected and copied over between programs is cumbersome, and there are also some encoding issues to be circumvented (for Northern Sotho, for instance, typing in the 'š' in the CQP is often a problem on standard South African keyboards).

3.   DWS + CQP: Wish list

What one would want to have is a single computational environment as dictionary compilation system, but without the loss of all the main advantages of using a CQP like WST. If one studies the functions that are needed on a daily basis for the compilation of a dictionary, then one arrives at the following minimum package:

— The ability to search for any word, combination of words, or multiple words simultaneously;
— The above in combination with wildcards;
— A function to easily sort the lines, according to the search node, as well as the word immediately to the left and immediately to the right;
— An indication of the occurrence (frequency) of the searched-for item;
— A function that allows for easy sampling; absolutely crucial for frequent words (such as function words);
— An indication of the source a particular corpus line comes from;
— An easy way to transfer data from the corpus analysis environment to the dictionary compilation environment.

All of the above are available in WST, although many clicks are needed for some of these basic functions that are needed in lexicography. (But then, WST is not primarily meant to assist lexicographers.)

Two further requirements may be added to the wish list. Firstly, a tool to encrypt the corpus data, so as to protect it from theft, is often desirable. Indeed, for all resource-scarce languages, a considerable amount of effort went into the creation of the now-available corpora, and while one wants all the dictionary compilers to be able to access the corpus data, one simultaneously does not want the data to be 'just out' there, 'in the open' so to say. Secondly, and implicit in the description of the South African setup so far, it would be much more meaningful to be able to work with corpora that are annotated for parts of speech, rather than having to work with the current 'raw' (and thus linguistically 'blind') corpora.

4.   DWS + CQP: TshwaneLex's F6 — Part 1: Working with raw corpus data

Figure 3 shows an overall picture of the TshwaneLex interface, here with data for a bidirectional Northern Sotho–English dictionary (in Linked View mode). For introductory, detailed as well as technical discussions of TshwaneLex, the reader is referred to the URL mentioned in the References of this article (Joffe et al. 2007). The focus here is on the built-in corpus functionality only, which is conveniently accessed by pressing the function key F6. In the interface, the corpus tool is therefore also the sixth tab of the attributes window (to be found at the bottom-right of each dictionary side). In the screenshot of Figure 3, the

Northern Sotho to English side of the dictionary can be seen on the left-hand side, while the English to Northern Sotho side may be seen on the right-hand side. Two different test corpora have been loaded, texts 1 to 4 for Northern Sotho, and four works by well-known authors for English.
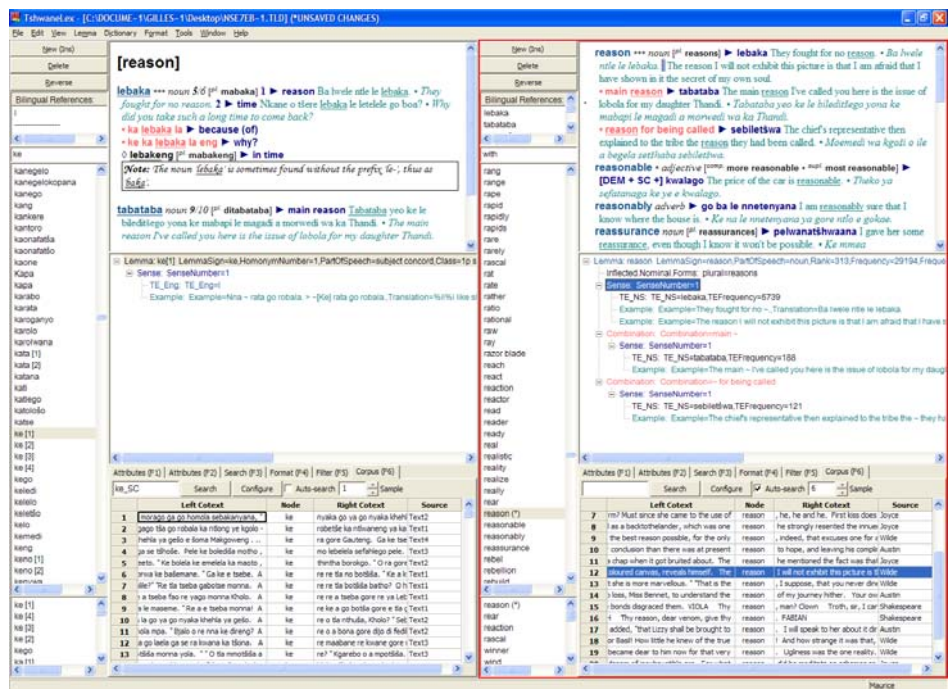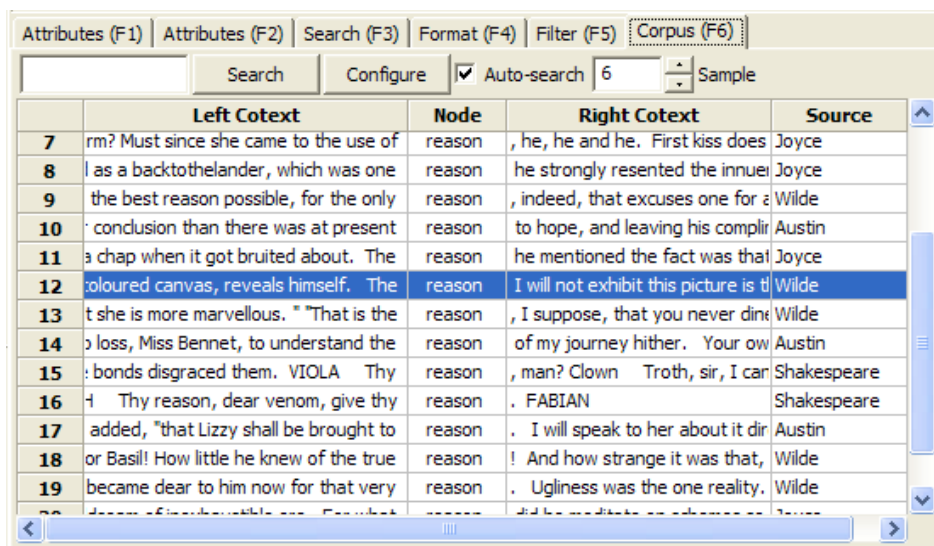

Figure 3: An overall picture of the TshwaneLex interface, with a focus on F6 for both sides (i.e. the built-in corpus tool)

All aspects of the 'minimum requirements' listed in the wish list of Section 3 have been implemented, and this seamlessly. Figure 4 zooms into the bottom-right corner of Figure 3, in this case the English corpus. In the search box one can type in any string of characters and press "Search" to initiate a search through the loaded corpus files. Regular expressions can also be used in this input box, which means that wild cards and more are easily available. To limit the number of results, sampling was performed in Figure 4, here with a random selection of each sixth line in the corpus. Actually, nothing was typed into the search box in Figure 4, as the "Auto-search" option had been ticked. This means that the corpus is searched automatically for the lemma sign of any article that one is working on. To further streamline the dictionary compilation process, one can simply stand on one (or more) corpus line(s) and press Ctrl+ F7: the highlighted sentence(s) will automatically be copied over to the sense one is working on. In Figure 4 this was done for line 12, the result of which can
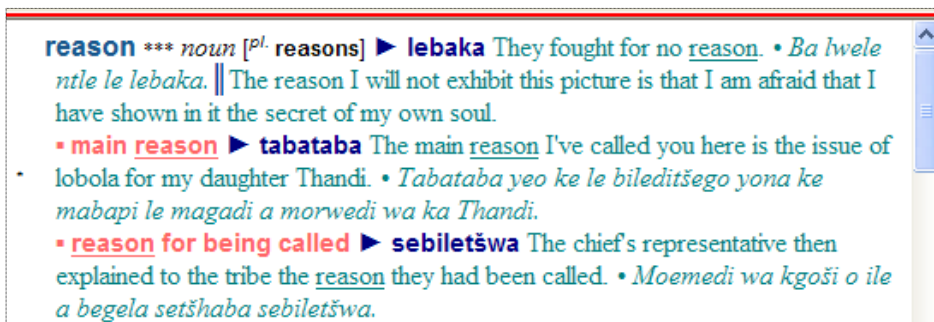
be seen in Figure 5. Compilation can then proceed as usual, which in the case of a bilingual dictionary means, amongst others, that one will translate the chosen example(s).

Figure 4: Using F6 (TshwaneLex's corpus tool) during dictionary compilation — Part 1: Selecting corpus lines for inclusion at the sense one is working on

Figure 5: Using F6 (TshwaneLex's corpus tool) during dictionary compilation — Part 2: Automatic transfer of entire sentences to the sense one is working on

Pressing the "Configure" button under F6 launches the dialog shown in Figure 6. From it, it can be deduced that functionality has indeed been included that allows for corpus data to be encrypted. TshwaneLex remembers corpus settings, so that corpora are automatically loaded whenever one starts work. With this dialog, the files that make up the corpora can also easily be changed.
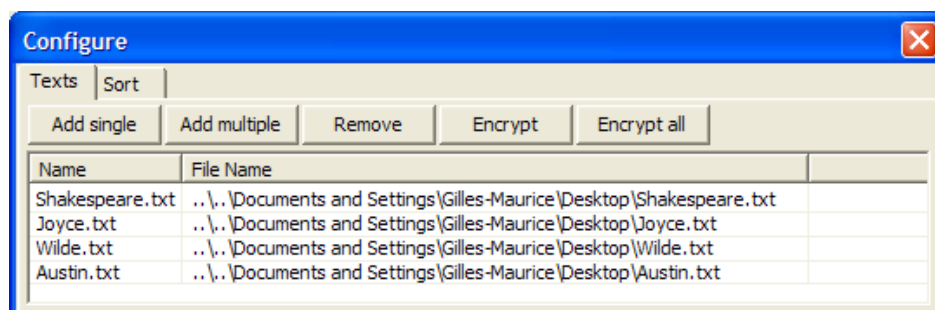
Figure 6: The F6 (TshwaneLex's corpus tool) dialog

What is still outstanding from the wish list, then, is the possibility to handle corpora that are marked up for parts of speech. There are two aspects that need attention here. Firstly there is the issue to annotate a corpus, and secondly there is the issue of how to handle such annotated corpora in TshwaneLex. Given corpus annotation has never been pursued in full in South Africa (nor for any African language for that matter, bare Swahili, cf. Hurskainen 1999), we feel that the lexicographic community would gain much from a presentation that shows how this can be accomplished with a minimum of effort. The next section illustrates this.

5.   Intermezzo: Part-of-speech tagging under-resourced languages

Research in the field of Natural Language Processing (NLP) is inconceivable without the availability of digital resources developed by the lexicographic community. All too often though, this natural and apparent link between the two related communities is ignored and true collaborative research efforts are few and far between. In the experiments described in this section, we show how one can directly exploit lexical resources in a cost-effective development of language technology applications. As a case study, we describe how dictionary data can be used to create an accurate part-of-speech tagger for Northern Sotho with a minimal amount of manual effort. Once such a tagger is available, *better* dictionaries can be made *faster*. One thus comes full circle.

5.1   Data-driven part-of-speech tagging

In recent years, part-of-speech (POS) taggers have become established language technology tools for most of the major Indo-European languages, such as English, French, Dutch and German. POS-taggers are an essential component for many commercial NLP applications, such as machine translation or text mining, serving as one of the most valuable disambiguation steps in the processing

chain. Assigning the parts of speech to the words in a sentence indeed provides invaluable morphological and syntactic information that percolates to all levels of linguistic analysis.

While historically POS-taggers have most often been constructed by hand-crafting rule-based systems, the last two decades have seen a definite shift to more robust, data-driven approaches. Rather than capturing the solution to the POS-tagging problem in a set of manually designed rules, these data-driven methods exploit large, manually annotated corpora, to train statistical methods and machine learning approaches that can automatically induce the solution to the disambiguation problem.

The advantages of the data-driven approach are many. First and foremost, it is inherently language independent: the construction of a POS-tagger for a given language or domain is relatively straightforward, provided there is anno-tated data available for it. This contrasts the construction of rule-based POS-taggers, which is typically a labour-intensive process, requiring the involve-ment of linguistic experts and producing taggers that are not portable to other languages. Furthermore, data-driven taggers have been shown to outperform hand-crafted taggers on most types of language data, not only in terms of development costs, but also in terms of coverage and robustness (Kupiec 1992).

While the advantages of data-driven tagging are obvious, they are re-stricted by the fact that they require annotated data for a given language. While annotated resources are abundantly available for languages such as English and French, this is not the case for the majority of the world's languages. Con-sequently, this impedes the automatic development of language technology tools for the under-resourced languages.

Many researchers assume that data-driven approaches to NLP require the availability of hundreds of thousands of annotated tokens. A relatively new area of NLP research is now investigating how more modest-sized data sets can also bootstrap language technology for resource-scarce languages. In this section, we describe how we can build a relatively accurate data-driven POS-tagger for the resource-scarce language of Northern Sotho on the basis of a manually annotated corpus of only 10 000 words.

5.2   From lexical database to annotated corpus

Typically, the construction of a POS-tagged corpus requires a carefully de-signed tagging protocol, which involves the definition of a well-rounded tag set and meticulously stipulated tagging guidelines. While this is a valuable exercise in its own right, we hypothesise that most aspects of the protocol can be dynamically constructed and refined during actual annotation, provided there is a strong lexical backbone that can bootstrap the annotation process.

The TshwaneDJe HLT Northern Sotho lexical database provides exactly this backbone, since it includes for each lemma its frequency information and its possible POSs. From this database we can extract the top 5 000 words and

list the POS-tags associated with those words. The typically Zipfian distribution of language tokens in a corpus ensures the tag set extracted from the lexical database has sufficiently large coverage. Furthermore, the annotation environment easily allows minor adjustments to be made to this tag set, should it be unable to adequately cover some tokens. This on-the-fly approach enables the organic construction of a consistent tag set, grounded in linguistic, corpus-based evidence. The complete Northern Sotho tag set induced from the lexical database and further refined during annotation, is shown in Addendum 1.

The annotators were asked to tag a corpus of literary texts. The data was tokenised and imported into a spreadsheet, listing one word per row. The TshwaneDJe HLT Northern Sotho lexical database then further allowed us to guide and speed up the manual annotation efforts: for over 90% of the words in the corpus, the lexical database can provide the annotator with a 'shortlist' of possible tags for each word, thereby minimising manual input.

The procedure is illustrated in Figures 7 and 8, which are screenshots of the annotation environment in the spreadsheet. Column B in Figure 7 contains the word to be tagged, while columns D, E, F, etc. provide the possible tags for the word, as retrieved from the lexical database, and ordered according to overall frequency of occurrence. Words with more than one possible tag, thus ambiguous words, are highlighted in green, indicating the annotator needs to disambiguate this word. Words associated with just one tag are not highlighted, although the annotator can still change the tag, should it not be correct.

Figure 7: Spreadsheet containing the initial material for the annotator

Disambiguation is done as follows: if the tag in Column D is correct, nothing needs to be done and the annotator can proceed to the next word. If it is incorrect, the cell needs to be deleted and the annotator checks the tag in Column E. If this tag is correct, the annotator can move on to the next word. If it is not, the cell is deleted and Column F is considered and so on and so forth. If the correct tag is not provided by the lexical database in any of the columns, the annotator can use the drop-down box in Column C, which contains all the tags of the tag set (see Addendum 1). Post-processing will only retrieve the leftmost tag, which

means annotators need not spend time deleting incorrect tags to the right of the correct tag.

Unknown words are highlighted in grey. These are words that fall beyond the scope of the top 5 000 words in the lexical database. For these words, the annotator has to choose a tag from the drop-down box in Column C, or, if quicker, he/she can simply type the required tag in the empty cell in Column C. 'New' words that are likely to recur may be tagged in one go: once the annotator has provided the correct POS-tag for such a new word, he/she may sort Column B, copy-and-paste the tag for other occurrences of the same word, and then restore the original order, by sorting the data back using the indices provided in Column A. Once tagged, the sample from Figure 7 is as shown in Figure 8.

	A	B	C	D	E	F	G	H	I	J
6144	6145	<utt>								
6145	6146	**Ga**				NEG	PC			
6146	6147	**ke**		SC	COPp	AUX_V	AGTp			
6147	6148	**ye**			V					
6148	6149	**go**						CP15		
6149	6150	šutha	V							
6150	6151	tabeng		N+ng						
6151	6152	**ya**				PC				
6152	6153	**ka**						PRO_poss	POT	INSp
6153	6154	**ya**				PC				
6154	6155	gore		CONJ						
6155	6156	**ga**				NEG	PC			
6156	6157	**re**		SC	OC	V				
6157	6158	reke		V						
6158	6159	dijo		N						
6159	6160	.		Punc						
6160	6161	</utt>								

Figure 8: Spreadsheet containing the material annotated by the annotator

The contents of the drop-down box, that is the tag list, are defined on a separate worksheet. This tag list can be dynamically adjusted, should a new tag need to be created for a particular word. Spreadsheet functionality allows for the drop-down box in the annotation worksheets to automatically reflect the adjustments.

Despite the availability of dedicated annotation tools, using a spreadsheet for this annotation task has some significant advantages. Installation is trivial and most computer-literate users are familiar with this type of software, so that the learning curve for the annotators is favourable. While annotation is an unlikely use of a spreadsheet, the cell-based approach can significantly speed up an annotation task such as POS-tagging.

Practical limitations prevented us from annotating the entire dataset. Restricted to a total annotation time of a mere 10 person-hours, the design of the annotation environment just described nevertheless maximised the amount of annotated data. After post-processing the data, we had a manually tagged corpus of more than 10 000 words, in a format ready to be used as training material for a data-driven tagger. A sample of this is shown in Figure 9.
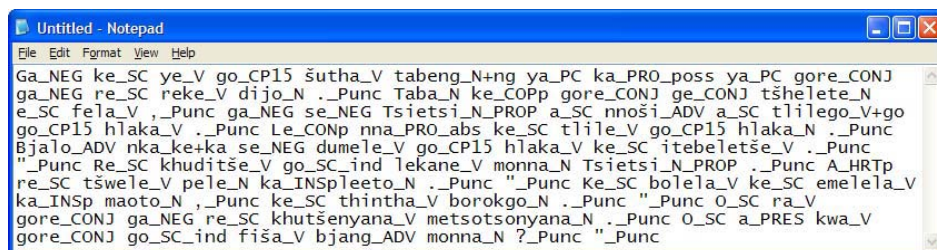
```
 Untitled - Notepad
File Edit Format View Help
Ga_NEG ke_SC ye_V go_CP15 šutha_V tabeng_N+ng ya_PC ka_PRO_poss ya_PC gore_CONJ
ga_NEG re_SC reke_V dijo_N ._Punc Taba_N ke_COPp gore_CONJ ge_CONJ tšhelete_N
e_SC fela_V ,_Punc ga_NEG se_NEG Tsietsi_N_PROP a_SC nnoši_ADV a_SC tlilego_V+go
go_CP15 hlaka_V ._Punc Le_CONp nna_PRO_abs ke_SC tlile_V go_CP15 hlaka_N ._Punc
Bjalo_ADV nka_ke+ka se_NEG dumele_V go_CP15 hlaka_V ke_SC itebeletše_V ._Punc
"_Punc Re_SC khuditše_V go_SC_ind lekane_V monna_N Tsietsi_N_PROP ._Punc A_HRTp
re_SC tšwele_V pele_N ka_INSpleeto_N ._Punc "_Punc Ke_SC bolela_V ke_SC emelela_V
ka_INSp maoto_N ,_Punc ke_SC thintha_V borokgo_N ._Punc "_Punc O_SC ra_V
gore_CONJ ga_NEG re_SC khutšenyana_V metsotsonyana_N ._Punc O_SC a_PRES kwa_V
gore_CONJ go_SC_ind fiša_V bjang_ADV monna_N ?_Punc "_Punc
```

Figure 9: UTF-8 encoded text-only version of the POS-tagged corpus

While a 10 000-word-tagged corpus is indeed modest, compared to the million-word corpora available for English (Marcus et al. 1993), the experiments described in the next section will show that even a small annotated data set can yield an accurate data-driven POS tagger.

5.3   Maximum Entropy Tagging

The general idea behind data-driven POS-tagging is that annotated data, as displayed in Figure 9, inherently encode the solution to the disambiguation problem of POS-tagging. Rather than having a linguist explicitly specify that a word is to be tagged with some POS-tag in a particular context, the data-driven approach tries to automatically induce this type of disambiguation from annotated data.

In recent years, a large number of data-driven POS-tagging toolkits have become freely available, such as MBT (Daelemans et al. 2003), MXPOST (Ratnaparkhi 1996), TnT (Brants 2000) and SVMTool (Giménez and Màrquez 2004). Given the availability of annotated data for some language, all of these tools become a viable option to construct a POS-tagger (De Pauw et al. 2006). While there are significant differences in the way these respective data-driven methods implement the solution to the problem, they all have in common that they try to 'mimic' the behaviour of the manual annotators, by trying to capture linguistic patterns using statistical and/or symbolic means.

The particular approach used in the experiments described in this article is based on the machine learning technique of Maximum Entropy Learning (Berger et al. 1996). This technique has previously shown to obtain state-of-the-art results for many languages, including Swahili. Rather than the stock maximum entropy tagger, MXPOST (Ratnaparkhi 1996), we used a self-constructed POS-tagger, called MaxTag, which acts as a front-end to the general machine learning package Maxent (Le 2004). It has the advantage of providing fast processing times and being more robust in handling morphologically rich languages.

MaxTag takes as its input POS-tagged data and extracts for each word in the corpus a number of features that are possibly relevant to the disambiguation problem. Just as in most other data-driven taggers, the features considered by MaxTag are a mixture of contextual and orthographic information. Ortho-

graphic features try to capture the morphological aspects of the word to be tagged, while contextual features describe the syntactic context of the word within the sentence.

Contextual features[1]

— **FW**: the word itself
— **FT**: a single token representing all possible tags for the focus word
— **W-1**: the previous word
— **W+1**: the next word
— **T-1**: the preceding tag
— **T+1**: a single token representing all possible tags for the next word

Orthographic features

— **P1**: the first letter of the word
— **P2**: the first two letters of the word
— **P3**: the first three letters of the word
— **S1**: the last letter of the word
— **S2**: the last two letters of the word
— **S3**: the last three letters of the word
— **CAP**: the presence of a capital

For our tagging experiments, we recorded a limited context, considering only one word before and after the word to be tagged. While one might be tempted to record a substantially larger context size, there is a trade-off: the larger the contexts that are being considered, the less linguistic evidence one can find in the data, making it harder for the data-driven taggers to generalise over the data. Particularly on small data sets such as the one for Northern Sotho, the sparse data problem restricts us to using a more limited context. We illustrate the extraction of features for the following example sentence:

Ke_SC a_PRES eletša_V._Punc

For each word in the corpus an 'instance' is extracted that contains both contextual and orthographic features. Each instance is then associated with the POS-tag for that word. For the example sentence above, this gives us the following four instances shown in Table 1.

Once MaxTag extracts one instance for each word in the annotated data, the Maxent machine learner consequently processes all of these instances. It observes the association of features with a particular tag and tries to optimally estimate the 'predictive' power of a particular feature. It will for example try to infer from instance (1) that a focus word (FW) 'Ke' at the beginning of a sentence (cf. "W-1=#") is likely to be associated with the tag "SC".

Table 1: Four 'instances' for the sentence "*Ke_SC a_PRES eletša_V ._Punc*"

	Instance	Tag
(1)	['W-1=#', 'T-1=#', **'FW=Ke'**, 'FT=SC_COPp', 'W+1=a', 'T+1=SC_PRES_PC_DEM_OC_HRTp', 'P1=K', 'S1=e', 'P2=Ke', 'S2=Ke', 'CAP']	**SC**
(2)	['W-1=Ke', 'T-1=SC', **'FW=a'**, 'FT=SC_PRES_PC_DEM_OC_HRTp', 'W+1=eletša', 'T+1=V', 'P1=a', 'S1=a']	**PRES**
(3)	['W-1=a', 'T-1=PRES', **'FW=eletša'**, 'FT=V', 'W+1=.', 'T+1=Punc', 'P1=e', 'S1=a', 'P2=el', 'S2=ša', 'P3=ele', 'S3=tša']	**V**
(4)	['W-1=eletša', 'T-1=V', **'FW=.'**, 'FT=Punc', 'W+1=#', 'T+1=#', 'P1=.', 'S1=.']	**Punc**

On the basis of these observations, the Maxent machine learner will try to construct a statistical model that optimally relates features to classes. It does this in an iterative movement, by attempting variations of the parameters in the statistical model and evaluating the tagging accuracy of the respective models on the annotated data. When the most optimal settings are eventually found, the resulting model will be able to tag new, previously unseen data, thereby roughly mimicking the tagging behaviour of the original annotators.

While tagging new data, both contextual and orthographic information is taken into account. In our evaluation, we will make a distinction between tagging accuracy on known words versus unknown words. The former are words that occur in the annotated data, and for which the tagger has observed some linguistic evidence. The latter are previously unseen words, that do not occur in the lexical database, nor in the annotated corpus. For these, orthographic features prove particularly useful, as they attempt to encode morphological information.

5.4   Experimental results

To evaluate the performance of the tagger, we performed a ten-fold cross validation experiment. This means we randomly distribute the sentences of the full data set over ten partitions and run ten experiments: in each experiment a different partition is used as the testing set, while the other nine partitions are used as the data to train the data-driven tagger. By comparing the tags output by the tagger to those in the original annotation, we can estimate the accuracy of the tagger on previously unseen data. Doing this ten times further reduces the risk of accidentally evaluating on an artificially (un)favourable training-test set partition, providing trustworthy experimental results.

We compare the results of the MaxTag method to that of a baseline. The baseline method is defined as the simplest solution to the problem. For known

words, the baseline method uses a unigram approach, always selecting the tag that is most frequently associated with a particular word in the training data. Unknown words are invariably tagged as nouns by the baseline method, since that is the tag most often associated with low-frequency words. The baseline method achieves a reasonable, but unspectacular tagging accuracy on unseen data, as can be seen from Table 2. Particularly the score for unknown words (about 8% of the test data) is underwhelming.

Table 2: Accuracy scores for the baseline method and MaxTag (all values in %)

	Known	Unknown	Total
Baseline	75.8	35.1	73.5
MaxTag	95.1	78.9	93.5

The MaxTag method achieves a significant increase on all accounts, particularly for unknown words. This amounts to a total tagging accuracy of 93.5%. Given the restricted size of the annotated corpus, this result is very encouraging, especially since the baseline indicates that the POS-tagging problem for Northern Sotho is far from trivial.[2]

5.5   Towards large POS-tagged corpora

The experiments described in this section have resulted in a POS-tagger that is able to tag unseen data with an accuracy of almost 94%. This opens up the possibility of semi-automatic annotation, where the data-driven tagger provides a first annotation, which is consequently checked by human annotators. This not only speeds up corpus annotation itself, but also ensures more consistent annotation throughout the corpus. This is an especially important point when working in a multi-annotator environment. Subsequent annotation efforts will undoubtedly greatly benefit from the Northern Sotho tagger developed in the context of this article.

While 93.5% tagging accuracy is an encouraging result, this is still not up to par with data-driven taggers for English (Van Halteren et al. 2001) or Swahili (De Pauw et al. 2006), achieving near-human type of tagging accuracies. This is undoubtedly due to the limited size of the annotated corpus.

We therefore ran some learning curve experiments to investigate the effect of the quantity of data on the quality of the resulting tagger. To this end, we isolated a single evaluation set and used the remaining data to incrementally train POS-taggers. We started out with a POS-tagger trained on just 1/10 of the available training data (roughly 1 000 words) and added 1/10 of the training data in each subsequent experiment. The result of this experiment can be found in the learning curve graph, displayed in Figure 10.[3] The graph shows that the learning curve is still linear and that we can still gain quite a bit of tagging accuracy by collecting more annotated data. Future research will therefore con-

centrate on the semi-automatic development of new annotated data. The more than encouraging experimental results show that the Northern Sotho version of MaxTag can provide an invaluable tool in this endeavour.
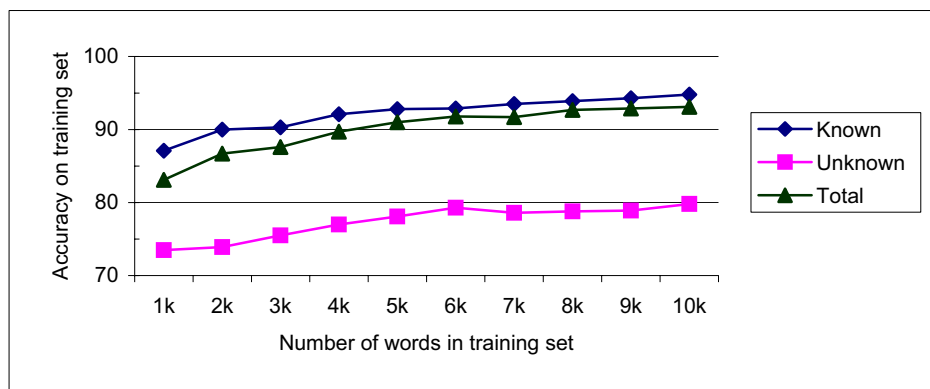

Figure 10: Graph for learning curve experiments

6.   DWS + CQP: TshwaneLex's F6 — Part 2: Working with POS-tagged corpus data

From Section 5 we see that POS-tagged corpora are now within reach for both well-resourced and under-resourced languages. In this last section we briefly show how working with POS-tagged corpus data can significantly improve the use of corpora during dictionary compilation. The most obvious, and most straightforward use is that automated searches can now take the part of speech of the article one is working on into account. In Northern Sotho, there are for example four different homonymous *fela*, one being an adverb ('only; just'), another a conjunction ('but'), another a verb ('finish; come to an end'), and lastly an auxiliary verb ('usually; continuously'). With the "Auto-search" option enabled, as illustrated in Figure 11, a lexicographer working on the adverbial form will only be presented with those cases in the corpus tagged as adverbs, greatly simplifying the task to select an appropriate example. This is especially true for function words, where manual disambiguation between homonyms during dictionary compilation can be very taxing. In general, patterns are also much quicker to spot. So a search for "nyaka_AUX_V" will immediately reveal that this form is virtually always followed by a main verb in the infinitive, which may prompt the lexicographer to add a Usage Note in this regard.

On a next level, TshwaneLex's corpus tool also allows lexicographers to search for particular 'word-POS pairs', as can be seen in Figure 12 (which is an enlargement of the bottom-left corner of Figure 3).

Needless to say, all of these may be combined with regular expressions during a search, which means that one can now truly find what one is looking

for on the one hand, but also that one can now also more easily pinpoint linguistically-informed patterns in the data, which can then be described lexico-graphically.

Figure 11: When working with POS-tagged corpora, an "Auto-search" automatically takes the part of speech into account

Figure 12: A search for all instances of the subject concord *ke* only ("ke_SC"), and not of the copulative particle *ke*, agentative particle *ke*, or auxiliary verb *ke*

7.   Conclusion

In this article it has been shown that the seamless integration of corpus query functionality within a dictionary compilation environment has now become a reality. The corpora may either be plain texts, or texts annotated for part of speech (POS). A method using minimal datasets was described with which such POS-tagged corpora may swiftly be obtained for resource-scarce languages. As a rsult, corpora can henceforth be queried more intelligently during dictionary compilation — and this for *any* language.

Acknowledgements

Endnotes

1.   The T-1 feature describes a single tag, while T+1 encodes all possible tags for the W+1 word. This is a consequence of the left-to-right tagging process. When disambiguating the focus

word, the right-hand side is not disambiguated yet, while the left-hand side is.
2.   We also attempted the data-driven taggers MBT, TnT and MXPOST on this data set, but none of them were able to outperform MaxTag.
3.   For unknown words there is a sudden decrease in tagging accuracy in the seventh partition, followed by a slow restoration of the linear learning curve. Error analysis indicates that this is due to the fact that the sixth partition contains a significant amount of idiosyncratic 'unknown words', disturbing the training phase when evaluating on the seventh partition.

References

Berger, A.L., S. Della Pietra and V.J. Della Pietra. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics* 22(1): 39-71.

Brants, T. 2000. TnT — A Statistical Part-of-Speech Tagger. *Proceedings of the Sixth Conference on Applied Natural Language Processing (ANLP 2000), Seattle, USA*: 224-231.

Daelemans, W., J. Zavrel, A. van den Bosch and K. van der Sloot. 2003. *MBT: Memory Based Tagger, Version 2.0, Reference Guide*. Technical Report Series 03-13. Tilburg: ILK Research Group.

De Pauw, G., G.-M. de Schryver and P.W. Wagacha. 2006. Data-Driven Part-of-Speech Tagging of Kiswahili. Sojka, P., I. Kopecek and K. Pala (Eds.). 2006. *Text, Speech and Dialogue, 9th International Conference, TSD 2006, Brno, Czech Republic, September 11–15, 2006, Proceedings*: 197-204. Lecture Notes in Artificial Intelligence (LNAI), subseries of Lecture Notes in Computer Science (LNCS), volume nr. 4188. Berlin: Springer-Verlag.

Giménez, J. and L. Màrquez. 2004. SVMTool: A General POS Tagger Generator Based on Support Vector Machines. *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004), Lisbon, Portugal*: 43-46.

Heid, U. et al. 2007. *The IMS Textcorpora and Lexicon Group* [online]. Available: http://www.ims.uni-stuttgart.de/projekte/corplex/.

Hurskainen, A. 1999. SALAMA. Swahili Language Manager. *Nordic Journal of African Studies* 8(2): 139-157.

Joffe, D. et al. 2007. *TshwaneLex Suite* [online]. Available: http://tshwanedje.com/tshwanelex/.

Kilgarriff, A. et al. 2007. *Sketch Engine* [online]. Available: http://www.sketchengine.co.uk/.

Kupiec, J. 1992. Robust Part-of-Speech Tagging Using a Hidden Markov Model. *Computer Speech and Language* 6: 225-242.

Le, Z. 2004. *Maximum Entropy Modeling Toolkit for Python and C++* (Technical Report) [online]. Available: http://homepages.inf.ed.ac.uk/s0450736/maxent_toolkit.html.

Marcus, M., B. Santorini and M. Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics* 19(2): 313-330.

Mojela, M.V. et al. 2007. *Pukuntšutlhaloši ya Sesotho sa Leboa ka Inthanete* (Explanatory Sesotho sa Leboa Dictionary on the Internet) [online]. Available: http://africanlanguages.com/psl/.

Ratnaparkhi, A. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Somerset, USA*: 133-142.

Scott, M. 2007. *WordSmith Tools* [online]. Available: http://www.lexically.net/wordsmith/.

Van Halteren, H., J. Zavrel and W. Daelemans. 2001. Improving Accuracy in Word Class Tagging through Combination of Machine Learning Systems. *Computational Linguistics* 27(2): 199-230.

Addendum 1:   Part-of-speech tags used for the Northern Sotho tagging experiment described in Section 5

POS-tag	Meaning	Example
ADJ	adjective	… nako tše **dintši** o a tseba.
ADV	adverb	**Gape** kgekolo yela yona …
AGTp	agentive particle ('ke')	Wena o dirwa **ke** gore o kwana …
ASP	aspectual prefix	Ba be ba **sa** hwile ka boroko.
ASP+go	aspectual prefix + relative marker ('go')	… **tšogo** …
AUX_V	auxiliary verb	Ba **be** ba sa hwile ka boroko …
AUX_V+go	auxiliary verb + relative marker ('go')	Ga go na yo a **ilego** a bona …
CONJ	conjunction	"Ga se ba go kwa **ge** o etšwa?"
CONp	connective particle ('le')	Ke realo **le** nna ke bolela ka …
COP_V	copulative verb	Ga go **na** yo a nkwelego ge ke …
COP_V+go	copulative verb + relative marker ('go' OR 'ng')	… yona ye e **lego** kgauswi …
COPp	copulative particle ('ke', 'ga se')	Taba **ke** gore ke di boletše …
CP15	infinitive prefix cl. 15 ('go')	… tšeo tša gago tša **go** robala …
DEM	demonstrative	Gape kgekolo **yela** yona e a …
DEM-COP	demonstrative copulative	Yena **šo**!
ENUM	enumerative	… wa di bolela ka nako e **tee**.
EXCL	exclamation	O reng o ntšhoša ngwana **tena**?
FUT	future morpheme ('tlo', 'tla')	… re **tla** ngwathagana tše tša …
FUT+go	future morpheme ('tlo') + relative marker ('go')	… **tlogo**…
HRTp	hortative particle	**A** re goge.
IDEO	ideophone	Ke no re ge ke re **phaphara** …
INSp	instrumental particle ('ka')	… o mpotšiša **ka** sehebehebe.
INTERJ	interjection	**Ai**!
ka+eng	instrumental particle ('ka') + question word ('eng')	O tla nkiša **kang** ka gore le …
ke+eng	copulative particle ('ke') + question word ('eng')	Molato **keng**?
ke+ka	first person singular ('ke') + potential morpheme ('ka')	Ke bona **nka** kgona go thuša …
LOCp	locative particle	… ke robetše **ka** ntlwaneng ya …
MPG	merged particle group	… fela ga re na taba **nayo**.
N	noun	… tlhokomelo e le ka **lebaka** la …
N_PROP	proper name	**Seboko** o mpotšiša ka …
N+ng	noun + locative marker ('ng')	Ke realo **mafelelong** ke …
NEG	negative morpheme ('ga', 'sa', 'se', 'ga se')	**Ga se** ba go kwa ge o etšwa?
NEUT	neutral subject marker ('e')	**E** sa le gosasa.
OC	object concord	"Ga se ba **go** kwa ge o etšwa?"
OC1sg+V	object concord 1p sg + verb	Bjale o nyaka go **ntlhanogela**?
OCcl1+V	object concord cl. 1 + verb	Ke a **mmotšiša** ke sa na le …
PAST	past tense morpheme ('a')	… ba **a** phetha modiro wa bona.
PC	possessive concord	… **sa** ka godimo **ga** letsogo …
POSS	possessive	Diaparo **tšagwe** di hlwekile.
POT	potential morpheme ('ka')	… ga go na seo o **ka** se dirago …
POT+go	potential morpheme ('ka') + relative marker ('go')	Go tla ba le **kago** ya mašole …
PRES	present tense morpheme ('a')	… kgekolo yela yona e **a** kwa.
PRO_abs	absolute pronoun	Ke realo le **nna** ke bolela ka …

POS-tag	Meaning	Example
PRO_comm_poss	communal possessive pronoun	… sa khehla ya **geno**?
PRO_poss	possessive pronoun	… seatla sa **ka** godimo ga …
PRO_quant	quantitative pronoun	… re aparetšwe ke seetša **gohle**.
Punc	punctuation	Wena o re ke swereng ka mo**?**
QSTp	question particle	… le eng **na**?
SC	subject concord	"**Ke** šikinya hlogo."
SC_ind	indefinite subject concord ('go')	… ka fao **go** ra gore ga se ba …
TMPp	temporal particle ('ka')	… tswalelwa **ka** Labohlano.
unknown	unknown POS	
V	verb	"Ga se ba go **kwa** ge o **etšwa**?"
V+eng	verb + noun ('eng')	Wena o re ke **swereng** ka mo?
V+go	verb + relative marker ('go' OR 'ng')	… ke o **swerego** ka seatla sa …
V+ng	verb + plural marker ('ng')	… a sa tsebe gore o **bolelang** …