

UniSpell and Lexical Databases

by Tom Laskowske

*A paper presented at The Asia Lexicography Conference,
Chiangmai, Thailand
24th – 26th May, 2004*

The need for a vernacular spell checker

Before spell checkers ever came along, where did you go to check the correct spelling of a word? A dictionary! Those who are compiling dictionaries would do well to keep in mind that the dictionary is often considered to be the highest standard when it comes to correctly spelled words. So one of the lexicographer's aims should be to have no spelling inconsistencies in the lexical database.

People who produce dictionaries and other literature on a computer can help to ensure the quality of their work by using a spell checker. However, spell checkers are language specific, and those that are available commercially are only for a few of the major languages of the world. So what can people do who produce literature in a vernacular language? UniSpell was created to meet the need of spell checking such languages.

The UniSpell program

UniSpell comes packaged with an "open source" text editor called AbiWord. In order to use UniSpell, however, one needs to learn how to use only a few of the functions displayed in the AbiWord menu.

Figure 1 is a screen shot of the UniSpell program. On the left side of the screen you can see a text file open. The language is Indonesian. On the right side of the screen is a wordlist that UniSpell has loaded to use for spell checking. We begin to check spelling by clicking Spelling > Run Spell Check (highlighted) on the menu bar. (Do not click the Tools > Spelling > Check Spelling button to check vernacular languages, because that accesses Ispell, a different spell checker that comes with AbiWord.)

Run spell check on an Indonesian file

A spell check of the sample text leads us to take the following actions:

1. Add *terjal* to the default user file Indonesian.usr
2. Correct spelling of *murupakan* to *merupakan*
3. Change caps and add *kabupaten* to Indonesian.usr
4. Correct spelling of *ketingian* to *ketinggian* and add to Indonesian.usr
5. Add *Sulawesi* to IndoNames.usr
6. Add *kalekaju* to Regional.usr

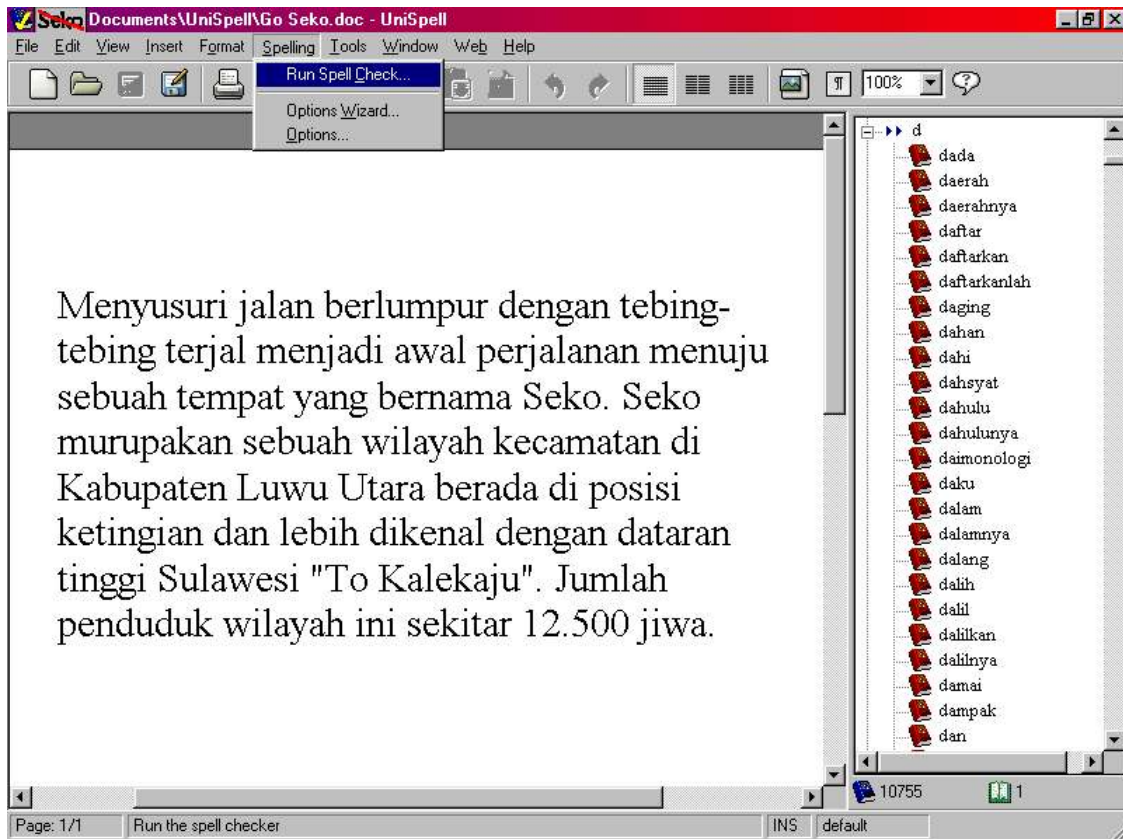


Figure 1 UniSpell screen

Find one or two of the words you have just added to the wordlist, for example "Sulawesi" and "terjal" from the sample text. Note the appearance of the icons. They are user file icons, in contrast to dictionary file icons. When you are confident that a user file is ready to be stored, merge it into a UniSpell dictionary file.

Create a vernacular spell check wordlist

A very valuable feature of the UniSpell program is that it is not limited to one language. You can create a different profile for each language you want to check. UniSpell remembers the last profile you used when you exited the program.

Let's look at Seko Padang, a language that's a little more exotic. Note the way the letters a and à show up side by side in the wordlist headings (Figure 2). They are supposed to do that. This and other "sort order" and alphabet issues are managed in the Spelling > Options > Dictionaries > Collating setting for the main dictionary.

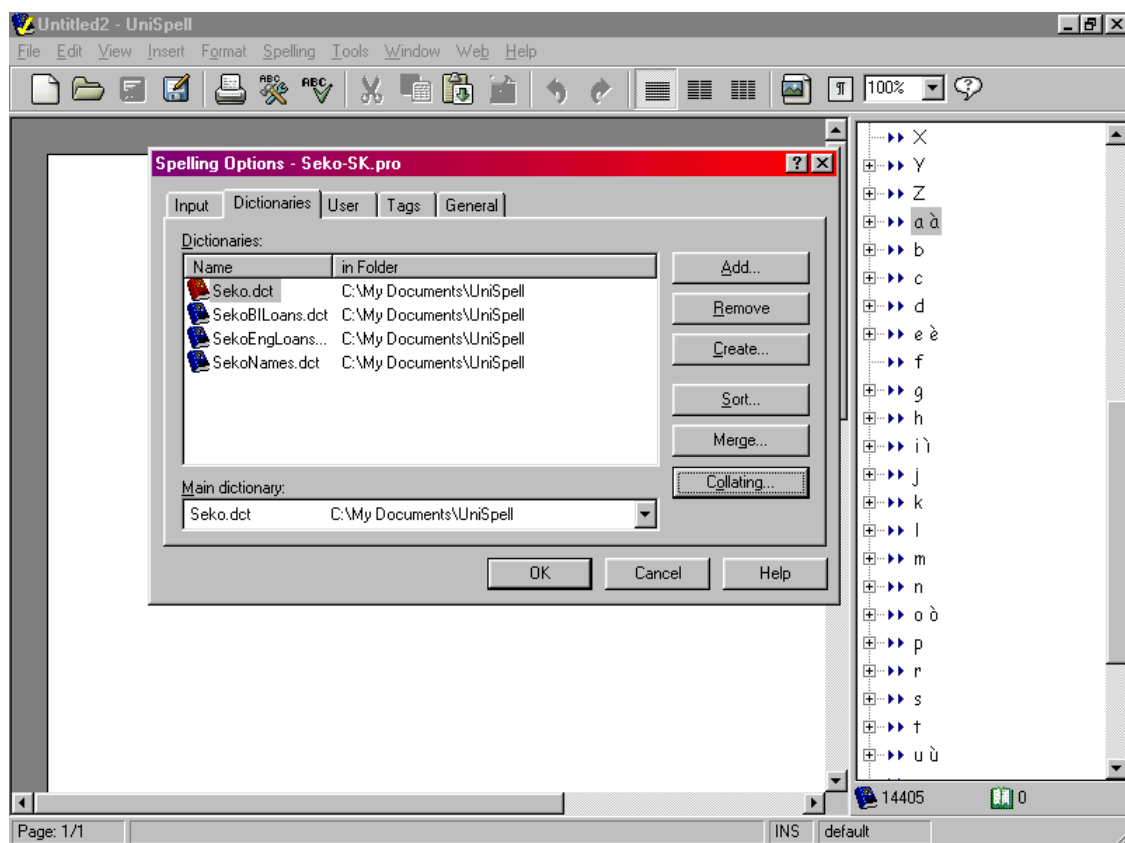


Figure 2 Seko profile loaded in UniSpell, Collating option highlighted

Before we spell check a Seko Padang file, I want to philosophize a bit, and this I believe to be the heart of the matter, especially for us who are lexicographers. How does one begin to create a vernacular spell checker? The simple answer is, you've got to **create a wordlist**. But there is a challenge involved, and that is, how can you create a wordlist **without introducing errors** in the process? That is the philosophical question.

Here is my philosophical answer: Create a vernacular spell check wordlist **from words that have been the most carefully examined**. For lexicographers, I would

posit that **the entries in your lexical database** are the best starting point. We can't get away from the task of eyeballing words and asking ourselves, "Is that word spelled correctly?" But if we can minimize the strain on our eyes and on our memories, I think we will have more satisfactory results. Let me show you where I went to begin building the Seko Padang spell check wordlist, and maybe it will give you ideas that will be useful in building yours.



Figure 3 Shoebox lexical database, lexeme field (\lx) highlighted

I keep my lexical database in a program called Shoebox. The database is organized by lexeme (not root), which is the header field (\lx) for each record. Those are the words I want to use in creating my initial UniSpell wordlist. I can create a separate file containing all those words with Shoebox's File > Export command.

I advise opening your exported file at this point in a text editor and looking it over. You should remove any bound forms, like prefixes and suffixes. When you are satisfied that the list contains only stand-alone words, load the file in UniSpell and run a spell check, adding all the words automatically to a .usr file. (I will demonstrate how to do that later.) After that, merge it into your UniSpell main dictionary. The result is an initial spell check dictionary of perhaps several thousand words. Hurray! You're done with the first process in creating your vernacular spell checker.

Now, not everybody who maintains a lexical database in Shoebox organizes it by lexeme. Several in Indonesia and the Philippines organize theirs by root. If you are one of those, you will need to take special precautions not to include "bound roots" in your spell check wordlist. Bound forms always have some kind of affix and would never be written as an independent word in a text. Otherwise, you might find yourself weeding out unnecessary roots from the spell check wordlist for years to come.

Building the vernacular spell check database

We can already begin to use the basic spell check database we have created. However, we're not finished building the spell check database yet. The situation can be exemplified by the pair of words in English "book" and "books". Only "book" would be found in the dictionary, not "books". But both words need to be in the spell check wordlist.

In the lexical database

book

-s

In the spell check wordlist

book

books

How can we add all those words that have so-called inflectional morphology to the spell check wordlist? It can be an enormous task. Let me philosophize again: Build on your spell check wordlist in small, doable chunks that are easy on your eyes, so that you don't get dizzy and start introducing errors. For just a few words, you can process them manually by pressing the Add button, if your knowledge of the language and editing skills are reliable. But if you do too many words at once, and the job gets tedious, it is very easy to stop thinking clearly and just press the Add button one more time, even though the word is actually misspelled.

I have found a [parser](#) to be a helpful tool for building up the spell check wordlist, even though it adds an extra step, and the complexity of it has risks of its own. But using a parser also has the advantage of helping you [keep your lexical database up to date](#) in relation to all the text material you have in your vernacular language. I find that advantage to be especially useful to me as a lexicographer.

Here's the process I followed in general outline. Further on I take you through a practical example in order to illustrate steps in using a parser. The way I built my spell check wordlist in doable chunks was to start out by checking all the example sentences in my lexical database. I exported them to a separate file, then checked them with UniSpell. After successfully passing new words through a parser, I added

them to the UniSpell dictionary. I then did the same with each of the other vernacular fields in the lexical database. After that, I spell checked nearly all Seko Padang texts and translated materials of publishable quality. From an initial list of about 7500 words, my spell check wordlist for Seko Padang now stands at about 14500 words, not counting proper names which I generally keep in a separate database. The entire process of learning how to use the UniSpell program and building the wordlist to that point took me less than 3 weeks. But keep in mind that at the same time I was able to clean up the vernacular in all those files and update the lexical database. I am confident that there are no words in the Seko Padang text files I have processed that are not accounted for somehow in my lexical database.

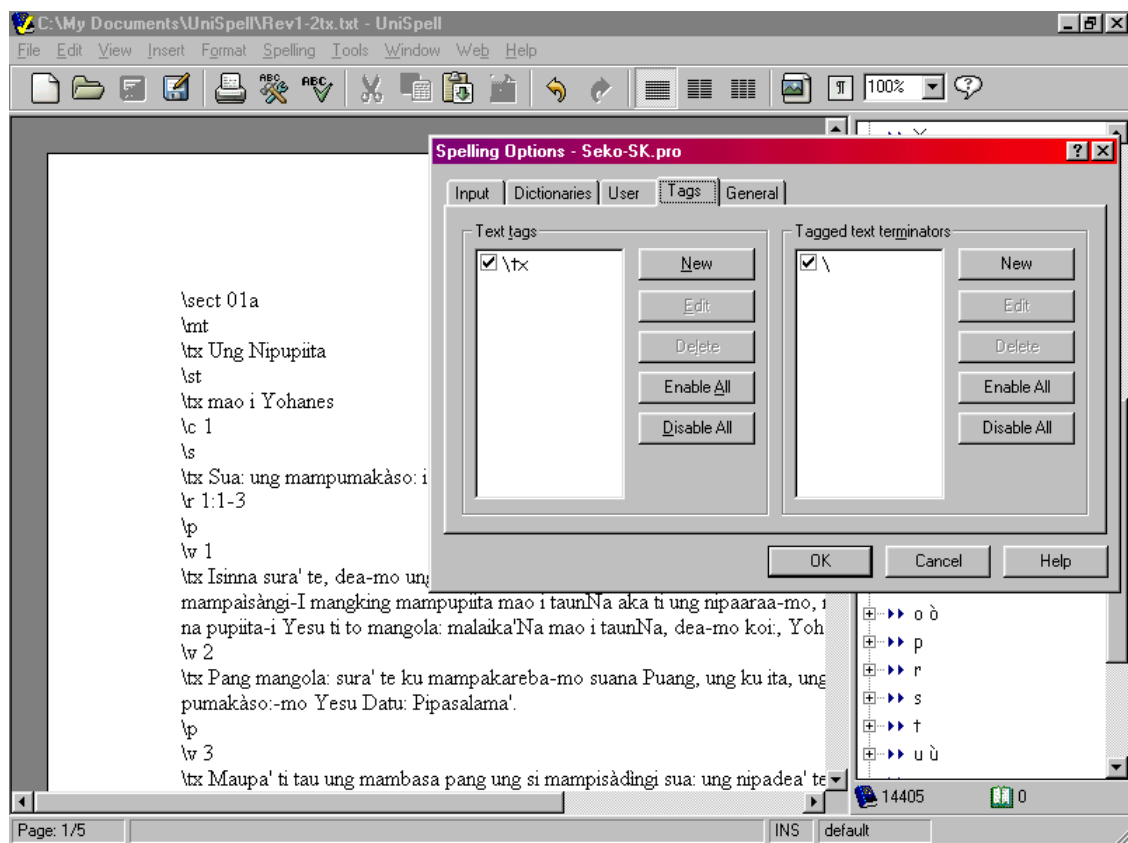


Figure 4 UniSpell with standard format text loaded, Tagged text dialog box displayed

I'll take us through a spell check of the next document I need to process in the project I'm working on. I am a translator and this file happens to be from the first two chapters of Seko Padang Revelation. One Option in UniSpell is to check only certain fields of a standard format text. You create your own list in the Tags tab. Figure 4 shows a simple list of only one marker \tx. According to the setting displayed, UniSpell stops checking when it reaches the next backslash code \.

Whether UniSpell should check only tagged text is toggled on or off under the Options > General tab (not displayed).

Now let's run a spell check on the Seko file. When it stops on a word, click Add To, and then check the "Add all unknown words without asking" box in the lower left corner (Figure 5). All words not found in the wordlist will automatically go to the default user file, in this case Seko.usr.

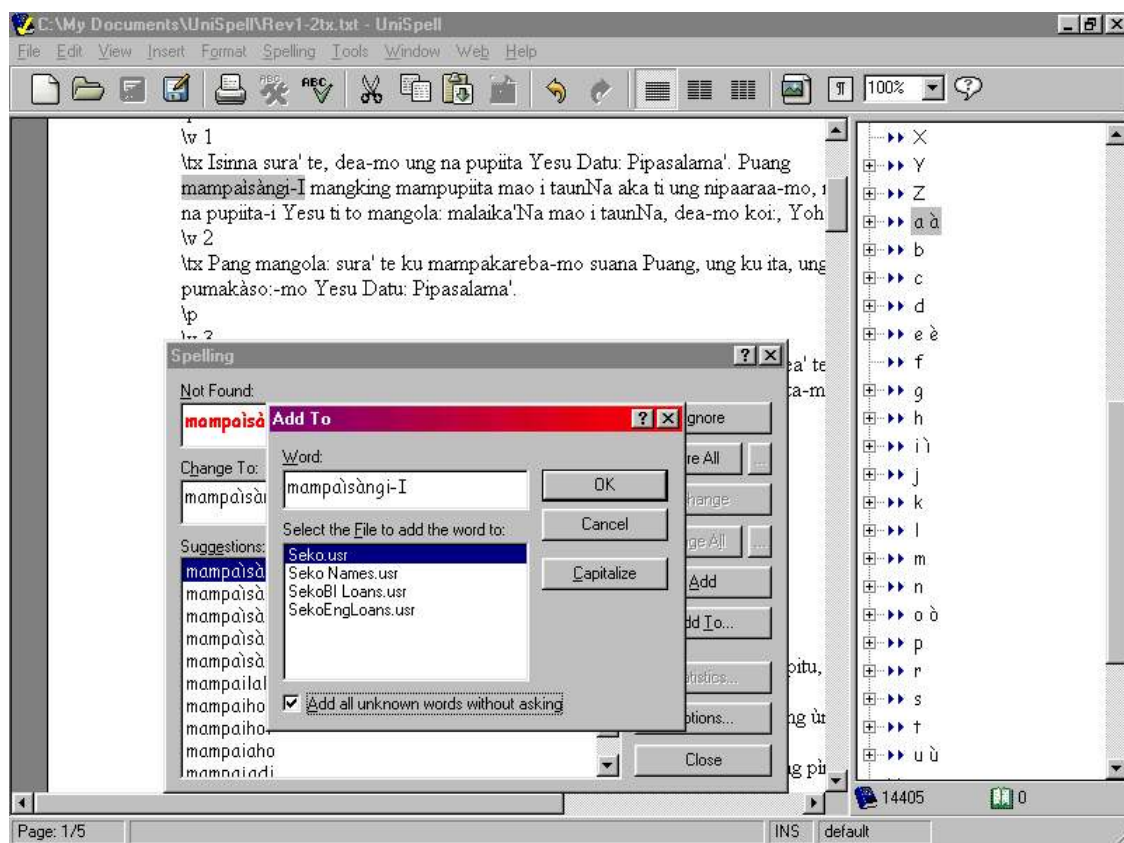


Figure 5 Spell checking the Seko Padang file, click Add To and check box "Add all unknown words without asking"

If we close the Seko file we just spelled checked and open the Seko.usr file in UniSpell, we can see the list of words the spell checker just found. Note that words are listed in the same order they were encountered in the text. This file will need to be saved as a file for Shoebox, preferably as a special database Type for parsing. UniSpell saves its files in Unicode, which Shoebox doesn't handle, although Toolbox does. I change the ID line, and Save As an Encoded Text file, Western European, Windows Code Page 1252. There are many options.

In Shoebox, I load the wordlist file I just created and renamed. I assign it as a "spell check parse" Database Type. Notice that the Interlinear tab has only a Parse process listed and no Lookup processes (Figure 6).

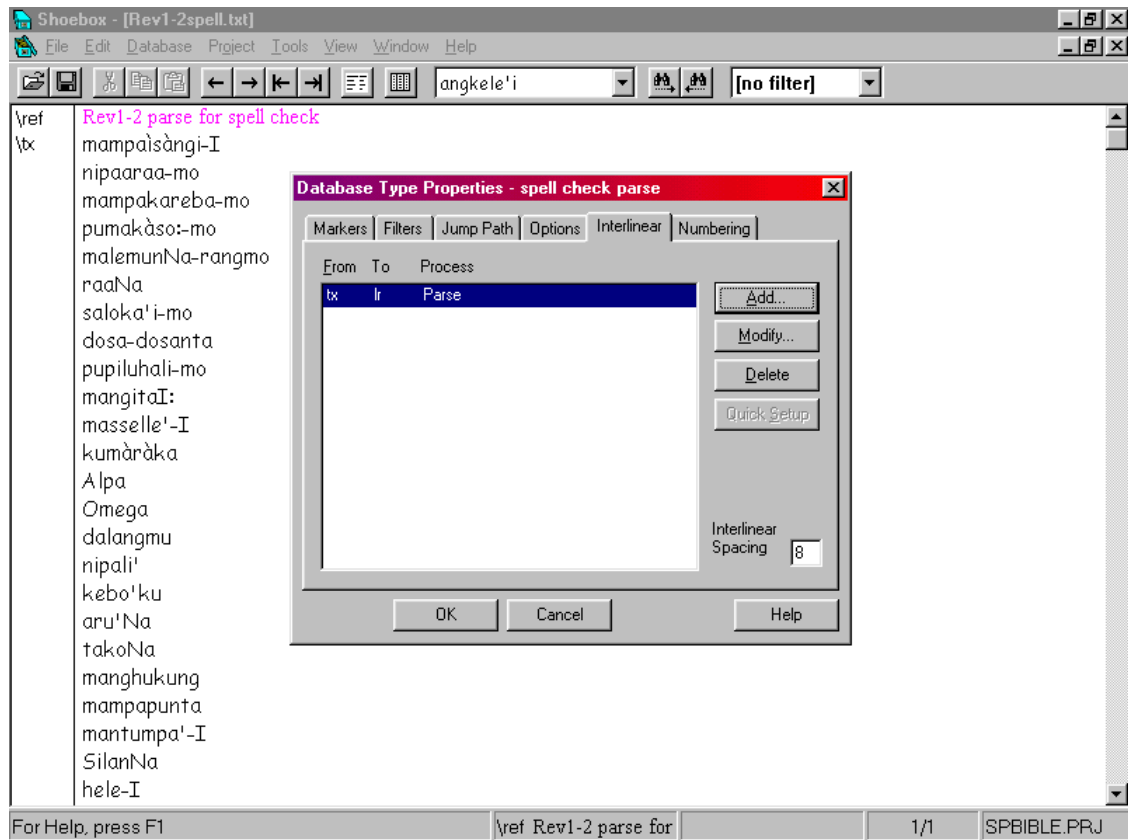


Figure 6 Shoebox list of words ready for parsing, same as Seko.usr list just created in UniSpell; Database > Properties > Interlinear parse process displayed

We interlinearize the wordlist and note that many of the words parse correctly (Figure 7). The only deficiency for those words is that they are not yet in a UniSpell wordlist for Seko. For correctly parsed words, the only thing to watch for is words (usually Names) that you may wish to place in a different UniSpell file other than the default.

In some cases, words fail to parse. In our Shoebox example, a failed parse is indicated by a string of three asterisks underneath the word (see again Figure 7). In fact, one of the advantages of using the Shoebox parser is that **the asterisks direct your eyes to where attention is needed most**. In other cases, words do parse, but incorrectly. So you still need to examine each parse to see if it makes sense. Words that fail to parse correctly need to be processed until they do.

There are four reasons why we might expect the parse to fail:

1. Spelling errors. These need to be corrected in the original file and also deleted from the .usr file. You can test by making a correction to the word in the parse file and reparsing, for example *hùra:* > *hura:*.
2. The word is not in a lexical database. Add it and reparse, for example *Alpa, Omega, mangangkele 'i (represent, wakili)*.
3. Incomplete parsing rule. These can be a real pain until you get the hang of how to handle making the improvements, for example *mangitaI: (-I: > +a i)*.
4. Complex words that the parser is not set up to handle. In my case, many reduplicated words are in this situation. I keep a special file in Shoebox for words like this where I tell the interlinearizer explicitly how to parse the word, for example *pampusua:-sua: kadetu-ing*.

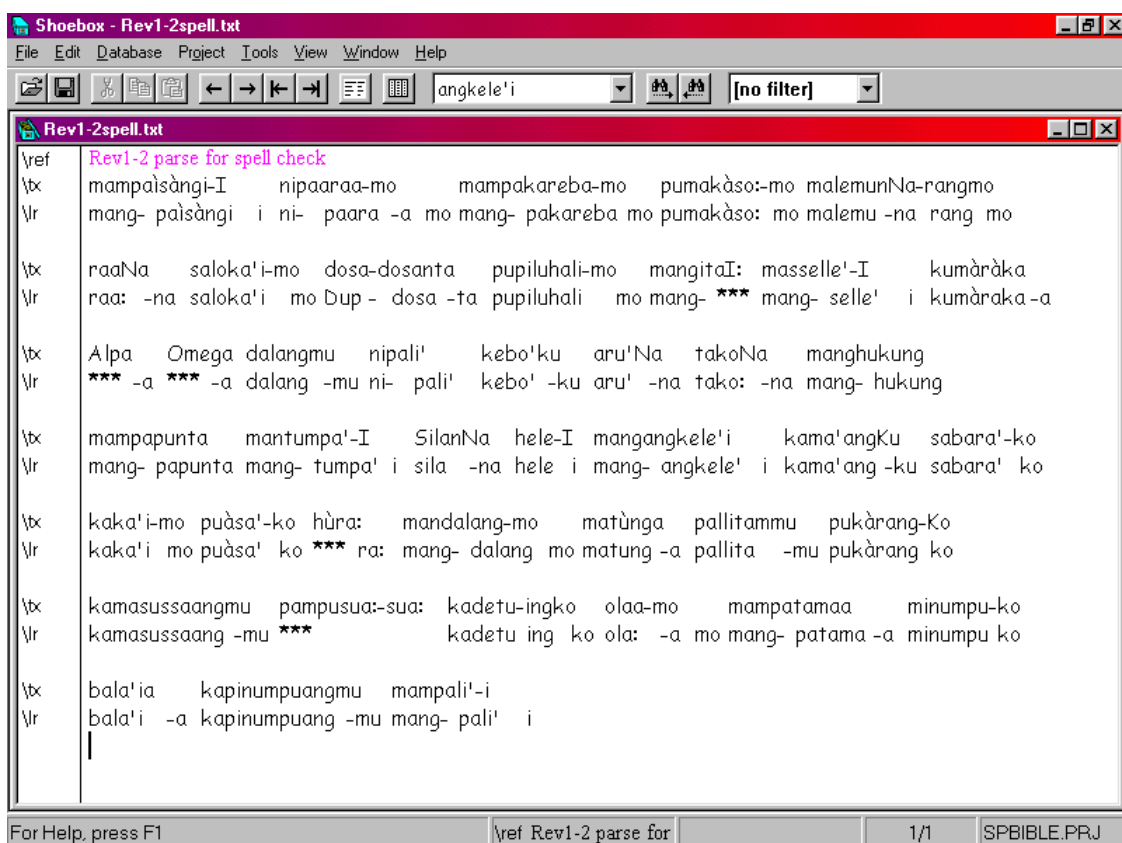


Figure 7 Parsed wordlist before corrections

Unfortunately, the parser can also fail due to weaknesses of the Interlinearizing function. These may be related to capital letters or untoward affix combinations. I won't go into how to handle those here.

Finally, after you've corrected spelling errors, updated your lexical databases, tweaked your parser, and moved any names to other files if you want to do that, you are ready to merge the .usr file with its corresponding dictionary.

After all your hard work, when you think everything was correctly processed, try running all your files through the spell checker once more and see what you get. That would make a good double check.

Multiple languages, but no standard format markers? No problem.

Some people may not use standard format markers in their lexical databases or have the capability of exporting only the words of one language from within a mixed-language database to be spell checked. In that case one could create a super spell checker in UniSpell that would check more than one language in the same file. I have tried it and got it to function properly, but the size of the combined wordlists totaled about 65,000 words, which is pretty big, and on my older computer it was slow to load or to alter the configuration.



Both AbiWord and UniSpell have active programming support, and I think we can expect upgraded versions of both those programs within a year, thanks to Urs Ruchti and other programmers.