University of Arizona

HLT Internship Portfolio

PanLex

Author:

Andréa Davis

Internship Supervisor:

Dr. Laura Welcher

October 10, 2012

Abstract

I spent my internship working as an editor of PanLex, a pan-lingual dictionary. I worked both at formatting sources for entry into the database, and on improving an algorithm for the normalization of expressions.

Contents

1 Introduction

PanLex is a pan-lingual lemmatic dictionary. The basic goal is to have translations between all languages of the world, ideally for a large number of expressions in each language. By entering data from various sources, including dictionaries, grammars, wiktionaries, dissertations, and thesauruses, PanLex will collect all lemmatic translations in a single location. Lemmas do not break down words into morphology, so PanLex does not take morphology into account.

At present, global communication - on the internet, in the academic world, in business, and in language technology - is dominated by a handful of languages. This severely limits those who do not speak one of these dominant languages, as they must either forego these areas of communication, or use one of the dominant languages, often to the exclusion of using their own. It is hoped that PanLex may make global communication more egalitarian, by allowing people to use their own language, at least more often. It is also hoped that increasing the opportunity to use a native language rather than a dominant language will help people to keep their languages, and will thus reduce language loss. These primary goals of PanLex will be described in greater detail, in Section 2.3. Additional applications of PanLex will be discussed in Section 2.4.

The structure of PanLex differs from other dictionaries, in that for a given search, a web of translations exist. For example, if in the original sources there is a translation from language X into language Y, and there is another translation of that lemma from language Y into language Z, there will then be a translation of that word from language X to language Z, even though there is no original source dictionary with that translation. This makes it possible to translate far more expressions than would be possible with a more typical dictionary.

2 PanLex

2.1 The structure of PanLex

PanLex's structure may be thought of as a web, where each expression is linked to many other expressions, which in turn are linked to other expressions. Primary links exist based on editorial input, while secondary links (links through a linked expression) are enabled by this web structure of PanLex. This is what enables PanLex to make translations that do not necessarily exist in the sources fed into PanLex.

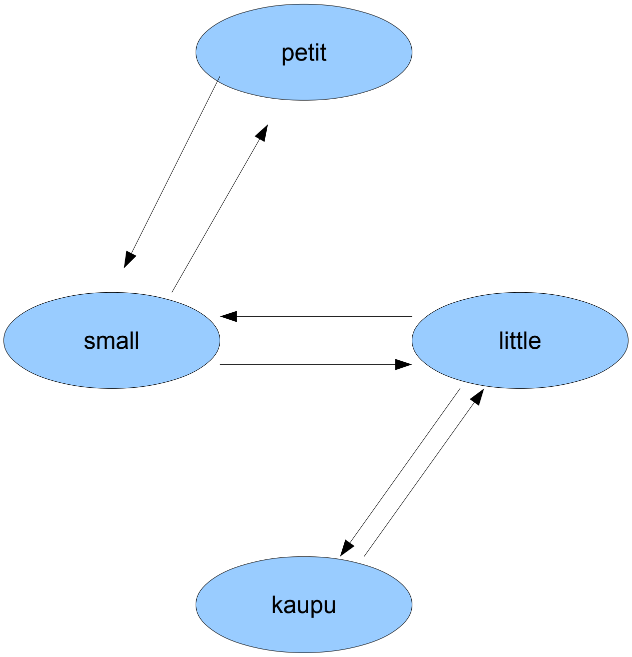


Figure 1: PanLex's Web of Translations

Figure 1 shows how this works. The French expression *petit* has a translation into the English expression *small*, which has the synonym *little*. Since the expression *little* has a translation into Purari, *kaupu*, then it is possible to translation *kaupu* into French, or to have two English translations for the expression *kaupu*, even though from the original source there was only one translation for *kaupu* - *little*.

2.2 Interface

PanLex is publicly accessible at http://panlex.org/u. Once a language is chosen for the interface, the user may choose among a number of options.

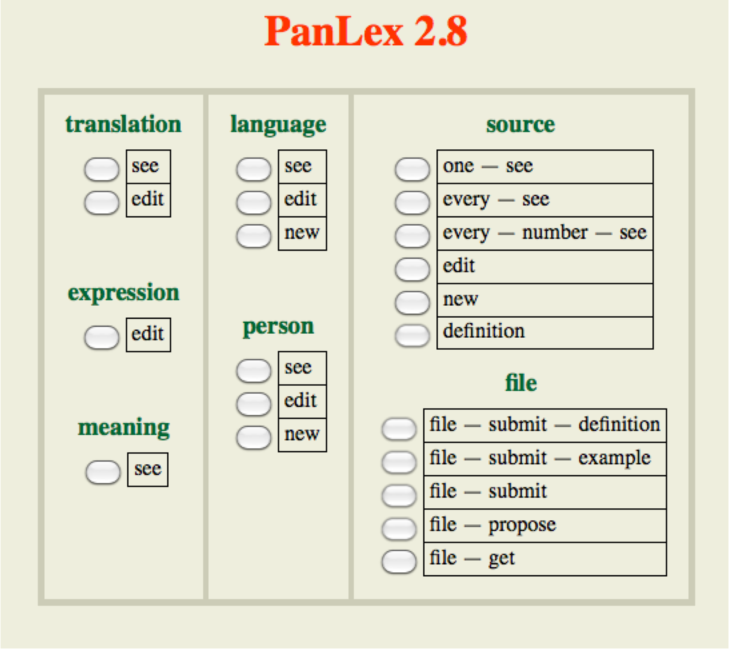


Figure 2: PanLex's interface

She may choose among looking up (through *see*) the translation of an expression, editing an expression, seeing the meaning/translation from a particular source (this will not produce secondary translations), seeing the sources existing for a particular language, seeing the people involved in editing the dictionary, seeing the sources that have been used, or particular formatted files that have been entered into PanLex.

Typically, the user would be interested in seeing translations. Editing options exist only for those with a username and password - those who have been approved to edit PanLex.

2.3 Language diversity and PanLex

The basic goal of PanLex is to get all the world's languages into a single, electronic source. The hope is that having translations between all languages in the world will actually help to preserve linguistic diversity, in a variety of ways. For one, simply having translations between languages reduces the need to use a single, dominant language. Currently, a handful of languages dominate in business, academia, the internet, and language technology. As these are the primary forms of global communication, those who do not speak one of these dominant language are at a disadvantage.

PanLex may be helpful for allowing speakers of minority languages to continue to use their language in global communication, using lemmatic communication. Lemmatic communication is a simplified form of language, in which all words used are lemmas - words that are not broken down by morphology. Using this kind of communication greatly simplifies translation, making it feasible to use a resource like PanLex. The availability of lemmatic communication is only useful if such simplified language can convey meaning. Tests of the comprehensibility of lemmatic communication have shown that when people are forced to compose messages using only expressions contained in PanLex, they are able to understand each others' meaning across languages with a high degree of accuracy (Everitt et al., 2010).

PanLex is unique among multi-lingual dictionaries, by the sheer number of languages it offers. Since 2007, PanLex has grown greatly in the number of language varieties, as well as in the number of expressions in each variety. Figure 3 shows the growth of PanLex between its beginnings in 2007 and 2011. The y-axis represents the number of language varieties, while the differently-colored lines represent the number of expressions. Thus, for the blue line, PanLex contains at least 6 expressions in about 5,500 language varieties. Other lines represent a larger number of expressions (20, 60, 200 600, 2000, 6000, 20000), and show that PanLex does not have as many language varieties with a larger number of expressions.

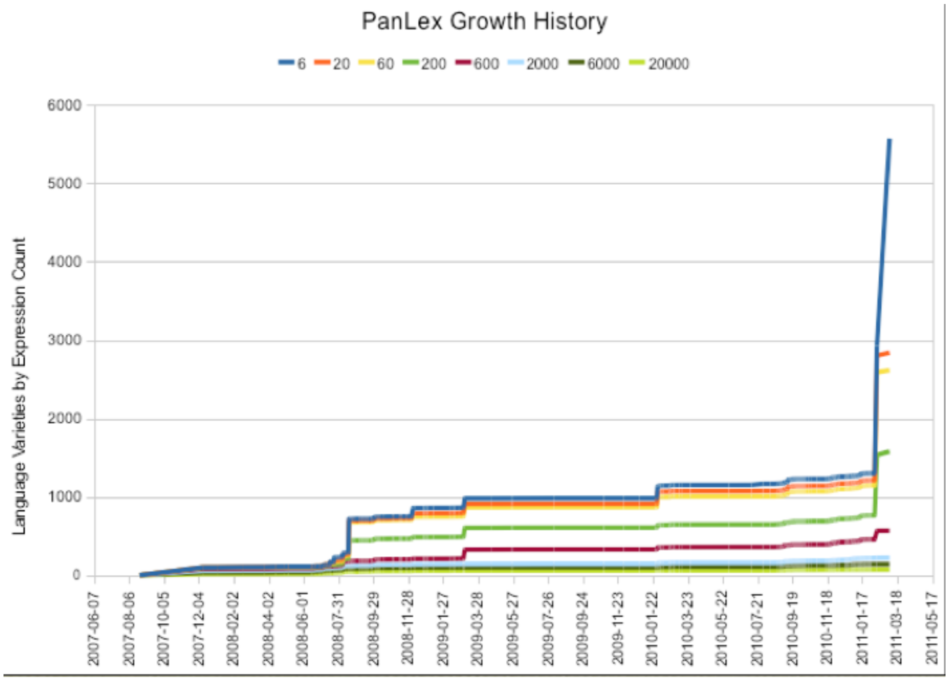


Figure 3: The growth of PanLex, from Pool & Colowick (2011)

Currently, PanLex has expressions in over 6,000 language varieties. The graph shows that not all language varieties yet have a large number of expressions, but PanLex continues to expand both in terms of language variety and the number of language varieties having a large number of expressions in PanLex.

2.4 Other uses of PanLex

PanLex has applications beyond facilitating translation between humans; it also may be useful for improving internet search algorithms. PanImages is one such application (Colowick, 2008). PanImages addresses a problem in image search: ambiguity. For example, the English word *spring* has several meanings: the season, a child's toy, the action of jumping, and potentially others. Frequency of meaning is one potential solution to this problem, but the user is not necessarily looking for the most frequent meaning. However, if there were a translation available for that word, this would considerably reduce the ambiguity. A meaning that is expressed *spring* in English and *printemps* in French can only be the season. PanImages suggests translations of the search expression, allowing the user to narrow the search, at least if he or she knows translations in at least one of the languages suggested by PanImages. The advantage of this solution is that it avoids directly encoding meaning, which may be difficult to do.

PanLex also has applications in machine translation. Its database in principle exceeds that of all other machine translators, by the number of languages available. PanLex already offers over 6,000 language varieties (out of an estimated 7,000); in contrast, Google Translate offers only about 300 language varieties. As PanLex continues to grow, more expressions will be offered in more language varieties, expanding its translation capabilities. However, while PanLex's database may be larger in terms of the number of language varieties, Google has a much larger database in terms of tokens, allowing the use of statistical techniques. Specifically, Statistical Machine Translation is dependent on access to large parallel corpora, which do not exist for a large number of language varieties. To overcome this problem, Soderland et al. (2009) propose PanLingual Translator, which uses Lemmatic MT. This style of translation does not yield elegant language, but it is effective at transmitting basic meaning. PanLingual Translator had high translation accuracy for between 59% and 99% of sentences, from 6 languages; in contrast, Google Translate had high translation accuracy for between 34% and 93% of the sentences. For sentences from 5 languages that are not handled by Google Translate, PanLingual Translator had high translation accuracy for between 27% and 82% of sentences. Thus, PanLex has the capability of vastly extending the number of languages between which it is possible to communicate, using MT.

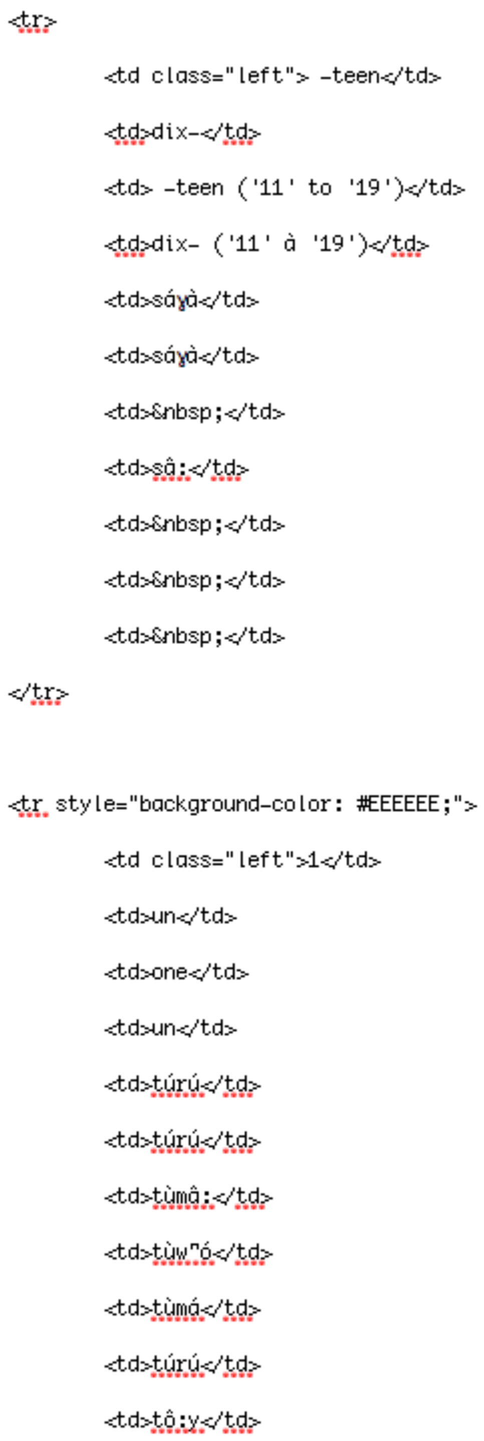
3 Adding to PanLex

The largest difficulty in constructing PanLex is inputting primary sources. Primary sources are the original sources, which have not yet been formatted for PanLex, but which have been collected in a database for future formatting. For example, a grammar of Parakanã would be a primary source. Each source has a different, and potentially unique, format, which is not compatible with PanLex. The difficulty, then, is in formatting the primary source into a file that can be read and understood by PanLex. This is very time-consuming, as each source must be processed individually, at least initially, since formatting algorithms written for one source will not apply to the next source, which will have its own, individual format.

Nonetheless, since certain common problems exist across sources, and since the kind of information that is extracted is mostly the same across sources, some of the formatting can be generalized into code usable by most sources, provided that the particular source is pre-formatted to an intermediate format. This intermediate format allows for more variability than PanLex itself.

3.1 Formatting

Formatting data is the most time-consuming part of inputting data into PanLex. Each source contains expressions as well as other potentially useful information (such as word class or other metadata) which needs to be extracted and re-formatted. As mentioned, since the original formatting is nearly always unique to that source, each source requires code tailored to that particular source. An example of an unformatted source is shown in Figure 4).

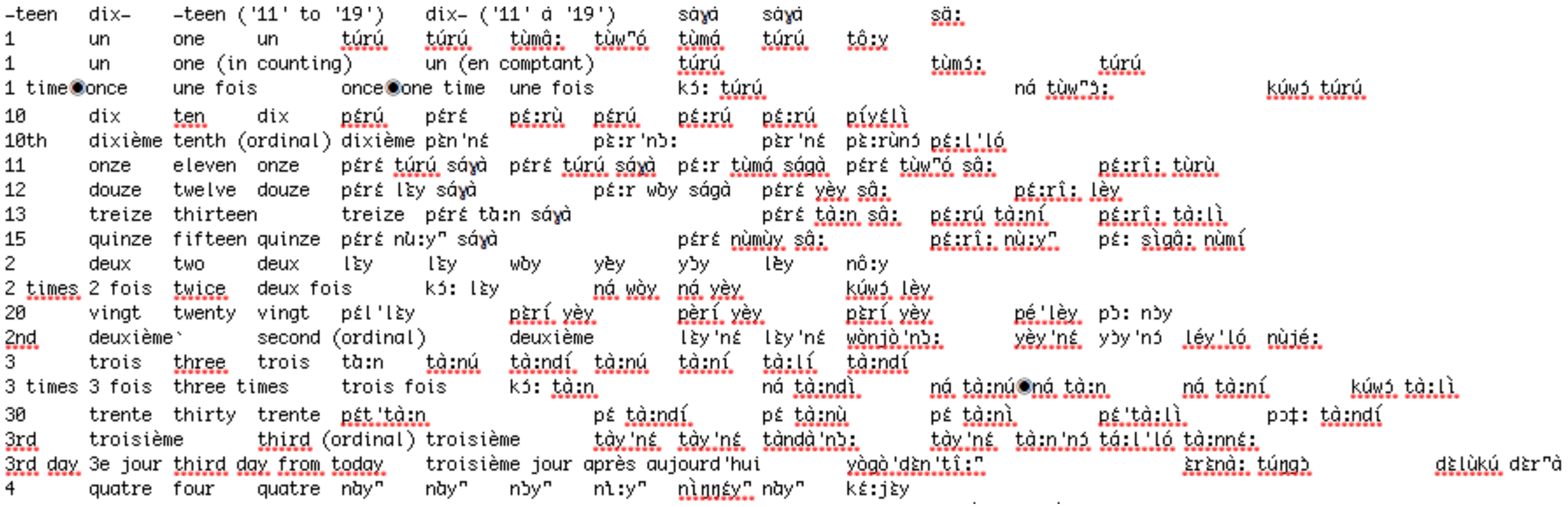


```
<tr>
        <td class="left"> -teen</td>
        <td>dix-</td>
        <td> -teen ('11' to '19')</td>
        <td>dix- ('11' à '19')</td>
        <td>sáyà</td>
        <td>sáyà</td>
        <td>&nbsp;</td>
        <td>sâ:</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
        <td>&nbsp;</td>
</tr>

<tr style="background-color: #EEEEEE;">
        <td class="left">1</td>
        <td>un</td>
        <td>one</td>
        <td>un</td>
        <td>túrú</td>
        <td>túrú</td>
        <td>tùmâ:</td>
        <td>tùw"ó</td>
        <td>tùmá</td>
        <td>túrú</td>
        <td>tô:y</td>
```

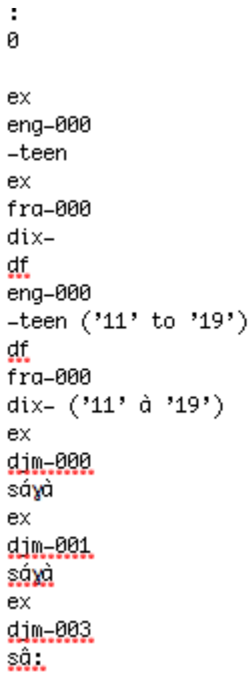
Figure 4: Original format of a source; this is in html format. In this case, each line contains an expression in one language. Some lines contain symbols understood by html, which are viewed as phonetic symbols on a website. The $< tr >$ separates meanings, so this shows two meanings, each with translations into a number of languages

This problem is made slightly easier by first formatting a source to an intermediate stage, which nonetheless allows more variability than the format required by PanLex. Accordingly, a source is first transformed into tabular format, with one line per entry, and a column for each kind of information contained in the entry, such as translations, word class, or other metadata. As can be seen in Figure 5, each column contains one kind of information; the black circles within a column are used to separate synonyms, so that in the final formatting, these expressions can be associated in PanLex as having the same meaning (the black circles exist only in columns contain expressions, and not in columns containing metadata). This is done using regular expressions to capture the relevant information in the original source. Once the original source is transformed into this intermediate format, more general scripts can be used to transform it into PanLex format (shown in Figure 6). These general scripts accomplish a variety of tasks: a) standardizing apostrophes, b) marking columns as expressions, word class, metadata, etc, c) normalizing the expressions, and d) marking the language variety of expression.



```
-teen   dix-    -teen ('11' to '19')    dix- ('11' à '19')      sàɣà    sàɣà            sä:
1       un      one     un      túrú    túrú    tùmâ:   tùwⁿó   tùmá    túrú    tô:y
1       un      one (in counting)       un (en comptant)        túrú                    tùmɔ́:           túrú
1 time●once     une fois        once●one time   une fois        kɔ́: túrú                        ná tùwⁿɔ̀:               kúwɔ́ túrú
10      dix     ten     dix     pɛ́rú    pɛ́rɛ́    pɛ́:rù   pɛ́rú    pɛ́:rú   pɛ́:rú   píyɛ́lì
10th    dixième tenth (ordinal) dixième pɛ̀n'nɛ́         pɛ̀:r'nɔ̀:        pɛ̀r'nɛ́  pɛ̀:rùnɔ́ pɛ́:l'ló
11      onze    eleven  onze    pɛ́rɛ́ túrú sáɣà  pɛ́rɛ́ túrú sáɣà  pɛ́:r tùmá ságà pɛ́rɛ́ tùwⁿó sâ:         pɛ́:rî: tùrù
12      douze   twelve  douze   pɛ́rɛ́ lɛ̀y sáɣà          pɛ́:r wɔ̀y ságà  pɛ́rɛ́ yèy sâ:           pɛ́:rî: lèy
13      treize  thirteen        treize  pɛ́rɛ́ tà:n sáɣà                  pɛ́rɛ́ tà:n sâ:   pɛ́:rú tà:ní     pɛ́:rî: tà:lì
15      quinze  fifteen quinze  pɛ́rɛ́ nù:yⁿ sáɣà                 pɛ́rɛ́ nùmùy sâ:          pɛ́:rî: nù:yⁿ    pɛ́: sìgâ: nùmí
2       deux    two     deux    lɛ̀y     lɛ̀y     wɔ̀y     yèy     yɔ̀y     lèy     nô:y
2 times 2 fois  twice   deux fois       kɔ́: lɛ̀y         ná wɔ̀y  ná yèy          kúwɔ́ lèy
20      vingt   twenty  vingt   pɛ́l'lɛ̀y         pɛ̀rí yèy        pèrí yèy        pɛ̀rí yèy        pé'lèy  pɔ̀: nɔ̀y
2nd     deuxième`       second (ordinal)        deuxième        lɛ̀y'nɛ́  lɛ̀y'nɛ́  wɔ̀njɔ̀'nɔ̀:       yèy'nɛ́  yɔ̀y'nɔ́  léy'ló  nùjé:
3       trois   three   trois   tà:n    tà:nú   tà:ndí  tà:nú   tà:ní   tà:lí   tà:ndí
3 times 3 fois  three times     trois fois      kɔ́: tà:n                ná tà:ndì       ná tà:nú●ná tà:n        ná tà:ní        kúwɔ́ tà:lì
30      trente  thirty  trente  pɛ́t'tà:n                pɛ́ tà:ndí       pɛ́ tà:nù        pɛ́ tà:nì        pɛ́'tà:lì        pɔ‡: tà:ndí
3rd     troisième       third (ordinal) troisième       tày'nɛ́  tày'nɛ́  tàndà'nɔ̀:       tày'nɛ́  tà:n'nɔ́ tá:l'ló tà:nnɛ́:
3rd day 3e jour third day from today    troisième jour après aujourd'hui        yɔ̀gɔ̀'dɛ̀n'tî:ⁿ                   ɛ̀rɛ̀nà: túŋgɔ̀            dɛ̀lùkú dɛrⁿà
4       quatre  four    quatre  nàyⁿ    nàyⁿ    nɔ̀yⁿ    nì:yⁿ   nìŋŋɛ́yⁿ nàyⁿ    kɛ́:jɛ̀y
```

Figure 5: Intermediate, tabular format, showing data for the language varieties (by column): 1) English 2) French 3) English 4) French 5) Jàmsay 6) Gourou 7) Nanga 8) Beni 9) Walo 10) Tabi 11) Najamba



```
:
0

ex
eng-000
-teen
ex
fra-000
dix-
df
eng-000
-teen (’11’ to ’19’)
df
fra-000
dix- (’11’ à ’19’)
ex
djm-000
sáɣà
ex
djm-001
sáɣà
ex
djm-003
sâ:
```

Figure 6: Formatted for PanLex

3.1.1 PanLex’s definition of lemma

PanLex’s entries are lemmatic, in the sense that they do not take morphology into account. However, PanLex’s definition of a lemma is not exactly a linguist’s definition of lemma. Morphology may be present in PanLex expressions; the expression is still considered lemmatic because PanLex does not take morphology into account. That is, there are not entries for morphemes independent of the expressions they are in; thus, there may be an entry for the expressions *swims* and *swimming*, but there is no entry for *-ing* or *-s*. Expressions like this which include extra morphology tend to have fewer sources that include them, meaning they will also have fewer common translations. Thus, although they are included in PanLex, they do not end up participating in most of the translations that PanLex produces. Likewise, a user looking for translations specifically for *swims* or *swimming* would have more luck looking up *swim*: *swims* and *swimming* have translations in 3 and 80 languages, respectively, while *swim* has translations into 1542 languages. Alternatively, the user can look up *swims*, but broaden the search by allowing PanLex to search for matches that are not exact (this is done by unchecking the *exact* box when entering a search term). Likewise, *nagez* - the second person plural form of the infinitive *nager* in French, has translations into only one language, while *nager* has translations into 1027 languages. Thus, while PanLex contains forms which are not lemmatic by a linguist's standards, the lemmatic form generally receives the most support from sources, and is connected with the most translations. This is also the case for Portuguese, where *nado* - the first person singular of *nadar* - has a translation into only one language, while *nadar* has translations into 11 languages.

Other expressions are excluded from entry as expressions (but may be entered as definitions, a category not used in PanLex's web of translations). The criterion for exclusion is simply the length of the expression, counted both in words and in characters. If the candidate expression exceeds the maximum length, as determined by the editor (but typically 3 words or 25 characters), then it is assumed to not be lemmatic (either it is very morphologically complex, or it is a short phrase or sentence). The cut-off may be adjusted, depending on the language variety (agglutinating languages may require a higher cut-off for number of characters, for example).

3.1.2 The importance of Unicode

The Unicode Standard was developed to make the world's writing systems' encoding consistent. Unicode encoding allows multi-lingual computing. This is not possible with traditional character encodings, which tend to be incompatible with each other. If technology is to be available to all, then it is essential to have a single character encoding.

The Unicode Standard was thus developed to address this problem of incompatibility. While Unicode requires more memory per character than ASCII (typically 4 bits per character, rather than one in ASCII), it allows the creation of multi-lingual technology like PanLex. PanLex relies crucially on Unicode, since there would be no way to have a single system of all the world's languages, if they could not all be encoded using a single standard. Accordingly, all sources for PanLex are encoded using Unicode.

3.2 Normalization

Once a source is transformed into formatting appropriate for PanLex, there is a question as to the quality of the source. Often an editor can make judgments, after working with the source, as to how trustworthy the translations are, and PanLex offers a way to assign a number for quality, for a given source.

However, even if a source is high quality, with trustworthy translations, a second problem lies in orthographic normalization. This problem arises because different sources may have orthographic variations in representing an expression. For example, the English expression *good night* is sometimes written *good-night* or even *goodnight*. This is a problem for PanLex, since to recognize an expression as being the same, the orthography must be identical. Further, sources occasionally contain expressions which are not, for PanLex's purposes, actual expressions. For example, I worked on a dissertation which had translations between both words and sentences in English and Parakaña. The sentences should not be entered into PanLex, since they are not lexical items.

A normalization algorithm can help with these problems, by identifying candidates for normalization(for well-represented language like English, good candidates are expressions that do not already exist in PanLex, or that minimally differ from a word in PanLex by punctuation marks, accents, capitalization, or spaces). The normalization script currently used for sources for PanLex both marks candidates for normalization and replaces the old expression with another that is minimally different, and is the most common in PanLex (that is, the one that is supported by the most sources), if there is such an expression. The way that this algorithm identifies candidates for normalization is to match the expressions from the source that is being formatted to expressions that already exist in PanLex, for the relevant language, where expressions that differ by only a space, a capitalization, an accent mark, a hyphen, or other non-alphanumeric characters are matches. For example, if *december* is an expression that is in the source being formatted, and is marked as belonging to the English language variety, then the normalization algorithm will find that it matches the already existing English expression *December.* Thus, it is marked as a target for normalization, and the algorithm gives the suggested normalization of *December* for its representation. Alternatively, if *Raindrops are falling on my head* is an expression that is in the source being formatted, and is also marked as English, then the normalization algorithm will not find a match (PanLex does not have an expression *Raindrops are falling on my head*). In this case, the default is to mark this expression as a definition, rather than as an expression, which means that it will be left as is, but will not be included in PanLex's web of translations, but rather will be stored as metadata (which is not currently used in PanLex translations). Since the assumption of this algorithm is that all the expressions it is acting on are already in PanLex, this algorithm is only used for language varieties that are already well-represented in PanLex (like English). It is also not recommended if the source is very particular in its subject matter, since presumably less common expressions would not exist in PanLex, even for well-represented language varieties.

This script, however, proved to be over-normalizing, even for languages that are well-represented. For example, two words in Portuguese differ by only an accent mark - *avó* and *avô.* The original normalization script incorrectly altered *avô* to *avó*, since, although *avô* existed in PanLex, it was not as common as *avó.*

There are a variety of ways to correct for this problem. Parameters were added to the script:

1. A parameter whose value determines the minimal goodness rating for an expres-

sion, in order for it to be entered into PanLex, where a goodness rating is based on the amount of support it receives from its previous entries in PanLex (which is in turn based on how many sources included it as an expression, and the editor-assigned rating of the source(s) including it).

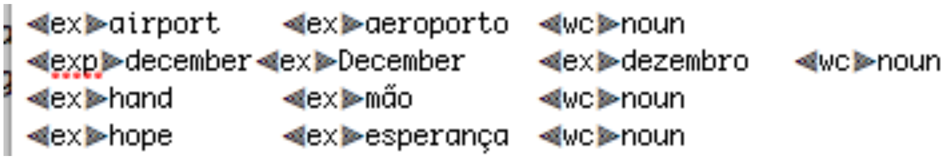
2. A parameter whose value determines how much more common a replacement expression had to be than an original expression, for the replacement expression to be chosen over the original. A negative threshold means that the expression with a better goodness rating will always be chosen, while a 0 threshold means that the expression is never replaced, provided that it exists in PanLex.

The first parameter can prevent expressions with low goodness ratings from being entered into PanLex, regardless of whether they already exist in PanLex or not. The second parameter specifically deals with the *avô-avó* problem.

A variety of values for the second parameter were tested on a formatted source, Brazilian Portuguese Vocabulary, and compared to a hand-corrected gold standard. A sample of the gold standard is shown in Figure 7. The gold standard was made by correcting the result of the original normalization algorithm (ie, before the parameters were added). Each expression is tagged with the tag ◀*ex*▶, while an expression that is to be normalized is tagged with ◀*exp*▶ (the tag ◀*wc*▶ is for word class). As can be seen, the expression *december* has the suggested normalization *December*. This is correct, and so is left as a proposed change in the gold standard. Other incorrect suggested changes, such as the change from *avô* to *avó*, were removed. Likewise, each expression was examined, to determine if any expressions were not given a suggested change when they should have been, but none of these were found.

Results (Table 1) showed that for at least this source, and likely for other sources, perfect normalization is not attainable using this algorithm.

Further, it is likely that choosing the best parameters will depend on the particular source; there is no single value for each parameter that will be best for all sources.



```
◀ex▶airport     ◀ex▶aeroporto  ◀wc▶noun
◀exp▶december◀ex▶December     ◀ex▶dezembro    ◀wc▶noun
◀ex▶hand        ◀ex▶mão        ◀wc▶noun
◀ex▶hope        ◀ex▶esperança  ◀wc▶noun
```

Figure 7: A sample of the gold standard

Threshold	True positives	False positives	True negatives	False negatives
Negative threshold	45	28	1931	0
5 threshold	37	22	1937	8
0 threshold	37	21	1938	8

Table 1: Results of normalization for different values for the second parameter

That said, using the current algorithm, it is best to assign a negative threshold, thus minimizing false negatives, and then to manually check all proposed changes. Since the number of proposed changes is generally reasonably small, this is not a terribly time-consuming task, relative to the amount of time already put into formatting a source.

3.3 Documentation

Documentation for uploading sources to PanLex is included in the appendix.

4 Conclusion

PanLex continues to grow. Over the summer, I formatted and uploaded 4 sources for PanLex: a) Brazilian Portuguese Vocabulary 2) Construindo um Dictionário Parakanã-Porugûes 3) Mono 2000-item digital wordlist: Presentation form and 4) Comparative Lexicon of Dogon Languages. This increased the database by thousands of words. Equally important, I increased its database by 10 language varieties, none of which were previously in PanLex: Brazilian Portuguese, Parakanã, Mono, Jamsay, Gourou, Nanga, Beni, Walo, Tabi, and Najamba. All of these had translations into at least one well-represented language (which increases the number of possible translations for each expression). Additionally, the quality of PanLex's entries continues to improve. In addition to formatting sources for PanLex, I assisted in improving its normalization algorithm, and in making suggestions for future editing. This will improve the accuracy of translations between languages.

At present, PanLex's web of translations allows lemmatic translation between a wide variety of languages. Translations between expressions are interpreted as broadly as possible. As such, it maximizes the number of translations between expressions and language varieties, but sometimes at the cost of accuracy of translation, or quality of translations. The decisions to maximize the number of possible translations, and to include only lemmatic translations, means that a large number of expressions can be entered into PanLex within a reasonable amount of time, which may be crucial for halting language death. It also means that more language varieties can have more translations, due to PanLex's web-like structure, thus increasing the representation of less common languages within PanLex. However, this means that PanLex cannot produce truly grammatical translations, since neither inflectional morphology, nor word order, are included. Lemmatic translation has been shown to be successful as a way to communicate, but it is more work for the user. Having lemmatic translations between all the languages of the world is certainly a step in the direction of having higher quality translations, however. If PanLex proves to work within its limitations, it may one day be possible to produce grammars that would add inflectional morphology, and give rules of word order, for higher quality translations. At present, the limitation lies in machine translation techniques, which currently rely on large corpora to produce high quality translations of full phrases and sentences.

As PanLex improves, its potential applications also grow. Some of the applications it has been used for were described in this paper, but the fact that it is not tailored to a specific use means that it may be applied to as yet unidentified problems. It continues to be hoped that its existence will ameliorate the language loss that is currently happening, by adding less common languages to the currently very small number of languages used in global communication and technology.

References

Colowick, S. (2008). Multilingual search with PanImages. *Multilingual*, *19*(2).

Everitt, K., Lim, C., Etzioni, O., Pool, J., Colowick, S., & Soderland, S. (2010). Evaluating lemmatic communication. *trans-kom*, *3 70–84*, 70–84.

Pool, J., & Colowick, S. (2011). Panlex: A panlingual lexicon. In *DELPH-IN Summit*.

Soderland, S., Mausam, C. L., Qin, B., Etzioni, O., & Pool, J. (2009). Lemmatic machine translation. In *Proceedings of Machine Translation Summit XI*, vol. I.

A Formatting Code

The following code was written to format four different sources.

A.1 A Brazilian Portuguese Vocabulary

```
#!/usr/bin/perl -w

#formats pak-por-Ferreira.txt-0 to tabular format
#first column is the source
#second column is the part of speech for PanLex
#third column is the part of speech in the original dictionary
#remaining columns are targets
#(if there’s only one translation, there will be only one target)

use warnings ’FATAL’, ’all’;
# Make every warning fatal.

use strict;
# Require strict checking of variable references, etc.

#use encoding ’utf8’;
# Make Perl interpret the script and standard files as UTF-8 rather than bytes.

my $input = "por-eng-Brasileiro-0.txt";
my $output = "por-eng-Brasileiro-1.txt";
```

```
open(DICIN, "<$input") or die "can’t open file: $!\n";
#open input

open (DICOUT, ’>:utf8’, ">$output") or die "can’t open file:$!\n";
#open output



my @entries;
#will be filled in

my $eng;

my $gram;

my $por;



while (my $line = <DICIN>) {

chomp $line;

my $entry = $line;

$entry =˜ s/\{tab\}/\t/g;

if ($entry =˜ m/(.+)\t(.+)\t(1A|1C|1E|1G)-\s*(.+)/) {
```

```
#the first column is English
#pattern match line

$eng = $1;

$gram = $4;

$por = $2;

}

elsif ($entry =~ m/(.+)\t(.+)\t(1B|1D|1F|1H)-\s*(.+)/) {
#the first column is Portuguese
#pattern match line

$eng = $2;

$gram = $4;

$por = $1;

}


if ($entry !~ m/Expressions/) {
#only remove function words for non-expressions
#expressions are short phrases
```

```
$eng =~ s/the\s+//g;

#get rid of determiner


$por =~ s/(^(o|a)\s+)|(\s+(o|a)\s+)//g;

#get rid of determiner


$eng =~ s/to\s+//g;

#get rid of intransitive to


}


$eng =~ s/,\s/?/g;

#separate synonym translations in English


$por =~ s/,\s*/?/g;

$por =~ s/\-a\s+|\-o\s+/?/g;

$por =~ s/\s+\-\s+|\s+\-\s*|\s*\-\s+/?/g;

#separate synonym translations in Portuguese


$entry = "$eng\t$por\t$gram";

#tab separate parts of entry


$entry =~ tr/A-Z/a-z/;

#get rid of any capital letters


$entry =~ s/\?|\!|\.//g;

#get rid of punctuation
```

```
push(@entries, $entry);
#add entry to @entries list


}

open(DICOUT, ">$output") or die "can’t open $output: $!\n";


foreach my $entry(@entries) {

print $entry."\n";

print DICOUT "$entry\n";
#print to output
}


close(DICOUT);

print "$output written\n\n";
```

A.2 Construindo um Dictionário Parakanã-Poruguês

```
#!/usr/bin/perl -w
```

```
#formats pak-por-Ferreira.txt-0 to tabular format

#first column is the source

#second column is the part of speech for PanLex

#third column is the part of speech in the original dictionary

#remaining columns are targets (if there’s only one translation, there will be only one t


use warnings ’FATAL’, ’all’;

# Make every warning fatal.


use strict;

# Require strict checking of variable references, etc.


use encoding ’utf8’;

# Make Perl interpret the script and standard files as UTF-8 rather than bytes.


my $input = "pak-por-Ferreira-0.html";

my $output = "pak-por-Ferreira-1.txt";


open(DICIN, "<$input") or die "can’t open file: $!\n";

#open input


open (DICOUT, ’>:utf8’, ">$output") or die "can’t open file:$!\n";

#open output






my $entries;
```

```
#empty for now - will be filled in


my @entries;

#also will be filled in




while (my $line = <DICIN>) {


if ($line =˜ m/<body>/) {

#if the line contains the entries; all entries are in a single line


$entries = $line;


$entries =˜ s/\<\/style\>\<\/head\>\<body\>\<h1\>A  \- a\<\/h1\>//g;

#get rid of beginning


@entries = split(/\<p class=\"s1\"\>|<p class=\"s2\"\>\<span

class=\"s1\"\>|\<h2\>\<span class=\"s1\"\>/, $entries);

#each item in the list is one entry

}




}
```

```
foreach my $entry(@entries) {


my $source;

#the source word


my $targ;

#the target translation


my $gram;

#the part of speech in the dictionary


my $wc;

#word class for PanLex




##source and part of speech ##


if ($entry =~ m/^([^\s\<]+)[\s\<]+[^A-Z]+(Morf)?[^A-Z]+

(Variante)?[^A-Z]+(([A-Z][^\s\<\>\.]*\.?\s*)+([ab]?))/)

#regular expression matches the entry, with parentheses around

the source, the part of speech, and the target translation

{


$source = $1;

#the source word
```

```
$gram = $4;



if ($source =~ m/\-/) {
$wc = "affx";
}
elsif ($gram =~ m/Prprio/) {
$wc = "name";
}
elsif ($gram =~ m/^N/) {
$wc = "noun";
}
elsif ($gram =~ m/Aux/) {
$wc = "auxv";
}
elsif ($gram =~ m/^V/) {
$wc = "verb";
}
elsif ($gram =~ m/Adv/) {
$wc = "advb";
}
elsif ($gram =~ m/Pron/) {
$wc = "pron";
}
elsif ($gram =~ m/Conj/) {
$wc = "conj";
}
```

```
elsif ($gram =~ m/Determ/) {

$wc = "detr";

}

else {

$wc = "misc";

}




##translation##


if ($entry =~ m/[0-9]\)[^A-Z\<\>]+\./) {

#if there are multiple translations, with differing meanings (but same part of speech)


while ($entry =~ /([0-9]\)[^A-Z\<\>\.]+)\./g) {


$targ .= $1;


}


$targ =~ s/[2-9]\)\s*/?/g;

#add delimiter



$targ =~ s/1\) //g;

#get rid of remaining numbered definitions
```

```
}

elsif ($entry =~ m/[^b]\>\s*([^A-Z\<\>]+)\.\s*\</) {
#if there is only one meaning
#find the target translation using a regular expression
$targ = $1;


}
else {
$targ = "NA";
}

if ($targ =~ m/[^NA]/) {

$targ =~ s/\s+/ /g;
#get rid of extra spaces

$targ =~ s/^\s+|\s+$//g;
#get rid of leading and trailing spaces

$targ =~ s/\&\#39\;/’/g;
#change to apostraphe

$targ =~ s/\,\s*|;\s*/?/g;
#separate multiple translations that are synonyms
```

```
$targ =~ s/\.|\-//g;

#get rid of hyphens and punctuation


$source =~ s/\&\#39\;/’/g;

#change to apostrophe


$source =~ s/\.+|\-+//g;

#get rid of hyphens and punctuation


$source =~ s/^\s+|\s+$//g;

#get rid of leading and trailing spaces


$gram =~ s/\s+/ /g;

#get rid of extra spaces


$gram =~ s/^\s+|\s+$//g;

#get rid of leading and trailing spaces


print DICOUT "$source\t$gram\t$wc\t$targ\n";

#print to output

}




}

}
```

```
# close file

close(DICOUT);


print "$output written\n\n";
```

A.3 Mono 2000-item digital wordlist: Presentation form

```
#!/usr/bin/perl -w


#formats eng-fra-mnh-Olson-1.txt to tabular format

#first column is the source

#second column is the part of speech for PanLex

#third column is the part of speech in the original dictionary

#remaining columns are targets (if there’s only one translation, there will be only one ta


use warnings ’FATAL’, ’all’;

# Make every warning fatal.


use strict;

# Require strict checking of variable references, etc.


use encoding ’utf8’;

# Make Perl interpret the script and standard files as UTF-8 rather than bytes.


my $input = "eng-fra-mnh-Olson-1.txt";

my $output = "eng-fra-mnh-Olson-2.txt";
```

```
open(DICIN, '<:encoding(utf8)', "$input") or die "can't open file: $!\n";

#open input


open (DICOUT, '>:encoding(utf8)', "$output") or die "can't open file:$!\n";

#open output



my @entries;

#will be filled in


my $num = "";


my $SILnum = "";


my $eng = "";


my $fra = "";


my $orth = "";


my $wc = "";





while (my $line = <DICIN>) {
```

```
if ($line =˜ m/\<td.*\>([0-9]+)\./) {


if ($num) {

#an empty string is false, so will only execute if the number is filled


my $entry = "$num\t$eng\t$fra\t$SILnum\t$orth\n";



push (@entries, $entry);

}


$num = "";


$SILnum = "";


$eng = "";


$fra = "";


$orth = "";


$wc = "";



$num = $1;
```

```
}

elsif ($line =~ m/\<td.*\>\&nbsp;([0-9]+)/) {
$SILnum = $1;

}

elsif (($line =~ m/\<td\s*width\=\"80\"\>(.+)\</) && ($eng eq "")) {


$eng = $1;

$eng =~ s/;\s*/?/g;
#separate synonomous definitions

if ($eng =~ m/(.+)\(([a-z]|[a-z]+\.)\)/) {
$eng = $1;
$wc = $2;

print $wc."\n";
}


}

elsif (($line =~ m/td\s*width="80"\>([^\<]+)/) && ($fra eq "")) {
```

```
$fra = $1;


$fra =˜ s/;\s*/?/g;

#separate synonomous definitions


}

elsif (($line =˜ m/td\s*width=\"110\"\>\&nbsp;([ˆ\[]+)\</) && ($orth eq "")) {


$orth = $1;


$orth =˜ s/;\s*/?/g;

#separate synonomous definitions

}

}


foreach my $entr(@entries) {


print DICOUT "$entr";


}


close(DICOUT);
```

A.4 Comparative Lexicon of Dogon Languages

```
#!/usr/bin/perl -w

#tabularizes djm-dtt-dbu-eng-fra-Heath.txt-0 to tabular format
#0th column is the target in English
#1st column is the target in French
#2nd column is the Jamsay translation
#3rd column is the Gourou translation
#4th column is the Nanga translation
#5th column is the Beni translation
#6th column is the Walo translation
#7th column is the Tabi translation
#8th column is the Najamba translation

#some translations may be blank
#translations and descriptions may have more than one, synonomous, translation
#also may have parentheses, which can be characterized
as further descriptions, but not synonyms


use warnings ’FATAL’, ’all’;
# Make every warning fatal.

use strict;
# Require strict checking of variable references, etc.

use utf8;
# Make Perl interpret the script and standard files as UTF-8 rather than bytes.
```

```
my $input = "eng-fra-djm-dwl-Elders-0.txt";

my $output = "eng-fra-djm-dwl-Elders-1.txt";


open(DICIN,  '<:utf8', "$input") or die "can't open file: $!\n";

#open input


open (DICOUT,  '>:encoding(utf8)', "$output") or die "can't open file:$!\n";

#open output



my $entry = "";


my @entries = ();



while (my $line = <DICIN>) {


$line =~ s/\n*//g;


if ($line =~ m/\<td/) {

#if the line contains data


$line =~ s/\s*/ /;

#get rid of extra spaces
```

```
$line =~ s/^\s+//;
#get rid of leading spaces at line beginning

$line =~ s/\s+$//;
#get rid of trailing spaces at line ending

$line =~ s/\t/ /g;
#get rid of any tabs, replacing with space

$line =~ s/\./tre/g;
#change abbreviation to full word (in French)

$line =~ s/, |\\| or | ou |; /?/g;
#separate synonymous translations

$entry .= "$line\t";
#add the data to the entry, tab-separating from following data
}


if ($line =~ m/\<\/tr/) {
#if the end of an entry has been reached
#clean up entry


$entry =~ s/\<(\/)?td\> *//g;
$entry =~ s/\<td class\=\"left\"\> *//g;
$entry =~ s/\<\/?tr\>\s+//g;
```

```
$entry =~ s/\<tr style=\"background-color: \#EEEEEE;\"\>//g;
#get rid of formatting info


$entry =~ s/\s+\t\s+/\t/g;
#get rid of leading and trailing spaces between columns


$entry =~ s/\&nbsp\;//g;
#get rid of blanks (in the table in the data source,
#blanks are written as &nbsp;)


$entry =~ s/\t$//;
#get rid of final tab



$entry =~ s/\[/\(/g;
#replace angled bracket with parentheses


$entry =~ s/\]/\)/g;
#replace angled bracket with parentheses


#if ($entry =~ m/(\([^\)]+)?/) {
# print "$1\n";
#}


$entry =~ s/(\([^\(\)]+)?/$1, /g;
#get rid of synonym separator within parentheses
##Add entry to list##


if ($entry =~ m/\t/) {

#make sure that $entry contains columns


push (@entries, $entry);

#put entry into the list of entries


$entry = "";

#clear entry


}



}




}




foreach my $entri(@entries) {


print DICOUT "$entri\n";

#print to output

}
```

```
close(DICOUT);

print "$output written\n\n";
```

B Code for comparing multiple normalization parameters

The following code was written to compare the output of a variety of normalization parameters against a gold standard. The code outputs true positives, false positives, true negatives, and false negatives. This code was used to determine if there was an ideal setting for the parameters for the normalization script.

```
#!/usr/bin/perl -w

#compares normalized expressions in one partially formatted file to a gold standard
# (which is hand-corrected)
#returns the number of false positives, true positive, false negatives, and true negatives

use warnings 'FATAL', 'all';
# Make every warning fatal.

use strict;
```

```
# Require strict checking of variable references, etc.

use encoding 'utf8';
# Make Perl interpret the script and standard files as UTF-8 rather than bytes.

my $expGold = "por-eng-Brasileiro-6-Gold.txt";
my $exp1 = "por-eng-Brasileiro-6-1Threshold.txt";
my $output = "compared.txt";

my $expColNum = 0;
#which column the expressions are in

open(GOLD, "<$expGold") or die "can't open file: $!\n";
#open input

open(EXP1, "<$exp1") or die "can't open file: $!\n";
#open input



open (DICOUT, '>:utf8', ">$output") or die "can't open file:$!\n";
#open output

#hashes containing expressions
my %gold;
my %exp1;

#hashes containing false positives, true positives, and false negatives
```

```
my %falsep;
my %truep;
my %falsen;

#counts for true positive, false positives, false negatives, and the total,
#for calculating true negatives
my $truepCount = 0;
my $falsepCount = 0;
my $falsenCount = 0;
my $totalCount = 0;
my $truenCount = 0;


while (my $line = <GOLD>) {

$totalCount += 1;

chomp $line;

if ($line =˜ m/?exp?/) {



my @cols = split(/\t/, $line);
#split line into columns

my $exCol = $cols[$expColNum];
#pick out column containing expressions
```

```
my $exp;

my $ex;

if ($exCol =˜ m/?exp?([ˆ??]+)?ex?([ˆ??]+)/) {


$exp = $1;
$ex = $2;

$gold{$exp} = $ex;

}

}

}

while (my $line = <EXP1>) {

chomp $line;

if ($line =˜ m/?exp?/) {

my @cols = split(/\t/, $line);
#split line into columns
```

```
my $exCol = $cols[$expColNum];
#pick out column containing expressions


my $exp;


my $ex;


if ($exCol =˜ m/?exp?([ˆ??]+)?ex?([ˆ??]+)/) {


$exp = $1;
$ex = $2;


$exp1{$exp} = $ex;


}


}


}




while (my ($key, $value) = each %exp1) {


if (exists ($gold{$key})) {
#a true positive
$truep{$key} = $value;
}
```

```
else {

#a false positive

$falsep{$key} = $value;

}



}


while ( my ($key, $value) = each %gold) {


if (exists ($exp1{$key})) {

#a true positive

$truep{$key} = $value;

}

else {

#a false negative

$falsen{$key} = $value;

}



}



print DICOUT "True positives\n\n";




while(my ($key, $value) = each %truep) {
```

```
print DICOUT "$key\t$value\n";

$truepCount += 1;
}



print DICOUT "\n\n\nFalse positives\n\n";

while (my ($key, $value) = each %falsep) {

print DICOUT "$key\t$value\n";

$falsepCount += 1;

}

print DICOUT "\n\n\nFalse negatives\n\n";

while (my ($key, $value) = each %falsen) {

print DICOUT "$key\t$value\n";

$falsenCount += 1;

}
```

```
$truenCount = $totalCount - ($truepCount + $falsepCount + $falsenCount);


print "$truepCount true positives, $falsepCount false positives,
$truenCount true negatives and $falsenCount false negatives\n";




close(DICOUT);
```

C Documentation for PanLex

The documentation included below was written for editors of PanLex, for use with the existing system.

PanLex Documentation

Abstract

All sources for PanLex must be normalized. This document explains the process of locating a source in the PanLex database that has not yet been entered, normalizing and formatting the source for PanLex, and finally adding it to the PanLex database.

Accessing the source

All the digitized sources are stored at a remote station, which can be accessed by going to panlex.net/panlex. Sources follow a naming convention as follows: the three letter ISO code for the source language, the three letter ISO code for the target language, the author's last name. If there is no known author, then the initials of the title are used (for example, EFD for English-French Dictionary). For example, for the English-French dictionary, the source would be named fra-eng:EFD. Since there is a length limit in PanLex, but not in panlex.net/panlex → Sources, the names are not always identical.

1. Go to panlex.net/panlex
2. Click on Sources
3. Click on queued (these are the sources that have not yet been entered)
4. For each source, there's typically one file, with the data, and a secondary folder, which will have extras (usually meta-data).
5. Click on the file to get access to the source data. This is typically what you will be normalizing.

If the file is in .pdf format, it can be converted to .html as follows:

1. Open the PDF file in Acrobat Professional.
2. Extract the dictionary pages into a new window.
3. Save that part as a new PDF file.
4. Save the PDF file as HTML (from Acrobat).
5. Additionally, use pdftotext, using Acrobat's HTML export. This will save as a text file
6. Compare the text file to the .html file, and use whichever one has a better output.

Having the file in .html format is desirable, as it provides extra formatting tags which can be used to parse the lines into appropriate sub-parts (for example, the source word, the target translation, part of speech, etc.).

Editing the source

Sometimes it will be necessary to edit the PanLex catalog record for the source, either before or after formatting the data for entry to PanLex. This is done in PanLem.

1. Go to panlex.org
2. Click on Try It
3. Click on PanLem (in the body of the text)
4. Click on Source → Edit
5. Sign in with your username and password
6. You will see a list of all the sources. Click on the one you want.
7. You will then see a table containing meta-data associated with the source.
8. Click on the button under 'change' for the one you want to edit, and edit appropriately.

9. When done editing, click on Submit at the bottom of the page.

What the values in the righthand column of the first table stand for is described below:

name This is the name assigned to the source, following the naming convention described in Accessing the Source.

World Wide Web The url where the source was obtained, if there is one.

ISBN ISBN if there is one

author The author or authors of the source

title The title of the source (not the name given for identification in PanLex)

publisher The publisher of the source

year The year of publication

good-number This number refers to the quality of the source, with rating 0-9. The higher the number, the better the source is judged to be. This number helps PanLex rate how likely a given translation is, based on its source.

source-number Any number the editor wants to assign for the source.

fact-other Any other facts that are of interest about the source. Do not put editorial comments here; those go under fact-other in the next table down (the editors table, which only the editor sees).

permission-kind The kind of permission for use of the source. For example, creative commons license

right-text Quoted from the source itself, if there is a line about how the source may be used.

right-person The person to contact with questions about rights of usage.

right-email-address The email address of the person to contact about rights of usage.

The second table is viewable only by editors, and contains more editorial-type meta-data. What the values in the righthand column of the second table stand for is described below:

good-number-edit-necessary-1/0 refers to whether or not the quality of the resource has been determined. (0=determined, 1=undetermined). This will be changed once the source has been edited, and an appropriate value for good-number has been entered).

file-difficult How difficult the file is likely to be to edit, from 0-9 with 9 being most difficult.

file-submit-necessary-1/0 whether the [normalized/processed/converted] source file needs to be submitted (0=no, 1=yes).

expression-edit-necessary-1/0 whether a submitted file needs further editing

expression-edit-done-language-(aaa-bbb-ccc) which of the languages in the source are finished (require no further editing)

file-address name of the folder, or directory, containing the files of the source. Typically, this is the same as the name, except that the colon is replaced with a hyphen.

fact-other any further editorial comments

The third table, language, contains the languages in the source file. The fourth table, file-kind, shows the kind(s) of file of the source (pdf, xml, etc). You can click on edit-person to update the identity of the editor. The first time, you would be adding the identity of the editor (most likely you); you can later add or delete other users who are entitled to edit the data from the source.

Formatting the source

Whatever kind of formatting the source is in, it must be converted to a form that PanLex can read. PanLex files have a standard format, with one entry taking up several lines. However, it is also possible to put the data into tabular format, and then use a number of existing scripts to transform this into the PanLex format. Often this is easier, and is what is described here.

The tabular formatting is relatively simple:

- each line contains one entry
- each tab-separated column contains one piece of information about the entry, which will vary depending on what information the source contains. For example, the first column might be one language, the second a translation in another language, the third a translation in a third language, the fourth the part of speech, etc.
- order of the columns does not matter, with one constraint: any columns that provide additional data (word class, metadata, etc) about expressions in a particular column must immediately follow that column.
- do not put a heading; you can define which column is which using the shell script

Most formatting scripts are written in perl. The most recent and general past scripts can be found in panlex.net/panlex → tools → tabularize. Other scripts can be found in panlex.net/panlex → sources → used → aaa-bbb-Name → secondary. It is a good idea to make your scripts as general (and readable) as possible, so that they are re-usable for any updates of the sources.

Once you have the data in tabular format, you can use the shell script main.sh to convert to PanLex format. main.sh (and accompanying perl scripts) can be found at panlex.net/panlex → tools → serialize.

Serializing

To use the serialization scripts found on panlex.net/panlex, make sure you have downloaded the latest versions of all scripts. Open main.sh, and look through the available scripts included. Not all will be needed for a given source, and which scripts are needed will depend on the source and the kinds of information it has.

The use of each script is documented in the comments of main.sh. General options that will need to be changed for each script are to replace aaa-bbb-Author with the particular name of the file being processed. For example, if the file were named pak-por-Ferreira-1.txt, then aaa-bbb-Author should be replaced with pak-por-Ferreira. The number at the end of a file name is what is referred to in the comments of main.sh as *version of the input file*. After deleting or commenting out unneeded scripts, go through each script and alter this number such that for the first, the number matches the number at the end of the input file name (in the example case, this number would be 1). Each number after should increase by 1, as each script will produce an output file with this number augmented by 1. The output file of one script is the input file for the next, so it is important to make sure that these numbers are entered correctly.

The purpose of each perl serializing script is described briefly below:

apostrophe.pl This is to ensure that apostrophes are entered in a standard way into PanLex. There are actually many kinds of apostrophes in Unicode. If a language is being entered into PanLex for the first time, then a decision needs to be made as to which apostrophe will be used for that language. Descriptions of three kinds of apostrophes and how they are most commonly used are described in the comments for this script.

extag.pl This script tags expressions as expressions recognizable by PanLex. This needs to be done for any file, as an entry without any expression is not a legitimate entry in PanLex. Additionally, expressions are split into either synonymous translations or separate expressions (meaning change), as appropriate.

exdftag.pl This is useful if the tabularized file sometimes contains definitions in parentheses, or longer expressions which are more appropriately categorized as definitions.

dftag.pl This is useful if the tabularized file contains a particular column of definitions

mitag.pl This is useful if the tabularized file contains a particular column of meaning identifiers (typically a number; useful for finding the entry in the original source, as the meaning identifier can be matched).

wctag.pl This is useful if the tabularized file contains a particular column of word class. Word class from the original source will be transformed, using this script, to the appropriate PanLex name (more specific information, such as the particular noun class of a word, will be tagged as metadata). If you suspect that the word classes identified in the source are not at all standard, it may be necessary to alter wctag.pl to include them. Standard word classes are defined in the hash *my %wc*, with the source name of the word class as the key and the PanLex name for it as the value.

mdtag.pl This is useful if the tabularized file contains a particular column of metadata, that cannot be described as a definition, a meaning identifier, a word class, or a domain. Alter the metadata tag as appropriate, so that it describes what kind of information it is. The default is *gram*.

dmtag.pl This is useful if the tabularized file contains a particular column of the domain (for example, *plant*, *tool*, etc).

normalist.pl This is one of two normalizing scripts. Normalization will be discussed more in Normalization.

mnsplit.pl This is useful if the tabularized file contains multiple meanings in a single entry (ie, not synonymous translations). This script will split these multiple meanings into separate entries. An example of multiple meanings for a word in French, *pâte*, in English are *dough*, *pasta*. While the spelling in French is the same for these two words, they do not have the same meaning.

wcshift.pl This is useful if word classes are prepended to something, but otherwise conform to PanLex specifications.

normalize.pl This is the second of two available normalizing scripts. Which to use will depend on the format of the input file.

out-simple-0.pl This is the first of several final scripts, which will transform the tabular file into the style of PanLex. Use out-simple-0.pl if the source has no other data besides expressions (making it a simple file) and there are more than two languages.

out-simple-2.pl Use out-simple-2.pl if the source is bilingual, has no other data besides expressions, and the first column contains only a single expression (with no synonyms).

out-full-0.pl Use out-full-0.pl if the source has data in addition to expressions (word class, definitions, etc), and contains more than two languages

out-full-2.pl Use out-full-0.pl if the source is bilingual, has data in addition to expressions, and the first column contains only a single expression (with no synonyms).

Normalization

Normalization of expressions is very useful, since if a single expression (that is, an expression with the same meaning and form) is entered in two ways into PanLex, PanLex will treat the single expressions as two different expressions. In turn, this limits PanLex's ability to link expressions between languages, based on a common translation into a third language for each.

Both normalist.pl and normalize.pl match expressions to what is already in PanLex, suggesting alternate expressions, or not, depending on the score parameters. Thus, it is advisable to normalize only if the language of the column to be normalized is already very well-represented in PanLex.

Even so, there are likely to be normalization errors, where an expression is altered to another expression with a higher score. For example, in Portuguese, there are two different words, *avô* and *avó*, which are very close in form. Parameters in normalize.pl should be adjusted to prevent PanLex from normalizing *avó* to *avô* or vice versa, since one or the other will have a higher score (scores of words are based on the number of different sources that they appear in, and each source's quality estimate, in PanLex). This may require some experimentation, as each source will differ as to which are the best parameters.

Even so, perfect normalization is unlikely from this algorithm alone. To get the best quality data, it may be helpful to correct the resulting file by hand (expressions to be altered/deleted are tagged *exp*, making it easier to search for them).

In all cases, it is helpful to have some knowledge of the language(s) whose expressions are to be normalized, as this will help with spotting errors in normalization.

Uploading the normalized source

Once a source has been formatted for PanLex, the last step is to upload it. This can be done at panlex.org → Try It → PanLem → file - submit. Enter username and password, then choose the appropriate file name. This may or may not exactly correspond to the one from panlex.net/panlex. Choose whether a simple file (out-simple-n.pl was used), whole file (out-full-n.pl was used), or xml (not discussed here, but useful if the original source file was in xml) is being submitted.

Before submitting, find the file (aaa-bbb-Author-final.txt) and click *Check*. This will show any impossible lines (lines that PanLex can't read or doesn't allow). It will not check for other errors, however, so look over the file before submitting it and see if there are normalization errors, mislabeled expressions or definitions, etc.).

When it is reasonably certain that there are no errors, click on *approve - more* or *delete - replace - whole* if a previous submission is to be replaced.