Defining a Probabilistic Translation Dictionaries Algebra

Alberto Simões[1], José João Almeida[2], and Nuno Ramos Carvalho[2]

[1] Centro de Estudos Humanísticos
Universidade do Minho
ambs@ilch.uminho.pt
[2] Departamento de Informática
Universidade do Minho
{jj,narcarvalho}@di.uminho.pt

Abstract. Probabilistic Translation Dictionaries are around for some time, but there is a lack of a formal definition for their structure and base operations. In this article we start by discussing what these resources are, what researchers are using them for, and what tools can be used to create this them. Including a formal definition and a proposal for a XML schema for dictionaries interchange. Follows a discussion of a set of useful operations that can be performed over probabilistic translation dictionaries, like union, intersection, domain restriction and composition. Together with this algebra formalization some insights on the operations usefulness and application are presented.

1 Introduction

Multilingual research is usually supported by translation dictionaries, some kind of mapping between words or terms written in different languages.

Translation dictionaries are not easy to find, especially for languages with few resources available. This lead to the development of methods for automatically creating translation dictionaries using as base parallel corpora [12]. There are different algorithms available to extract bilingual mappings, like Kvec [3], Kvec++ [20], Twente-Aligner [7], NATools [18] or Giza++ [13].

Although these algorithms compute different types of resources, they share a baseline: the possibility to extract probabilistic translation dictionaries from the resulting resources. A probabilistic translation dictionary associates, for each word from a language $\mathcal{L}_A$, a set of probable translations[3] in a language $\mathcal{L}_B$, together with a translation probability measure. The term Probabilistic Translation Dictionaries (PTD) is used to refer to these dictionaries. They can be seen as some kind of fuzzy translation dictionaries. Section 2 discusses briefly these objects, defining their formal structure.

PTD have been used for different tasks by different authors: Guinovart and Fontenla [6] describe a method to bootstrap a conventional translation dictionary from a PTD. A

[3] While these algorithms return high probable translations it is important to notice that they just assume that words and respective translations have a high probability to co-occur on parallel corpora. This means that some suggested translations may be imprecise or not correct at all.

PTD was created and a try-and-error approach was used to define a translation probability threshold for filtering purposes. The filtered dictionary was used for manual validation. Caseli et al. [2] use PTD for a similar task: bootstrapping a machine translation dictionary. In this case the manual validation was not necessary. Simões and Almeida [14] use PTD as a mechanism to present parallel concordances and guessing translations of searched terms, highlighting them when presenting the search result. Simões and Almeida [15] and Guinvart and Simões [5] present methods to extract bilingual terminology, using translation patterns and PTD for translations alignment. Simões and Almeida [16] also use PTD as a mechanism to align chunks of text when creating translation examples. Kraaij [9] uses PTD for cross-language information retrieval.

While PTD are being useful for these researchers, there is lack of a concrete definition of the PTD object (making them harder to share) and a concrete definition of PTD operations (helping researchers to manipulate and enrich these resources)[4].

In this article we, (i) define a formal model for PTD, and propose a simple XML schema for PTD interchange (section 2); and (ii) formalize a subset of a PTD algebra, defining functions and operators to add, filter, intersect, subtract and compose these dictionaries. They will be presented together with a discussion on their usefulness (section 3). Examples will be from word-to-word dictionaries, but the presented methods are easily extensible for multi-word term dictionaries.

2 Probabilistic Translation Dictionaries

PTD are extracted automatically from bilingual corpora, being it aligned [18] or simply comparable [4] corpora. Processing a bilingual corpus results in a pair of PTD: one, mapping words from the corpus source-language to its target-language, and another, mapping words from the corpus target-language to its source-language[5].

When talking about a PTD we are referring to one of these dictionaries. When needing to refer to the PTD that maps words in the opposite direction we will use the *inverse dictionary*.

2.1 PTD Definition

Before defining the formal structure of a PTD, let us analyze a real PTD entry example, extracted from a Portuguese/English dictionary:

$$\mathcal{T}\,(\text{codificada}) = \begin{cases} \text{codified} & 62.83\% \\ \text{uncoded} & 13.16\% \\ \text{coded} & 6.47\% \\ \dots & \end{cases}$$

[4] The operations defined in this article are implemented and available freely as a Perl module: `http://search.cpan.org/dist/Lingua-PTD/`

[5] Refer to [7] for further discussion about having a pair of directional dictionaries instead of just one non directional dictionary.

The common way of reading this structure is: "the Portuguese word *codificada* can be translated by the English words *codified*, *uncoded* and *coded*". The truth is that the algorithms that extract this data do not deal with translations, but correlations between words. Therefore, the correct reading would be: "the Portuguese word *codificada* is highly correlated with the English words *codified*, *uncoded* and *coded*", and the percentages are correlation measures.

Nevertheless, given the main goal for these resources, we will opt to use the first reading, and considerate the correlation measures as translation probabilities.

Each dictionary, maps words from a source-language $\mathcal{L}_S$[6] to a set of possible translations on a target-language $\mathcal{L}_T$. For each possible translations $w_t^i \in \mathcal{L}_T$ for a word $w_s \in \mathcal{L}_S$, we have an associated probability measure, that will be read as $\mathcal{P}\left(w_t^i \in \mathcal{T}\left(w_s\right)\right)$, where $\mathcal{T}$ stands for the translation function.

Together with this lexical information we keep track of the occurrence count for each word (usually for the source language, as the count for the target language can be found in the inverse dictionary).

Some meta-data should also be available. Two main meta-data values are required: the involved languages. Other optional meta-data information can contain: the creation date, used tool, source corpus, etc.

We will define formally the PTD type, from a language $\mathcal{L}_A$ to a language $\mathcal{L}_B$, as[7]:

$$\mathrm{PTD} = \mathrm{MetaData} \times \mathrm{Dictionary} \tag{1}$$

$$\mathrm{MetaData} = \mathrm{ID} \rightharpoonup \mathrm{Value} \tag{2}$$

$$\mathrm{Dictionary} = \mathcal{L}_A \rightharpoonup \mathbb{N} \times (\mathcal{L}_B \rightharpoonup [0..1]) \tag{3}$$

Equation 1 defines a PTD as a pair, where the first element is of type MetaData, and the second element of type Dictionary. Equation 2 defines the MetaData element as a finite function from identifiers to random values. Although not completely clear above, the "source" and "target" languages identifiers are present in the meta-data portion of the PTD. Finally, equation 3 defines the main structure of the dictionary itself: a finite function from words in language $\mathcal{L}_A$ to a pair of elements; where the first element is the total number of occurrences of the word in the source corpus, and the second element is a finite function mapping possible translations in language $\mathcal{L}_B$ to their translation probability.

Note that for efficiency purposes some systems discard translations whose probabilities are too small (for instance, ≤ 0.0005), or when the number of translations for some word is greater than a predefined number, storing only the k more probable translations. This leads to the fact that it is usual that the translation probabilities for a specific word do not sum up to 1.

2.2 PTD Interchange

Different tools store their probabilistic translation dictionaries in different formats, ranging from simple text files, serialization formats (some of them specific to some program-

[6] We will use $\mathcal{L}$ to represent a language, as the set of words accepted by that language.

[7] Different algorithms can extract these dictionaries with different kind of probabilities. Nevertheless, it should be possible to normalize these probabilities into percentage values.

ming languages) or even relational databases (usually non-server databases systems like SQLite). Nevertheless, usually these formats only store the dictionary itself, relegating some of the meta-information to the file name, or the file position in the file system. This leads to interchange problems.

Our proposal, following other formats widely used in computational linguistics, like TEI (Text Encoding Initiative), TMX (Translation Memory Exchange), TBX (Term Base Exchange), XLIFF (XML Localisation Interchange File Format) and others, is based on XML (Interchange Markup Language).

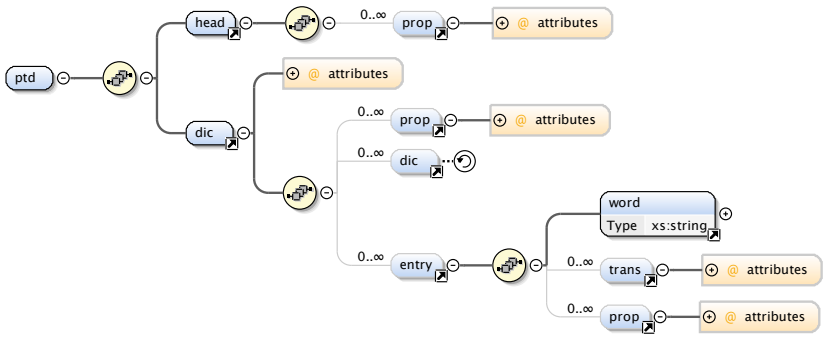

Fig. 1. Probabilistic Translation Dictionary Interchange Format.

Figure 1 shows our proposed format[8] to store a PTD. The structure is straightforward: a PTD is composed of a header, containing some meta-information represented by property elements, that map a key (element attribute) to some value. The body is a dictionary that is composed by sub-dictionaries, optional properties, and a list of entries. Each dictionary has source and target languages as element attributes. In its turn, each entry has a word in the source language, optional properties, and a set of translations, whose probability is encoded as an element attribute.

The structure was designed to be extensible, making it easy to add property elements to most levels of the dictionary. These properties can be used to store word morphological properties, their domain, etc. Also, instead of coding directly into the document structure the occurrence count for each word, we encourage users to encode that information as a property. This way there is no need to define a set of elements that would be empty when this information is not available.

3 PTD Algebra

There are some relevant operations that can be performed with Probabilistic Translation Dictionaries. Some of them are trivial to define and implement, and their use is mostly

[8] PTD schema is available at `http://natura.di.uminho.pt/PTD`.

standard, but others need to be defined carefully, and can be applied for some surprising results.

A special detail must be made clear: we will use standard mathematical operation names and symbols, but their meaning might not be exactly the expected one. As a rule of thumb, remember that these dictionaries are, in fact, a special case of multisets (bag).

3.1 Dictionary Union (or sum)

The PTD Union (or sum) is the base of the NATools[9] algorithm when dealing with large corpora, this makes the tool scalable. Large corpora are spliced in smaller chunks that are processed independently, and the extracted dictionaries are added at the end.

The union has a single prerequisite: both dictionaries should be for the same source and target languages.

When formalizing this operation our main goal is to give different weights to translation probabilities, accordingly with the size of the corpora the dictionary was based on, and the number of occurrences of those words on that specific corpora.

When adding two entries for a specific word, the following rules are used:

- the occurrence count for the resulting entry is the sum of the separate occurrences of the word in each one of the dictionaries being summed;
- the resulting translations set is the union of the translations from both dictionaries;
- for each translation, the probability is computed by a weighted mean, as follows[10]:

$$\frac{\mathcal{P}\left(w_{\mathcal{L}_B} \in \mathcal{T}_{d_1}\left(w_{\mathcal{L}_A}\right)\right)\mathcal{O}_{d_1}\left(w_{\mathcal{L}_A}\right)\mathcal{S}_{d_2} + \mathcal{P}\left(w_{\mathcal{L}_B} \in \mathcal{T}_{d_2}\left(w_{\mathcal{L}_A}\right)\right)\mathcal{O}_{d_2}\left(w_{\mathcal{L}_A}\right)\mathcal{S}_{d_1}}{\mathcal{O}_{d_1}\left(w_{\mathcal{L}_A}\right)\mathcal{S}_{d_2} + \mathcal{O}_{d_2}\left(w_{\mathcal{L}_A}\right)\mathcal{S}_{d_1}} \tag{4}$$

Given this operation relevance when processing big corpora (as tools depend on it for a *map-reduce* approach), we did some experiments comparing if the result of a PTD computed for a specific corpus is similar with the PTD obtained by summing up PTDs computed for n chunks of that same corpus:

$$ptd(c) \stackrel{?}{=} ptd(c_1) \cup ptd(c_2) \cup \ldots \cup ptd(c_n) \quad \text{for a corpus} \quad c = c_1 \cdot c_2 \cdot \ldots \cdot c_n \tag{5}$$

This experiment was done using JRC-Acquis[19] corpus for the English–Portuguese pair, that contains 1 293 815 translations units. This corpus was also divided in ten chunks: the first nine chunks with 130 000 translations units each, and the last chunk with the remaining 123 815 translation units.

Each one of these ten corpora were aligned independently, but we also aligned the full original corpus in a single run (without using the *map-reduce* approach).

The final dictionaries were then filtered using the same heuristics (removing non-words and translations with probabilities below 0.005%) and compared. Table 1 shows the differences on the obtained dictionaries.

[9] Available from `http://search.cpan.org/dist/Lingua-NATools/`

[10] In this and future formulae, $w_{\mathcal{L}_A}$ represents a word, $w_{\mathcal{L}_A} \in \mathcal{L}_A$; $\mathcal{O}_{d_i}\left(w_{\mathcal{L}_A}\right)$ is the number of occurrences of word $w_{\mathcal{L}_A}$ using dictionary d_i; and $\mathcal{S}_{d_i}$ is the sum of occurrences for all

Table 1. Comparison between two PTD: first extracted from a single corpus file, and the second resulting of the sum of the extraction from ten chunks.

	$ptd(c)$	$\bigcup_i ptd(c_i)$
number of entries	220 155	219 895
average translations per entry	4.07	5.03
average of 1st trans. prob.	62%	52%
entries sharing best translation: 71%		

Table 1 summarizes the differences between these two dictionaries[11]. Analyzing the results, we can conclude that:

- the number of translations available per entry is higher in the second dictionary (the result of the union) because the extraction algorithm has limitations on the number of translations calculated. In order to maintain memory usage low only the eight more probable translations are extracted.
- the average probability for the best translation is lower for the union as the number of translations is higher. If we reduce the number of translations per entry to the same number as the one computed by the extractor, the probabilities are similar.
- nevertheless, for 29% of entries we have different best translations. This happens mainly for low occurrence words, that when aligned independently in the 10 chunks have yet lower occurrence counts.
  To understand better this problem we used a smoothed version of the Kullback-Leibler divergence [11] (equation 6, where P and Q are the translations distributions to compare, and P' and Q' are the ϵ smoothed distributions) to compare dictionaries entries accordingly with their probability distribution.

$$D_{KL}(P,Q) = \sum_w P'(w) \ln \frac{P'(w)}{Q'(w)}$$
$$\text{where} \tag{6}$$
$$P'(w) = \begin{cases} P(w) - \frac{\epsilon}{n} & \text{if } w \in P \\ \epsilon & \text{if } w \notin P \end{cases} \quad \text{and} \quad Q'(w) = \begin{cases} Q(w) - \frac{\epsilon}{n} & \text{if } w \in Q \\ \epsilon & \text{if } w \notin Q \end{cases}$$

  Table 2 shows some dictionary entries that have big differences when using Kullback-Leibler measure.
  This table shows that more different entries are entries with a low number of occurrences. This leads to a main problem: each chunk will have still lower number of occurrences, and therefore, the algorithm will have more difficulty succeeding in the alignment process. Nevertheless, the average Kullback-Leibler divergence for the full dictionary is of 1.296 (a relatively low value).

entries in dictionary d_i (that is, the corpus size). Therefore, $\frac{\mathcal{O}_{d_i}(w_{\mathcal{L}_A})}{\mathcal{S}_{d_i}}$ is a percentage value of the occurrence of the word $w_{\mathcal{L}_A}$ in the dictionary. This ratio is then used to weight the mean formula.

[11] The compared dictionaries have Portuguese as source-language, and English as target-language.

Table 2. High difference entries comparison.

word/occs.	using $ptd(c)$		using $\bigcup_i ptd(c_i)$	
Guiana	Guiana	100%	State	99%
(36)	—	—	—	—
Centro-Africana	African	100%	Central	100%
(35)	—	—	—	—
Carina	Carina	100%	EL	23%
(10)	—	—	Kyrgyzstan	14%
Maestro	Maestro	100%	NL	36%
(13)	—	—	H	9%
radioastronomia	astronomy	100%	radio	71%
(13)	—	—	pertaining	6%

Although we can argue that the union algorithm is not good enough, as the resulting dictionary is not exactly the same as the obtained when extracting directly from the full corpus, note that:

- for some huge corpora (and nowadays we have bigger and bigger corpus, as Europarl [8], JRC-Acquis [19] or even the newest DGT-Acquis[12]) the full alignment in memory is not possible for most machines. Using the *map-reduce* approach researchers are able to align these corpora and extract useful dictionaries, where without it, probably no dictionary would be available;
- in some situations there are different dictionaries obtained from different corpora that could be merged to create a bigger and, hopefully, better dictionary. While full-realignment of the concatenated corpora might give better results, it would take extra time, and the original corpora might not be available (for instance, it might be possible to get some corpora owners to extract and share the dictionary, but not the full corpus).

3.2 Dictionary Intersection

The definition of the intersection operation is not as complex as the union:

- the domains are intersected (therefore, removing any word not present in both dictionaries);
- for each kept word, the minimum number of occurrences is used;
- for probable translations, only the ones that are proposed as translations in both dictionaries are kept;
- regarding the word probability, again the minimum value is used.

This process returns a dictionary with fewer words, and fewer translations. Nevertheless, the maintained translations are highly probable as they occur in both dictionaries. Also, the associate probabilities not only decrease (as we are using the minimum), but also lose their significance as a probability (and become some kind of measure).

Intersection can be used for different tasks as, for instance:

[12] Available from `http://ipsc.jrc.ec.europa.eu/?id=783`

– compute the intersection of dictionaries obtained from different domain corpora, to compute the shared or *base* lexicon (that can be used later to subtract and obtain a specific domain dictionary);
– also, the intersection with a specific small hand-controlled dictionaries can be used to tune PTD extraction algorithms;
– the intersection function can be also used to compute semantic distances between entries (words).

3.3 Dictionary Domain Restriction and Subtraction

These two operations have high similarities, being the main difference the definition of the words to be kept, or to be removed from the dictionary.

Domain restriction is similar to dictionaries intersection: instead of intersecting the domains of two dictionaries, this operation intersects the domain of one dictionary with a set of words[13].

Domain restriction main application is the reduction of dictionary size. While the usual approach is to accumulate dictionaries creating bigger ones, with more certain, they grow quickly in size and may become heavy for some specific processes. Restricting the domain of a dictionary to the set of words occurring in the corpus to process may reduce drastically the dictionary size.

Domain subtraction is the complementary operation. Instead of restricting the dictionary domain to set of words, it removes from the dictionary all entries whose words appear in the supplied set of words.

This operation can complement the dictionary intersection as a method to extract terminology lists, or at least, detect main terminological differences. For example, after computing $PTD = PTD_1 \cap PTD_2$, we can subtract the PTD domain from PTD_1 to get a terminological dictionary PTD_1^T, that only contains entries for those words that are specific to PTD_1:

$$PTD_1^T = PTD_1 \setminus (PTD_1 \cap PTD_2) \tag{7}$$

However, using this approach, all common entries to the two dictionaries that have completely different translations sets are lost. For these cases, the distance measure defined earlier is a better approach.

3.4 Dictionary Totalization

There are some operations (like the intersection) that remove some possible translations from dictionary entries. The resulting dictionary will have entries whose translation probabilities do not sum to 1 (100%). In fact, this is even true when obtaining a dictionary using some tools that limit the number of extracted probable translations.

[13] Although domain restriction and domain subtraction are defined in this article as operating on a PTD and a set of words, their implementation supports domain restriction and domain subtraction between two PTD (where the second is converted to the set of words from its domain).

For some situations this is not a problem. But when comparing dictionaries, for instance, probabilities take a relevant role. These probabilities can be recomputed (or totalized), forcing the sum to be one, and computing the remaining probabilities accordingly.

3.5 Dictionary Composition

Composition is another interesting operator that can be used to obtain very curious results. Although a dictionary is not a proper map, or a function, we can define some way for it to behave as one:

$$f_{D(\mathcal{L}_A \to \mathcal{L}_B)} : \mathcal{L}_A \longrightarrow (\mathcal{L}_B \rightharpoonup [0..1]) \tag{8}$$

Given this functional behavior, we can define the composition for these functions (similar to [10]):

$$f_{D(\mathcal{L}_A \to \mathcal{L}_B)} \circ f_{D(\mathcal{L}_B \to \mathcal{L}_C)} = f_{D(\mathcal{L}_A \to \mathcal{L}_C)} \tag{9}$$

Each value in the range of this function

$$(w_{\mathcal{L}_C}, \mathcal{P}(w_{\mathcal{L}_C})) \in \operatorname{ran}\left(f_{D(\mathcal{L}_A \to \mathcal{L}_C)}\right) \tag{10}$$

exists only if

$$\exists w_{\mathcal{L}_A}, w_{\mathcal{L}_B}, w_{\mathcal{L}_C} : w_{\mathcal{L}_B} \in \mathcal{T}_{D(\mathcal{L}_A \to \mathcal{L}_B)}(w_{\mathcal{L}_A}) \wedge w_{\mathcal{L}_C} \in \mathcal{T}_{D(\mathcal{L}_B \to \mathcal{L}_C)}(w_{\mathcal{L}_B}) \tag{11}$$

and the translation probability of $w_{\mathcal{L}_C}$ being a translation of $w_{\mathcal{L}_A}$ is computed as:

$$\sum_{w_{\mathcal{L}_B}} \mathcal{P}\left(w_{\mathcal{L}_C} \in \mathcal{T}_{D(\mathcal{L}_A \to \mathcal{L}_B)}(w_{\mathcal{L}_B})\right) \times \mathcal{P}\left(w_{\mathcal{L}_B} \in \mathcal{T}_{D(\mathcal{L}_A \to \mathcal{L}_B)}(w_{\mathcal{L}_A})\right) \tag{12}$$

for all $w_{\mathcal{L}_B}$ that is a translation of $w_{\mathcal{L}_A}$, and translated by $w_{\mathcal{L}_C}$.

The composition of two dictionaries, taking into account what occurrence count, and what translation probability to use, is defined textually as:

- the occurrence count for each entry is maintained as the occurrence count for the word in the original first dictionary;
- the probability of a translation is computed summing up the products from translating from the original language to the pivot language, and from the pivot language to the target language.

This operation can be used for different tasks:

- compute a new dictionary for a language pair that does not exist as parallel corpora, or which parallel corpora exists but in small quantities [17]. One problem of this approach is that the obtained probabilities are smaller than the ones we would obtain with a direct extraction. Nevertheless, the use of the totalization operation can help mitigate this issue.

– compute similar or parent words, or even, some kind of probabilistic *synsets*, composing a dictionary with its inverse. The result is a dictionary from a language to itself. In this dictionary, each entry maps a word to a set of *probable translations* or *probable synonyms*. To illustrate this mechanism, consider the following two entries obtained when composing the dictionaries extracted from EuroParl corpus:

$$\text{house} \begin{cases} \text{house} & 49\% \\ \text{parliament} & 29\% \\ \text{assembly} & 7\% \\ \text{chamber} & 6\% \\ \text{plenary} & 2\% \\ \text{home} & 2\% \end{cases} \quad \text{evil} \begin{cases} \text{badly} & 17\% \\ \text{wrong} & 14\% \\ \text{evil} & 12\% \\ \text{bad} & 6\% \\ \text{poorly} & 5\% \\ \text{scourge} & 3\% \end{cases}$$

A similar approach to build weighted *synsets* using PTD, has also been used in the context of software reverse engineering to build mappings between program identifiers (terms) and real word concepts [1].

3.6 Dictionary Filtering and Mapping

Depending on the source corpus, the dictionaries may contain non-words, like punctuation, numbers, e-mails, URLs or any other kind of junk. Although this might be achieved using domain restriction, a robust filtering mechanism is relevant.

In fact, it might be useful to filter dictionaries in other ways. For instance, removing all entries with a low number of occurrences, or all translations with low probability. To allow different kind of filtering approaches the filtering function has been defined as an high-order function, receiving as argument a predicate that evaluates if an entry should be preserved.

For example, creating a PTD composed only of verbs (PTD^V) can be defined using the filter function as:

$$PTD^V = filter(PTD, verb)$$

where, $verb$ is a function defined as:

$$verb(entry) = \begin{cases} True & \text{if } word \text{ in } entry \text{ is a verb} \\ False & \text{otherwise} \end{cases}$$

Considering the predicate as a function, which instead of returning a boolean value, returns a dictionary entry (modified or unmodified) or a null value, this filter mechanism can work also as a function mapper (typically known as map function).

This mapping approach is even more flexible if we can produce lateral effects. In this case this operation can work as an iterator, to process each entry, and save its result some where else.

This set of filters is quite useful and versatile. In fact, most of the functions defined in this article are implemented as filters.

3.7 Dictionary Algebra Summary

Table 3 presents a summary of the defined algebra, together with the operation signature, and a reference to the section that discusses it[14].

Table 3. Summary of Dictionary Algebra operations and their signatures.

Function	Syntax	Domain	Range	Section
domain	$\mathrm{dom}\,(\_)$	$\mathrm{PTD}_{A\to B}$	$\longrightarrow \mathrm{W}_A{}^*$	
range	$\mathrm{ran}\,(\_)$	$\mathrm{PTD}_{A\to B}$	$\longrightarrow \mathrm{W}_B{}^*$	
union	$\_ \cup \_$	$\mathrm{PTD}_{A\to B} \times \mathrm{PTD}_{A\to B}$	$\longrightarrow \mathrm{PTD}_{A\to B}$	3.1
intersection	$\_ \cap \_$	$\mathrm{PTD}_{A\to B} \times \mathrm{PTD}_{A\to B}$	$\longrightarrow \mathrm{PTD}_{A\to B}$	3.2
composition	$\_ \circ \_$	$\mathrm{PTD}_{A\to B} \times \mathrm{PTD}_{B\to C}$	$\longrightarrow \mathrm{PTD}_{A\to C}$	3.5
domain restrict	$\_ / \_$	$\mathrm{PTD}_{A\to B} \times \mathrm{W}_A{}^*$	$\longrightarrow \mathrm{PTD}_{A\to B}$	3.3
domain subtract	$\_ \setminus \_$	$\mathrm{PTD}_{A\to B} \times \mathrm{W}_A{}^*$	$\longrightarrow \mathrm{PTD}_{A\to B}$	3.3
totalization	$\mathrm{tot}\,(\_)$	$\mathrm{PTD}_{A\to B}$	$\longrightarrow \mathrm{PTD}_{A\to B}$	3.4
filter	$\mathrm{filter}\,(\_, \_)$	$\mathrm{PTD}_{A\to B} \times (\mathrm{ENTRY}_{A\to B} \longrightarrow Bool)$	$\longrightarrow \mathrm{PTD}_{A\to B}$	3.6
map	$\mathrm{map}\,(\_, \_)$	$\mathrm{PTD}_{A\to B} \times (\mathrm{ENTRY}_{A\to B} \longrightarrow \mathrm{ENTRY}_{A\to B})$	$\rightarrow \mathrm{PTD}_{A\to B}$	3.6

4 Conclusions

Probability Translation Dictionaries are being used widely, although not always identified as such. The lack of a standard name and a formal definition makes resources hard to share, and discussion on their usage mostly impossible.

While there is not a standard way of defining them, or a standard way to compute them, all the approaches share a common trunk: words are mapped to probable translations, together with a probability measure.

In this article we defined formally the base structure of a PTD (that can be extended easily with further information) as well as a XML schema to make the sharing process easier.

The definition of a PTD algebra clarifies a set of standard operations that can be performed with these objects, and gives some insights on what can be obtained using these operations, with some usage examples.

The algebra was not just defined, but is being already used as a Perl module to manage and operate with PTD. This module also includes a set of programs that let the user apply these operations directly to dictionaries without the need of writing any code. It includes other tools, like the computation of statistics of a dictionary.

Acknowledgements

This work is partially supported by Per-Fide.

The Per-Fide project is supported in part by a grant (Reference No. PTDC/CLEL-LI/108948/2008) from the Portuguese Foundation for Science and Technology and it is co-funded by the European Regional Development Fund.

[14] For compactness this table abbreviates $\mathcal{L}_A$ as just A. So, $w_{\mathcal{L}_A}$ is represented as W_A, etc.

We would like to thank all contributing authors, translators, publishers and institutions for their generosity in allowing us to include their texts in the *Per-Fide* Corpus.

References

1. Nuno Ramos Carvalho, José João Almeida, Maria João Varanda Pereira, and Pedro Rangel Henriques. Probabilistic synset based concept location. In Alberto Simões, Ricardo Queirós, and Daniela da Cruz, editors, *SLATE'12 — Symposium on Languages, Applications and Technologies*, volume 21, pages 239–253. OASIC – Open Access Series in Informatics, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany, June 2012.
2. Helena M. Caseli, Maria G. V. Nunes, and Mikel L. Forcada. Evaluating the LIHLA lexical aligner on Spanish, Brazilian Portuguese and Basque parallel texts. *Procesamiento del Lenguaje Natural*, September 2005.
3. Pascale Fung and Kenneth Church. Kvec: A new approach for aligning parallel texts, 1994.
4. Pablo Gamallo Otero and José Ramom Pichel Campos. Automatic generation of bilingual dictionaries using intermediary languages and comparable corpora. In *Proceedings of the 11th international conference on Computational Linguistics and Intelligent Text Processing*, CICLing'10, pages 473–483, Berlin, Heidelberg, 2010. Springer-Verlag.
5. Xavier Gomez Guinovart and Alberto Simões. Terminology extraction from English-Portuguese and English-Galician parallel corpora based on probabilistic translation dictionaries and bilingual syntactic patterns. In António Teixeira, Miguel Sales Dias, and Daniela Braga, editors, *I Iberian SLTech 2009*, pages 13–16, Porto Salvo, Portugal, September, 3–4 2009.
6. Xavier Gómez Guinovart and Elena Sacau Fontenla. Métodos de optimización de la extracción de léxico bilingüe a partir de corpus paralelos. *Procesamiento del Lenguaje Natural*, 33:133–140, 2004.
7. Djoerd Hiemstra. Using statistical methods to create a bilingual dictionary. Master's thesis, Department of Computer Science, University of Twente, August 1996.
8. Philipp Koehn. EuroParl: A parallel corpus for statistical machine translation. In *Proceedings of MT-Summit*, pages 79–86, 2005.
9. W. Kraaij. TNO at CLEF-2001: Comparing Translation Resources. *Lecture Notes in Computer Science*, pages 78–93, 2002.
10. W. Kraaij. Exploring transitive translation methods. In *Proceedings of DIR*, 2003.
11. Solomon Kullback and Richard A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.
12. I. Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.
13. Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.
14. Alberto Simões and J. João Almeida. NatServer: a client-server architecture for building parallel corpora applications. *Procesamiento del Lenguaje Natural*, 37:91–97, September 2006.
15. Alberto Simões and José João Almeida. Bilingual terminology extraction based on translation patterns. *Procesamiento del Lenguaje Natural*, 41:281–288, September 2008.
16. Alberto Simões and José João Almeida. Bilingual example segmentation based on Markers Hypothesis. In António Teixeira, Miguel Sales Dias, and Daniela Braga, editors, *I Iberian SLTech 2009*, pages 95–98, Porto Salvo, Portugal, September, 3–4 2009.

17. Alberto Simões and Xavier Gómez Guinovart. Translation dictionaries triangulation. In Carmen Mateo, Francisco Díaz, and Francisco Pazó, editors, *FALA2010 – II Iberian SLTech Workshop*, pages 171–174, Vigo, November 2010.
18. Alberto M. Simões and J. João Almeida. NATools – a statistical word aligner workbench. *Procesamiento del Lenguaje Natural*, 31:217–224, September 2003.
19. Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaž Erjavec, Dan Tufiş, and Dániel Varga. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *5th International Conference on Language Resources and Evaluation (LREC'2006)*, Genoa, Italy, 24–26 May 2006.
20. Nitin Varma. Identifying word translations in parallel corpora using measures of association. Master's thesis, University of Minnesota, 2002.