Report TER
Master 1 Informatique – IAAA

TIAD – Translation inference across dictionaries

THAI Ba Tuan

Supervisor
Carlos Ramisch

Key words
Translation inference, Word Embeddings, cross-lingual word embeddings

Marseille, May 2021

Table of Contents

List of Figures

List of Tables

1. Introduction

TIAD – Translation Inference Across Dictionaries shared task held by Jorge Gracia, University of Zaragoza, Spain from 2018. It aims to explore methods and techniques to automatically generate new bilingual (and multilingual) dictionaries from existing dictionaries. The initiative also aims to promote further research on the topic of inferring translations between languages.

The objective of TIAD-2021 shared task is to generate automatically bilingual dictionaries between English (EN), French (FR) and Portuguese (PT) based on 53 known translations contained in the Apertium RDF graph (because of these three languages are not directly link in the graph). The result of the inference dictionary will be built in the format TSV which contains "source word, target word, part of speech (pos), confidence score" and submit to the organisers on 14/05/2021. The results will be published at 14/06/2021 on the main page of TIAD-2021.

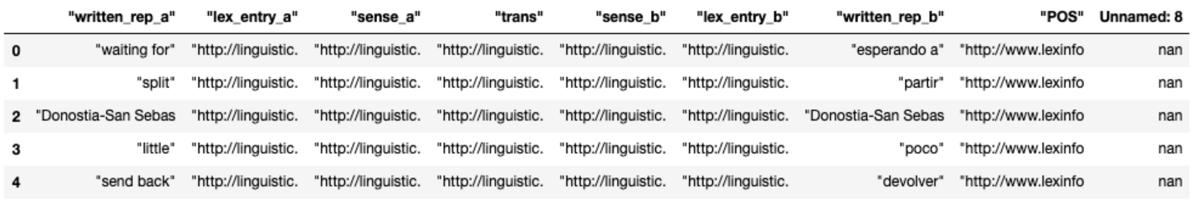
2. Description of the systems

2.1 Data processing

There are two ways to obtain the data from Apertium RDF graph, first is use SPARQL to access to the database, second is use the CSV "shortcut" given by the organisers. For saving time, I choose the second ways to get the data.

The CSV "shortcut" dictionary in the TIAD-2021 shared task has 51 csv files (missing 2 files due to the technical problem of the organisers), one per translation set and containing of the following information: "source word (String)", "source lexical entry (URI)", "source sense (URI)", "translation (URI)", "target sense (URI)", "target lexical entry (URI)", "target written representation", "part of speech (URI)". Entry, part of speech (POS), sense use the Ontolex lemon core model to represent the data.

Example: The dictionary EN-ES



	"written_rep_a"	"lex_entry_a"	"sense_a"	"trans"	"sense_b"	"lex_entry_b"	"written_rep_b"	"POS"	Unnamed: 8
0	"waiting for"	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"esperando a"	"http://www.lexinfo	nan
1	"split"	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"partir"	"http://www.lexinfo	nan
2	"Donostia-San Sebas	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"Donostia-San Sebas	"http://www.lexinfo	nan
3	"little"	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"poco"	"http://www.lexinfo	nan
4	"send back"	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"http://linguistic.	"devolver"	"http://www.lexinfo	nan

Figure 1 Dictionary of English to Spanish

In this project, I don't pay attention in the "sense" of word, so I remove this column and just keep the information about: "source word", "target word", "part of speech", processing and save in another csv files. The POS can has one the values: "properNoun, noun, adjective, verb, adverb, preposition, determiner, numeral, …", the distribution of all POS of each language represent in the table below:

Table 1 List all POS size in data

	all	EN	PT	FR	CA	ES
noun	307599	26707	11475	27798	33050	35829
properNoun	166166	43856	24210	36626	42573	34225
adjective	107067	9990	9670	11519	17100	19939
verb	70248	6640	2519	6208	6574	9015
adverb	44566	4756	2556	3670	8010	9556
preposition	3819	263	133	394	419	464
numeral	2300	265	162	220	229	236
determiner	1487	216	30	67	100	154

Indeed, the mission of the shared task is to build the dictionary from “EN”, “FR”, “PT” from the inference dictionary, so I care about the path from “source word” to the “target word”. For example: to build the dictionary between the English and French, the dictionary of the pair: “EN-ES, ES-FR” or “EN-ES, ES-OC, OC-FR” are consider. For looking in the graph, I can recognize not all of pair dictionary is needing to build the inference dictionary (ex: EN-CY, EN-KK, …) so, by using the algorithm for path finding (Depth First Search) in the all-pair Graph, I can find all path form EN, FR, PT to each other and using this dictionary for experimenting the algorithms. Example a long path find from EN to FR is: [(‘EN’, ‘EO’), (‘EO’, ‘CA’), (‘CA’, ‘SC’), (‘SC’, ‘IT’), (‘IT’, ‘ES’), (‘ES’, ‘OC’), (‘OC’, ‘FR’)]. There are 188, 146, 241 paths form EN-FR, EN-PT, PT-FR respectively. I can limited the length of the path finding but it means that the coverage of the word for each language will reduce, so I keep all path, and extract all node which contains in all path, they are: [‘EN’, ‘PT’, ‘IT’, ‘OC’, ‘FR’, ‘SC’, ‘GL’, ‘AN’, ‘CA’, ‘EU’, ‘EO’, ‘ES’, ‘RO’]. By getting all file with contain these node, I got 27 csv files for building the inference dictionary.

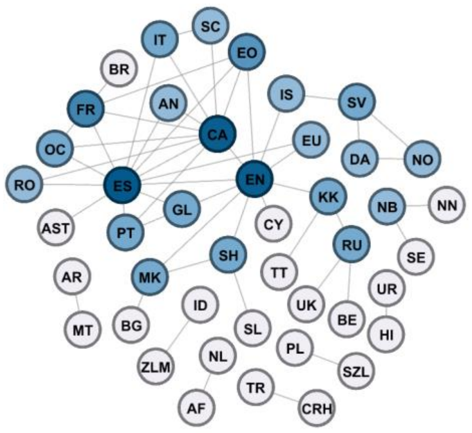


Figure 2 The graph of all dictionaries in Apertium RDF v2 source: TIAD-2021

A table below will show the details of all word of each language and its POS need to find the translate.

Table 3 Size of word need to translate

	size
EN	59576
PT	82200
FR	51035

Table 2 POS of each lang need to inference

	EN	PT	FR
properNoun	20978	24210	36626
noun	19945	11475	27798
adjective	7996	9670	11519
verb	5299	2519	6208
adverb	4266	2556	3670
preposition	212	133	394
determiner	193	30	67
numeral	168	162	220

All of this work is done by using the Pandas library for processing the CSV files and NetworkX library for path finding.

2.2 Algorithms

In this section, I will show the details about the algorithms for building the dictionary. There are 2 approaches in my project:

2.2.1 Word Embedding approach

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation [1], this method using the real-values vector to represent the word in the context from a corpus. Word2Vec, GloVe, fasttext are the word embedding method most using nowadays.

a) Word2Vec

About Word2Vec:

Introduced in 2013 by Tomas Mikolov, et al. [2] and according to [1] Word2Vec is a statistical method for efficiently learning a standalone word embedding from a text corpus. Word2Vec technique for learning is based on a feed-forward, fully connected neural network. It is using two different learning model to learn the word embedding:

- Continuous Bag-of-Words - CBOW model: learn by predicting the current word based on its context
- Continuous Skip-Gram Model: learn by predicting the surrounding context word of the current word.

The context of word is defined by a window size of neighboring words, this window size is a configurable parameter of the model. The embeddings of a target word is extracted by calculate the probability of predicting context words given by its.

Implementation the Word2Vec algorithms:

For each translation word in each dictionary, I will create a pseudo-sentence, which have the form below:

X_lang1_pos Y_lang2_pos X_lang1_pos Y_lang2_pos

Where "X Y pos" is the translate from language "lang1" to "lang2"
For example:
With the dictionary EN-ES and the translation "life vida noun", I will reform this translation by:

life_EN_noun vida_ES_noun life_EN_noun vida_ES_noun

Building corpus:

By applying the transform above to all translation in the 27-bilingual dictionary (CSV files), and then concatenate all together, I will have a corpus text for learning the word embeddings.

Why this approach works?

It is clearly that when I build the sentence above, if I am using the window size is 1, so the word "life_EN_noun" completely based on the word "vida_ES_noun" in the context and vice versa. And if I have the other sentence, example is the dictionary FR-ES:

vie_FR_noun vida_ES_noun vie_FR_noun vida_ES_noun.

It may indicate that the word "life" in English is translate to French is "vie" with part of speech "noun"

How to build the inference dictionary:

Note: Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space. With two vector A and B, the cosine between two vectors is calculated by:

$$\cos\theta = \frac{A*B}{||A||*||B||}$$

The $\cos(\theta)$ is close to 1, the vector A and B is related, 0 mean A and B is unrelated, close to -1 mean A and B is opposite.

After using the word2vec embedding to learn the vector representation of each word in the corpus, I can build the inference dictionary by using the cosine similarity. The algorithm below will create the dictionary of the given source and target lang:

Algorithme 1: Building Inference dictionary Word2Vec
Input: source lang, target lang, word embedding model
Output: dictionary of two languages

```
1. Pick all the word of source and target language in Word2Vec model by get all the
   word which have "_source-lang_" or "_target-lang". They are W1 and W2
2. For each word in W1:
    1. Calculate score cosine similarity of this word to all word in W2
    2. Order score descending
    3. Get the first word with the same pos with entry word
    4. Make the output: "source word \t target word \t pos \t score"
       Where score is the confidence of this translate
3. Return Dictionary
```

With the input is EN-FR, EN-PT, FR-EN, FR-PT, PT-EN, PT-FR, I can make 6 dictionaries.

b) Glove and fasttext

Glove and fastText is the newer word embedding method. In this project, after testing the embeddings with Glove and fastText, I saw it was not worked if using the corpus create like the Word2Vec, so I decided not using those method.

2.2.2 Cross-lingual word embedding (CLWE) approach

What is CLWE?

Cross-lingual word embeddings simply refers to word embeddings in two or more languages that are aligned to a common space, so that words that translation pairs of words between languages are similar. There are various approaches for learning CLWE: Monolingual mapping, Pseudo-cross-lingual, Cross-lingual training, Joint optimization. In this project, I used monolingual mapping to learn the mapping between each pair of language.

MUSE

Monolingual mapping: These models initially train monolingual word embeddings on large monolingual corpora. They then learn a linear mapping between monolingual representations in different languages to enable them to map unknown words from the source language to the target language. Suppose that we have a dictionary with n pairs of words from $\{x_i, y_i\}_{i=1..n}$ we will learn a linear mapping W between the source and target by optimize the equation:

$$W^* = argmin_{W\in dxd}||WX - Y||$$

Which d is the dimension of the embeddings and X, Y are the aligned matrices of size d × n containing the embeddings of the words in the dictionary. MUSE: Multilingual Unsupervised and Supervised Embeddings is the state-of-the multilingual word embeddings, it includes two methods:

- Supervised – that use the bilingual dictionary for align the matrix W. MUSE using the Procrustes for finding the matrix W
- Unsupervised – using adversarial network for learning the mapping matrix.

In this project, I used the supervised method, and the align bilingual dictionary is the dictionary of the TIAD shared task.

Why MUSE

Muse is using fastText embedding which have the advantages than other word embedding that it can handle the out of vocabularies problem. For example: the are many properNoun which may be harly to appear in the popular corpus like: "Campus per la Pau" and the subordinatingConjunction like "*let's put the record straight*", "*but the early worm was eaten*". By using the fastText embedding, it can be calculate the vector based on the "sub word" of each word.

Implementation MUSE.

To create the dictionary form "source-lang" to "target-lang" we will create the mapping matrix from the word embeddings of lang1 to lang2. Indeed, the are no bilingual dictionary form "source-lang" to "target-lang", so it can't be directly creating the mapping matrix for two languages. For doing this, I chosen the language pivot, from that mapping the source-lang to the pivot-lang, and from pivot-lang to target-lang. It shows in the figure 2.

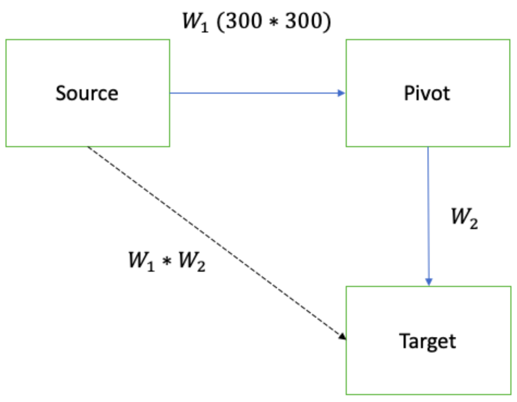


Figure 3 Getting the mapping matrix from Source to Target

Why it works?

Suppose that the matrix W_1 and W_2 is mapping matrix from source-pivot and pivot-target respectively. Then, with a vector x, representation for the word in the source language, we can find the vector y in the pivot-lang that close to x, and we can find the vector z in the
$x \to x\, W_1 \sim y$ **in the pivot-lang**
and $y \to y\, W_2 \sim z$ in the target-lang
so with a source-lang word the $x \to x\, W_1 W_2 \sim yW_2 \sim z$ in the target-lang
That lead to we can assume that $W_1 W_2$ is the mapping matrix form source-lang to target-lang.

Building the dictionary

```
Algorithm 2: Building Inference dictionary from MUSE
Input: source-words, target-words, source embeddings, target embeddings, mapping matrix
source-pivot W1, mapping matrix pivot-target W2
Output: dictionary of two languages
    1. Calculate the mapping matrix from source to target $W_1W_2$
    2. For each word $x$ in source-words:
        1. Get the vector representation of x is v
        2. Calculate $vW_1W_2$ is v1
        3. Calculate cosine similarity score of v1 to all vector in target embeddings
        4. Sort the score and get the word with the same pos with input
        5. Make the output: source word \t target word \t pos \t score
           Where score is the confidence of this translate
    3. Return Dictionary
```

***Comment*:** if the true translated of the source word no appear in the target embedding, this word is translated with very low confidence (maybe negative). If the confidence is negative, I will change it to 0 (not remove to keep better the coverage)

2.2.3 Systems workflow

In fact, there are a lot of properNoun in the dataset, and if the source lang and the target lang has the same word and the same pos, maybe it is the translate from source to target, so, by the filter all word the same in source and target (Common Set Dictionary), we will have lots of words no need to find the translate, example:

- For the EN and FR dictionary: “intelligible”, “international” have the “adjective” POS or “Brook”, “Robineau” have the same “properNoun” POS.
- For the FR-PT dictionary: “Santa Coloma de Gramenet”, “Sarroca de Lleida”: properNoun, or “de sorte que” have the same “subordinatingConjunction” POS,

I will list all the same word and add into one TSV file for each pairs of languages, set the confidence score is “1.0” this is “Common set”. The remain words which need to find the translate, will be in input of Word2Vec or MUSE for finding the translate. After that, the result will merge with the “Common Set” to build the complete Inference Dictionary.

Table 5 Number of the same word-pos for each pair of Langague

Pair	size
EN-FR	20738
FR-PT	13373
PT-EN	23219

Table 4 Number of words need to find the translate for each pair

	size
EN-FR	40131
EN-PT	46857
FR-PT	58996
FR-EN	62754
PT-EN	38314
PT-FR	27830

Note: The sum of EN-FR same + EN-FR difference is greater than EN-words below because of: the same word of each pairs is extract from whole dataset (51 dictionaries) not only of 27 dictionaries. Same as the another paris.

So, the workflow for building Inference dictionary is:

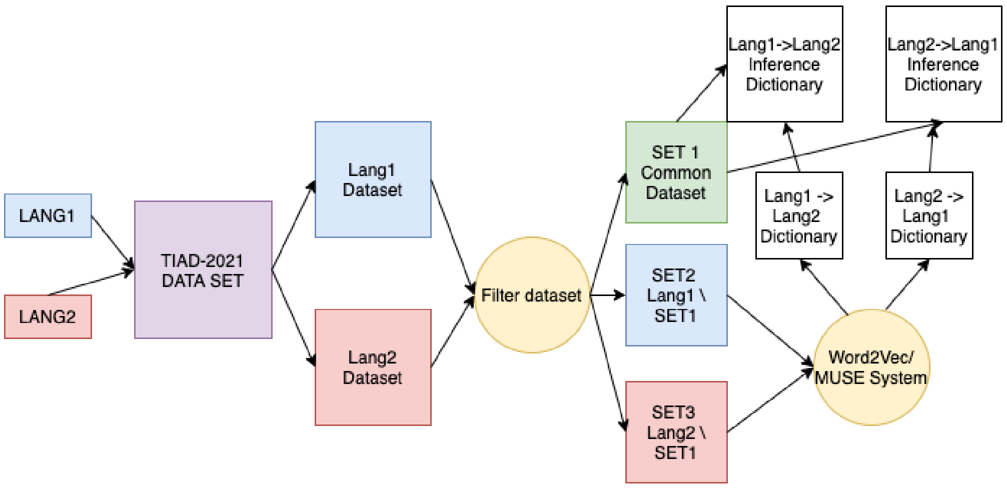


Figure 4 Details of System to build the Inference Dictionary

3. Experiments and Results

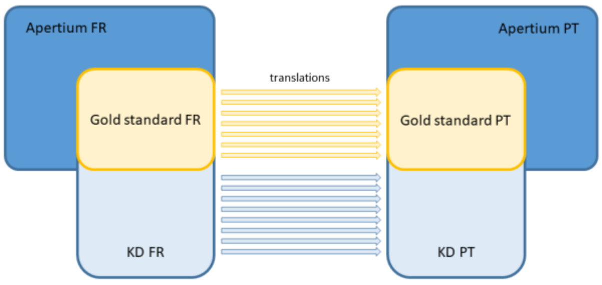
In this part, I will talk about how to evaluate the result and the details of experiment for the Word2Vec and MUSE

3.1 Evaluation Data

3.1.1 Data Evaluation of shared task

Gold Standard is extracted translations by the organisers from manually compiled pairs of K Dictionaries (KD), particularly its Global series. The coverage of KD is not the same as Apertium. To allow comparisons, they took the subset of KD that is covered by Apertium to build the gold standard, i.e., those KD translations for which the source and target terms are present in both Apertium RDF source and target lexicons. The gold standard remained hidden to participants.

The evaluation of the results was carried out by the organizers against manually compiled language pairs of K Dictionaries, extracted from its Global series, particularly the following pairs: PT-EN, EN-PT, FR-EN, EN-FR, FR-PT, PT- FR.
Because the "gold standard" hidden to participants, I will build my own dictionary for the testing and using another metrics for calculate the performance of the results.



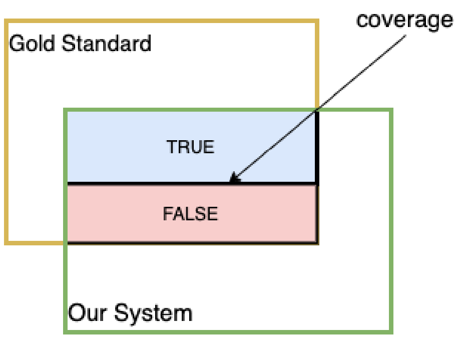


Figure 6 The Gold standard Dictionary (source: TIAD) *Figure 5 Coverage and the result*

With the builder dictionary, the operator will calculate the performance of the dictionary by F-measure score. The coverage is the intersection between the "Gold Standard" set and the "our system" dictionary set, and then the F-measure score will be calculated by:

$$Precision(P) = \frac{|TRUE|}{|TRUE|+|FALSE|}\ Recall(R) = \frac{|TRUE|}{|Gold\ Standard|}$$

$$F - measure\ = 2/(\frac{1}{R}+\frac{1}{P}) = \frac{2*R*P}{R+P}$$

- Precision - show the performance in our System
- Recall calculates the performance in the gold standard (include in coverage)
- F-measure is the geometric mean, if P or R is low, that lead to the F-measure low. The perfect system is where P = R = F-measure = 1.

3.1.2 Building my own dataset

Because of the dataset for valuation of the author isn't public and using the API of KDictionaries for get all data to evaluate, is cost $20 for 20,000 monthly calls + $0.0005 per extra call. Therefore, It is impossible to evaluate the true performance of the inference dictionary. So, I were build my own dictionary for the valuation of the performance of the dictionary built. For this task, I have two options to build the valuation dictionary: First using Wikipedia dictionary, and second using Google translate.

a. Wiktionary

According to the definition in the Wikipedia: Wiktionary is a multilingual, web-based project to create a free content dictionary of terms (including words, phrases, proverbs, linguistic reconstructions, etc.) in all-natural languages and in several artificial languages. These entries may contain definitions, images for illustrations, pronunciations, etymologies, inflections, usage examples, quotations, related terms, and translations of words into other languages, among other features. It is collaboratively edited via a wiki. Its name is a portmanteau of the words wiki and dictionary. It is available in 171 languages and in Simple English [3].

In this TER I will try to craw the data from Wikionray of the 6-language pair: EN-FR, EN-PT... and using it to evaluate this quality of the building dictionary. It can be done by download the Wikidump file (link) and then extract the dictionary, it is quite of difficult. I chose the other way, using the direct dictionary which built by Matthias Buchmeier. First I will get the link of the dictionary and then Parser the html page using library BeautifulSoup for extract the word, translation, sense of this word and the part of speech. I was completed building 6 dictionaries by this way, the details in the source code. The size of all dictionaries is listed in the table below:

Table 6 Size of the evaluation dictionary

Lang, Pair	Size (Wiki)	Size (Google Translate)
EN-FR	118272	159104
EN-PT	102269	103597
FR-EN	125945	68584
FR-PT	29340	74851
PT-EN	91813	120399
PT-FR	12235	49517

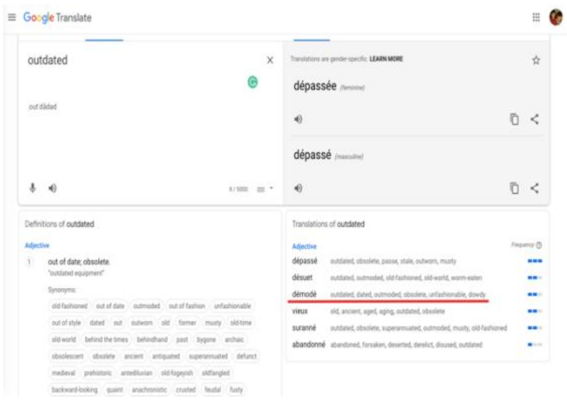
b. Google Translate

Given the fact that there are many properNoun in the RDF dictionary dataset, so the Wiktionary dictionary will give the bad result of evaluation in this case (because out of vocabulary) so I think I can using the google translate to build another dictionary for increase the performance of the evaluation test. The google translated have an advantage that can give

the POS and have a lot of word similaries. For example: the English word: "outdated" when translate to France have many word: "dépassé", "désuet", "démodé", "veux" which have more translate than the dictionary of Wiktionary. And if google translate give the translate of target word equal to the source word, it may be a properNoun part of speech.
We can easily get all the translate by using Selenium – a web Scapping library. We can create dictionary by the pseudocode:
Example: Building the dictionary English - French

```
a. Input English word
b. Using selenium to send text to the text area of source (using XPath of <textarea>)
c. Wait until the page response (time out 3s)
d. Get the translate:
    i. If have no POS, we will set the POS is properNoun
    ii. Else get the POS and all translate in the "Translations of" tag (using XPath)
e. Return to step 1
```

For each word, it takes 1 second to get the translate, so for ~59000 English words it take 16 hours to get all translated. To reduce the time running, I split in to 20 files, each file has 3000 words and running in 4 google colab tab and 2 remote computers of VDI-AMU.



Adjective		Frequency
dépassé	outdated, obsolete, passe, stale, outworn, musty	▪▪▪
désuet	outdated, outmoded, old-fashioned, old-world, worm-eaten	▪▪
démodé	outdated, dated, outmoded, obsolete, unfashionable, dowdy	▪▪
vieux	old, ancient, aged, aging, outdated, obsolete	▪▪
suranné	outdated, obsolete, superannuated, outmoded, musty, old-fashioned	▪▪
abandonné	abandoned, forsaken, deserted, derelict, disused, outdated	▪

Figure 7 Translate of word "outdated"

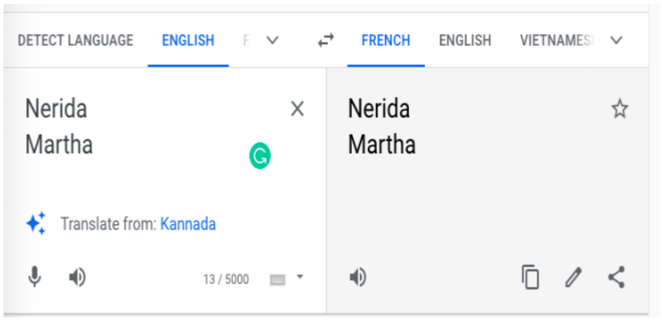


Figure 8 translate of 2 properNoun word

For evaluating the performance of the dictionary, I will add another metric: **accuracy**, mean that:
For each translation pair, if this pair with its POS are in the evaluation dictionary, this is True translation, and the accuracy is calculated by:

$$acc = \frac{|True\ Translate|}{|Building\ Dictionary|}$$

When I have the evaluation dictionary, ready for the experiment of the algorithm.

3.2 Experiment setup

3.2.1 Word Embedding

a. Traning

Building the corpus of pseudo-sentence from 27 bilingual dictionaries (section 2.2.1), I had 66.6 MB text and 810598 lines. For Training Word2Vec model, I used the Gensim implementation. Training on Google Colab with two version: Skip-Gram and Cbow, 200 epochs, vector size 300, window size = 1, min count = 2 (to get all vocabularies). It took about

30 minutes for completing the training. After that is building the corpus like the algorithm given

in the section 2.

b. Result

Table 7 The result of Building dictionary for Word embedding with Cbow

	Wikipedia			Google Translate			Accuracy	
	Precision	Recall	F-measure	Precision	Recall	F-measure	Wiktionary	Google translate
EN-FR	0.7606	0.3304	0.4607	0.5185	0.1898	0.2779	0.3325	0.4966
EN-PT	0.7604	0.3153	0.4458	0.3372	0.1112	0.1673	0.3077	0.1915
FR-EN	0.4502	0.2526	0.3236	0.5001	0.2009	0.2866	0.1877	0.165
FR-PT	0.5127	0.344	0.4118	0.2069	0.2007	0.2037	0.0888	0.1828
PT-EN	0.6005	0.3172	0.4152	0.5532	0.2372	0.3321	0.1995	0.5527
PT-FR	0.7519	0.666	0.7064	0.4146	0.4274	0.4209	0.1024	0.4146

Table 8 The result of Building dictionary for Word embedding with Skip-gram

	Wikipedia			Google Translate			Accuracy	
	Precision	Recall	F-measure	Precision	Recall	F-measure	Wiktionary	Google translate
EN-FR	0.7592	0.3298	0.4598	0.5168	0.1892	0.2770	0.3318	0.4951
EN-PT	0.7615	0.3157	0.4464	0.3361	0.1109	0.1667	0.3081	0.1909
FR-EN	0.4474	0.251	0.3216	0.4936	0.1983	0.2829	0.1865	0.1629
FR-PT	0.5118	0.3434	0.411	0.2069	0.2008	0.2038	0.0886	0.1828
PT-EN	0.5932	0.3134	0.4101	0.5482	0.2351	0.3291	0.1971	0.5477
PT-FR	0.7481	0.6625	0.7027	0.4137	0.4264	0.42	0.1018	0.4136

Comment:

- The accuracy when using the google translate is better than Wiktionary because "google translate" have more words
- Using Cbow model quite better than skip-gram
- Some correct translate of EN-FR:

apple	pomme	noun	0.99
approximate	approximatif	adjective	0.95
arm	bras	noun	0.93

- Some wrong translate of EN-FR:

in practice	effrénément	adverb	0.56
in reply	à bout de souffle	adverb	0.5
20th Century Fox	Gemeaux	properNoun	0.51

- A translate that correct with google translate but not with Wiktionary:

  isometric isométrique adjective

- A translate that correct by google translate and Wiktionary have detected wrong:

  counterattack contre-attaquer verb 0.67

  google translate:

  counterattack#contre-attaque#None
  counterattack#la contre-attaque#noun
  counterattack#riposter#verb

So, sometimes a correct translation is evaluated as incorrect because the wiktionary/google-translate do not contain that pair.

Note that the accuracy is just for reference.

3.2.2 Cross-lingual approach

a. Build the embedding with fastText

For training the cross multilingual dictionary, the first step is building the word embedding, in MUSE it require for using fasttext embedding [4]. So, we must have the corpora which have the sentence, in this the word have the POS. The Universal Dependencies (UD) framework have the annotation of grammar (parts of speech, morphological features, …) across different human languages. It contains over 100 languages and nearly 200 treebanks, but in fact the corpus of UD is very small example: Spanish have 34 693 sentences, it is difficult to embedding with this dataset. Then, in this project, I used another copora with have the sentence annotated, this is CoNLL 2017 Shared Task - Automatically Annotated Raw Texts and Word Embeddings data, with have the raw text which automatically annotated, and it is big corpus.

After processing the data by concating the word and its pos (example with EN word "data": data_noun), I got over 5 GB text for each language (EN, FR, PT, ES, CA) and ready to build the word embedding with fastText

Computer : Intel core i7-1050H, 16 GB RAM
Library : fastText skipgram
Configurations: vector size 300, min count 4, epochs 15, window size 4, loss hs (hierarchical softmax for better speed - link)

It takes about ~130 minutes for completing training. All the corpus, and model embedding is on the link in GitHub repo.

b. Build the mapping matrix

By Using the data of TIAD-2021, I can create the align dictionary for:
"EN-ES, FR-ES, PT-ES, EN-CA, FR-CA, PT-CA, ES-EN, ES-FR, ES-PT, CA-EN, CA-FR, CA-PT" and use those for running on MUSE algorithm

Library : MUSE Supervised
Configuration: n_refinement 15, cuda False
I take 40 minutes for completing build the mapping matrix for each pair of language.

Build the Inference Dictionary

Running the algorithm in the section 2, to build the dictionary, It take at least 300 minutes for building a dictionary. Result:

Table 9 Result on using ES as pivot

	Wikipedia			Google Translate			Accuracy	
	Precision	Recall	F-measure	Precision	Recall	F-measure	Wiktionary	Google translate
EN-FR	0.8714	0.3143	0.4619	0.6631	0.1314	0.2194	0.3772	0.6531
EN-PT	0.1316	0.0424	0.0642	0.1609	0.0234	0.0409	0.0563	0.0958
FR-EN	0.1558	0.0801	0.1058	0.1892	0.035	0.0591	0.0698	0.0552
FR-PT	0.0632	0.0363	0.0461	0.1879	0.0842	0.1163	0.0095	0.1709
PT-EN	0.1338	0.0683	0.0904	0.4148	0.0841	0.1398	0.0387	0.4141
PT-FR	0.0407	0.0356	0.038	0.2569	0.2314	0.2435	0.006	0.2569

Table 10 Result on using CA as pivot

	Wikipedia			Google Translate			Accuracy	
	Precision	Recall	F-measure	Precision	Recall	F-measure	Wiktionary	Google translate
EN-FR	0.3938	0.1379	0.2043	0.5154	0.0949	0.1603	0.1592	0.507
EN-PT	0.5987	0.2344	0.3369	0.2977	0.0679	0.1106	0.2793	0.169
FR-EN	0.2182	0.1161	0.1516	0.2597	0.035	0.0617	0.0853	0.0552
FR-PT	0.067	0.0384	0.0488	0.1919	0.0842	0.1171	0.0097	0.0673
PT-EN	no file	no file	nofile	no file	no file	nofile	nofile	nofile
PT-FR	0.0402	0.0352	0.0376	0.2554	0.2314	0.2428	0.0059	0.2553

Comment: The result on using MUSE method is good, due to the reasons:

- MUSE don't handle the word upper case – because of the source code provide by Facebook do not handle this case (line 58 in link), and the time is limited in this project for modifying the code.
- Because of using MUSE to build the dictionary has a lot of time, so some dictionary I did not finished to translate all word to submit to the contest. For example: MUSEes_trans_en-fr have only 32000 translation out of ~60000 words need to translate and the missing of PT-EN in the CA as pivot.
- An advantage of using MUSE is it can get the translate word which not appear in the TIAD-2021 dataset. For example: From EN-FR:

diversified	diversifié	adjective	0.4661188

  the word "diversifié" not appear in the TIAD_2021 dataset, but appear in the corpus to build the fastText embeeings

4. Conclusions and future work

With this project, I have learned a lot of knowledge:

- Word2Embeedings and is application
- Cross-lingual word embedding with MUSE
- Processing the CSV dataset with Pandas
- Web scraping with selenium, beautiful soup

- Practice programming with Python, using bash scripts, using source version control with Github
- Reading paper and writing report

Future work:

Because of MUSE don't work well for example: using ES as pivot for the translate EN-FR having many fault: there are a lot of translate word is *"</s>",* so I think in the future, I will find down, why this error is happen.

The "sense" of word is important part to finding the translate, finding the method to apply "sense" to the system is a very interesting challenge

Reference

[1] B. Jason, “What Are Word Embeddings for Text?” https://machinelearningmastery.com/what-are-word-embeddings/ (accessed May 20, 2021).
[2] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” 1st Int. Conf. Learn. Represent. ICLR 2013 - Work. Track Proc., pp. 1–12, 2013.
[3] “Wiktionary - Wikipedia.” https://en.wikipedia.org/wiki/Wiktionary (accessed May 22, 2021).
[4] A. Conneau, G. Lample, R. Marc’Aurelio, L. Denoyer, and H. Jégou, “Word translation without parallel data,” arXiv, pp. 1–14, 2017.

Source code: https://github.com/batuan/TER_TIAD_2021

List of Acronyms

EN	English
FR	French
PT	Portuguese
CA	Catalan
ES	Spanish
POS	Part of speech
MUSE	Multilingual Unsupervised and Supervised Embeddings

Acknowledgements

The completion of this report was made possible through the patience and guidance of my advisor Carlos Ramisch who was always ready to answer my questions and give to me many suggestions.

THAI Ba Tuan