Semi-Supervised Learning for Bilingual Lexicon Induction

Gauthier Guinet
Ecole Polytechnique

Théophile de Bonnaventure
Ecole Polytechnique

Laurent Massoulié
Inria

ABSTRACT

We consider the problem of aligning two sets of continuous word representations, corresponding to languages, to a common space in order to infer a bilingual lexicon. It was recently shown that it is possible to infer such lexicon, without using any parallel data, by aligning word embeddings trained on monolingual data. Such line of work is called unsupervised bilingual induction. By wondering whether it was possible to gain experience in the progressive learning of several languages, we asked ourselves to what extent we could integrate the knowledge of a given set of languages when learning a new one, without having parallel data for the latter. In other words, while keeping the core problem of unsupervised learning in the latest step, we allowed the access to other corpora of idioms, hence the name semi-supervised. This led us to propose a novel formulation, considering the lexicon induction as a ranking problem for which we used recent tools of this machine learning field. Our experiments on standard benchmarks, inferring dictionary from English to more than 20 languages, show that our approach consistently outperforms existing state of the art benchmark. In addition, we deduce from this new scenario several relevant conclusions allowing a better understanding of the alignment phenomenon.

KEYWORDS

Bilingual Unsupervised Alignment, Word Embeddings, Learning to Rank, Wasserstein Procustes

1 INTRODUCTION

Word vectors are conceived to synthesize and quantify semantic nuances, using a few hundred coordinates. These are generally used in downstream tasks to improve generalization when the amount of data is scarce. The widespread use and successes of these "word embeddings" in monolingual tasks has inspired further research on the induction of multilingual word embeddings for two or more languages in the same vector space.

The starting point was the discovery [26] that word embedding spaces have similar structures across languages, even when considering distant language pairs like English and Vietnamese. More precisely, two sets of pre-trained vectors in different languages can be aligned to some extent: good word translations can be produced through a simple linear mapping between the two sets of embeddings. As an example, learning a direct mapping between Italian and Portuguese leads to a word translation accuracy of 78.1% with a nearest neighbor (NN) criterion.

Embeddings of translations and words with similar meaning are close (geometrically) in the shared cross-lingual vector space. This property makes them very effective for cross-lingual Natural Language Processing (NLP) tasks. The simplest way to evaluate the result is the Bilingual Lexicon Induction (BLI) criterion, which designs the percentage of the dictionary that can be correctly induced. Thus, BLI is often the first step towards several downstream tasks such as Part-Of-Speech (POS) tagging, parsing, document classification, language genealogy analysis or (unsupervised) machine translation.

Frequently, these common representations are learned through a two-step process, whether in a bilingual or multilingual setting. First, monolingual word representations are learned over large portions of text; these pre-formed representations are actually available for several languages and are widely used, such as the Fasttext Wikipedia vectors used in this work. Second, a correspondence between languages is learned in three ways: in a supervised manner if parallel dictionaries or data are available to be used for supervisory purposes, with minimal supervision, for example by using only identical strings, or in a completely unsupervised manner.

It is common practice in the literature on the subject to separate these two steps and not to address them simultaneously in a paper. Indeed, measuring the efficiency of the algorithm would lose its meaning if the corpus of vectors is not identical at the beginning. We will therefore use open-source data from Facebook containing embeddings of several dozen languages computed using Wikipedia data. The fact that the underlying text corpus is identical also helps to reinforce the isomorphic character of the point clusters.

Concerning the second point, although three different approaches exist, they are broadly based on the same ideas: the goal is to identify a subset of points that are then used as anchors points to achieve alignment. In the supervised approach, these are the words for which the translation is available. In the semi-supervised approach, we will gradually try to enrich the small initial corpus to have more and more anchor points. The non-supervised approach differs because there is no parallel corpus or dictionary between the two languages. The subtlety of the algorithms will be to release a potential dictionary and then to enrich it progressively.

We will focus in the following work on this third approach. Although it is a less frequent scenario, it is of great interest for several reasons. First of all, from a theoretical point of view, it provides a practical answer to a very interesting problem of information theory: given a set of texts in a totally unknown language, what information can we retrieve? The algorithms we chose to implement contrast neatly with the classical approach used until now. Finally, for very distinct languages or languages that are no longer used, it is true that the common corpus can be very thin.

Many developments have therefore taken place in recent years in this field of unsupervised bilingual lexicon induction. One of the recent discoveries, which pushed us to do this project, is the idea that using information from other languages during the training process helps improve translating language pairs. We came across this idea while searching for multi-alignment of languages instead of bi-alignment.

We chose to approach the problem in a different way, keeping an unsupervised alignment basis however. We asked ourselves to what extent we could integrate the knowledge of a given set of languages when learning a new one, without having parallel data for the latter. The scenario of multi-alignment unsupervised assumes that there is no parallel data for all language pairs. We think it is more realistic and useful to assume that we do not have this data only for the last language.

The underlying learning theme led us to formulate the problem as follows: **is it possible to gain experience in the progressive learning of several languages?** In other words, how can we make good use of the learning of several acquired languages to learn a new one? To our knowledge, this problem has never yet been addressed in the literature on the subject. This new formulation led us to consider the lexicon induction as a ranking problem for which we used recent tools of this machine learning field called Learning to Rank.

In summary, this paper make the following main contributions:

- We present a new approach for the unsupervised bilingual lexicon induction problem that consistently outperforms state-of-the-art methods on several language pairs. On a standard word translation retrieval benchmark, using 200k vocabularies, our method reaches 95.3% accuracy on English-Spanish while the best unsupervised approach is at 84.1%. By doing this, we set a new benchmark in the field.
- We conduced a study on the impact of the idioms used for the learning and for the prediction step, allowing us to have a better core understanding of our approach and to forecast the efficiency for a new idiom.
- Our results further strengthen in a new way the strong hypothesis that word embedding spaces have similar structures across languages. [26]

We will proceed as follows: Sections 2 and 3 will outline the state of the art and the different techniques used for unsupervised learning in this context. In particular, we will explain the Wasserstein Procustes approach for bilingual and multi alignment. We then emphasize the lexicon induction given the alignment. Section 4 presents the Learning to Rank key concepts alongside the TensorFlow framework used in our main algorithm. Section 5 describes our program, the different subtleties and the key parameters. Finally, Section 6 presents the experimental results we obtained.

2 UNSUPERVISED BILINGUAL ALIGNEMENT

In this section, we provide a brief overview of unsupervised bilingual alignment methods to learn a mapping between two sets of embeddings. The majority are divided into two stages: the actual alignment and lexicon induction, given the alignment. Even if the lexicon induction is often taken into account when aligning (directly or indirectly, through the loss function), this distinction is useful from a theoretical point of view.

Historically, the problem of word vector alignment has been formulated as as a quadratic problem. This approach, resulting from the supervised literature then allowed to presume the absence of lexicon without modifying to much the framework. That is why we will deal with it first in what follows.

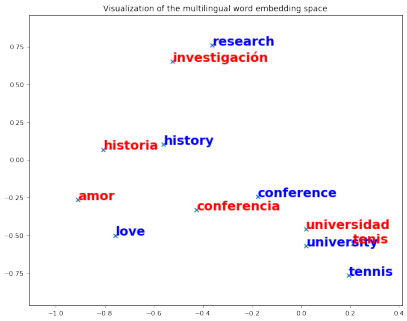

Figure 1: Word embeddings alignment (in dimension 2).

2.1 Orthogonal Procrustes Problem

Procustes is a method that aligns points if given the correspondences between them (supervised scenario). $\mathbf{X} \in \mathbb{R}^{n \times d}$ and $\mathbf{Y} \in \mathbb{R}^{n \times d}$ are the two sets of word embeddings or points and we suppose, as previously said, that we know which point $\mathbf{X}$ corresponds to which point Y. This leads us to solve the following least-square problem of optimization, looking for the W matrix performing the alignment [2] :

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times d}} \|\mathbf{XW} - \mathbf{Y}\|_2^2$$

We have access to a closed form solution with a cubic complexity. Restraining W to the set of orthogonal matrices O_d, improves the alignments for two reasons: it limits overfitting by reducing the size of the minimization space and allows to translate the idea of keeping distances and angles, resulting from the similarity in the space structure. The resulting problem is known as Orthogonal Procrustes and it also admits a closed form solution through a singular value decomposition (cubic complexity).

Thus, if their correspondences are known, the translation matrix between two sets of points can be inferred without too much difficulties. The next step leading to unsupervised learning is to discover these point correspondences using Wasserstein distance.

2.2 Wasserstein Distance

In a similar fashion, finding the correct mapping between two sets of word can be done by solving the following minimization problem:

$$\min_{\mathbf{P} \in \mathcal{P}_n} \|\mathbf{X} - \mathbf{PY}\|_2^2$$

$\mathcal{P}_n$ containing all the permutation matrices, the solution of the minimization, Pt will be an alignment matrix giving away the pair of words. This 1 to 1 mapping can be achieved thanks to the Hungarian algorithm. It is equivalent to solve the following linear program:

$$\max_{\mathbf{P} \in \mathcal{P}_n} \operatorname{tr}\left(\mathbf{X}^\top \mathbf{PY}\right)$$

The combination of the Procustes- Wasserstein minimization problem is the following:

$$\min_{\mathbf{Q} \in O_d} \min_{\mathbf{P} \in \mathcal{P}_n} \|\mathbf{XQ} - \mathbf{PY}\|_2^2$$

In order to solve this problem, the approach of [2] was to use a stochastic optimization algorithm. As solving separately those 2 problems was leading to bad local optima, their choice was to select a smaller batch of size$b$, and perform their minimization algorithm on these sub-samples. The batch is playing the role of anchors points. Combining this with a convex relaxation for an optimal initialization, it leads to the following algorithm:

Stochastic optimization algorithm

1: for $t = 1$ to T do
2: Draw $\mathbf{X}_t$ from $\mathbf{X}$ and $\mathbf{Y}_t$ from $\mathbf{Y}$, of size b
3: Compute the optimal matching between $\mathbf{X}_t$ and $\mathbf{Y}_t$ given the current orthogonal matrix $\mathbf{Q}_t$

$$\mathbf{P}_t = {}_{\mathbf{P}\in\mathcal{P}_b} \operatorname{tr}\left(\mathbf{Y}_t\mathbf{Q}_t^\top\mathbf{X}_t^\top\mathbf{P}\right)$$

4: Compute the gradient $\mathbf{G}_t$ with respect to $\mathbf{Q}$:

$$\mathbf{G}_t = -2\mathbf{X}_t^\top\mathbf{P}_t\mathbf{Y}_t$$

5: Perform a gradient step and project on the set of orthogonal matrices:

$$\mathbf{Q}_{t+1} = \Pi_{O_d}\left(\mathbf{Q}_t - \alpha\mathbf{G}_t\right)$$

For a matrix $\mathbf{M} \in \mathbb{R}^{d\times d}$, the projection is given by $\Pi_{O_d}(\mathbf{M}) = \mathbf{U}\mathbf{V}^\top$, with $\mathbf{U}\mathbf{S}\mathbf{V}^\top$ the singular value de- composition of M.
6: end for

2.3 Other unsupervised approaches

Other approaches exist but they are currently less efficient than the one described above for various reasons: complexity, efficiency.... We will briefly describe the two main ones below.

- **Optimal transport:** Optimal transport [32] formalizes the problem of finding a minimum cost mapping between two word embedding sets, viewed as discrete distributions. More precisely, they assume the following distributions:

$$\mu = \sum_{i=1}^{n} \mathbf{p}_i\delta_{\mathbf{x}^{(i)}}, \quad \nu = \sum_{j=1}^{m} \mathbf{q}_j\delta_{\mathbf{y}^{(i)}}$$

and look for a transportation map realizing:

$$\inf_T \left\{\int_{\mathcal{X}} c(\mathbf{x}, T(\mathbf{x}))d\mu(\mathbf{x}) | T_\#\mu = \nu\right\}$$

where the cost $c(\mathbf{x}, T(\mathbf{x}))$ is typically just $\|\mathbf{x}- T(\mathbf{x})\|$ and $T_\#\mu = \nu$ implies that the source points must exactly map to the targets. Yet, this transportation not always exist and a relaxation is used. Thus, the discrete optimal transport (DOT) problem consists of finding a plan Γ that solves

$$\min_{\Gamma\in\Pi(\mathbf{p},\mathbf{q})} \langle\Gamma, \mathbf{C}\rangle$$

where $\mathbf{C} \in \mathbb{R}^{n\times m}$ e.g., $C_{ij} = \left\|\mathbf{x}^{(i)} - \mathbf{y}^{(j)}\right\|$. is the cost matrix and the total cost induced by Γ is $\langle\Gamma, C\rangle := \sum_{ij}\Gamma_{ij}C_{ij}$. where Γ belongs to the polytope

$$\Pi(\mathbf{p}, \mathbf{q}) = \left\{\Gamma \in \mathbb{R}_+^{n\times m} | \Gamma 1_n = \mathbf{p}, \Gamma^\top 1_n = \mathbf{q}\right\}$$

A regularization is usually added, mostly through the form of an entropy penalization:

$$\min_{\Gamma\in\Pi(\mathbf{p},\mathbf{q})} \langle\Gamma, \mathbf{C}\rangle - \lambda H(\Gamma)$$

Some works [30] are based on these observations and then proposes algorithms, effective in our case because they adapt to the particularities of word embeddings. However, we notice that even if the efficiency is higher, the complexity is redibitive and does not allow the use of large vocabularies. Moreover, the research is more and more oriented towards an improvement of the algorithm described above with optimal transport tools rather than towards a different path. This is why we will not focus in particular on this track. We should however note the use of Gromov-Wasserstein distance [31], which allows to calculate in an innovative way the distance between languages, although they are in distinct spaces, enabling to compare the metric spaces directly instead of samples across the spaces.

- **Adversarial Training:** Another popular alternative approach derived from the literature on generative adversarial network [29] is to align point clouds without cross-lingual supervision by training a discriminator and a generator [5]. The discriminator aims at maximizing its ability to identify the origin of an embedding, and the generator of W aims at preventing the discriminator from doing so by making WX and Y as similar as possible. They note θ_D. the discriminator parameters and consider the probability $P_{\theta_D}(\text{ source } = 1|z)$ that a vector z is the mapping of a source embedding (as opposed to a target embedding) according to the discriminator. The discriminator loss can then be written as:

$$\mathcal{L}_D\left(\theta_D|W\right) = -\frac{1}{n}\sum_{i=1}^{n}\log P_{\theta_D}\left(\text{source} = 1|Wx_i\right) - \frac{1}{m}\sum_{i=1}^{m}\log P_{\theta_D}\left(\text{source} = 0|y_i\right)$$

On the other hand, the loss of the generator is:

$$\mathcal{L}_W\left(W|\theta_D\right) = -\frac{1}{n}\sum_{i=1}^{n}\log P_{\theta_D}\left(\text{source} = 0|Wx_i\right) - \frac{1}{m}\sum_{i=1}^{m}\log P_{\theta_D}\left(\text{source} = 1|y_i\right)$$

For every input sample, the discriminator and the mapping matrix W are trained successively with stochastic gradient updates to respectively minimize $\mathcal{L}_W$ and $\mathcal{L}_D$. Yet, papers [5] on the subject show that, although innovative, this framework is more useful as a pre-training for the classical model than as a full-fledged algorithm. Hence our choice not to explore in more details this avenue.

2.4 Multilingual alignment

A natural way to improve the efficiency of these algorithms is to consider more than 2 languages. Thus,when it comes to aligning multiple languages together, two principle approaches quickly come to mind and correspond to two types of optimization problems:

- Align all languages to one pivot language, often English, without taking into account for the loss function other alignments. This leads to low complexity but also to low efficiency between the very distinct language, forced to transit through English.
- Align all language pairs, by putting them all in the loss function, without giving importance to any one in particular. If this improves the efficiency of the algorithm, the counterpart is in the complexity, which is very important because it is quadratic in the number of languages.

A trade-off must therefore be found between these two approaches.

Let us consider $\mathbf{X}_i$ word embeddings for each language i, i=0 can be considered as the reference language, $\mathbf{W}_i$ is the mapping matrix we want to learn and $\mathbf{P}_i$ the permutation matrix. The alignment of multiple languages using a reference language as pivot can be resumed by the following problem:

$$\min_{\mathbf{W}_i \in O_d, \mathbf{P}_i \in \mathcal{P}_n} \sum_i \ell\left(\mathbf{X}_i \mathbf{W}_i, \mathbf{P}_i \mathbf{X}_0\right)$$

As said above, although this method gives satisfying results concerning the translations towards the reference language, it provides poor alignment for the secondary languages between themselves. Therefore an interesting way of jointly aligning multiple languages to a common space has been brought through by Alaux and al.[1]. The idea is to consider each interaction between two given languages, therefore the previous sum becomes a double sum with two indexes i and j. To prevent the complexity from being to high and in order to keep track and control over the different translations, each translation between two languages is given a weight α_{ij} :

$$\min_{\mathbf{Q}_i \in O_d, \mathbf{P}_{ij} \in \mathcal{P}_n} \sum_{i,j} \alpha_{ij} \ell\left(\mathbf{X}_i \mathbf{Q}_i, \mathbf{P}_{ij} \mathbf{X}_j \mathbf{Q}_j\right)$$

The choice of these weights depends on the importance we want to give to the translation from language i to language j.
The previous knowledge we have on the similarities between two languages can come at hand here, for they will have a direct influence on the choice of the weight. However choosing the appropriate weights can be uneasy. For instance giving a high weight to a pair of close languages can be unnecessary and doing the same for two distant languages can be a waste of computation. In order to reach this minimization effectively, they use an algorithm very similar to the stochastic optimization algorithm described above.

At the beginning, we wanted to use this algorithm to incorporate exogenous knowledge about languages to propose constants α_{ij} more relevant and leading to greater efficiency. Different techniques could result in these parameters: from the mathematical literature such as the Gromov-Wasserstein distance evoked above or from the linguistic literature, using the etymological tree of languages to approximate their degree of proximity or even from both. In the article implementing this algorithm, it is however specified that the final α_{ij} are actually very simple: they are N if we consider a link to the pivot or 1 otherwise. Practical simulations have also led us to doubt the efficiency of this idea. This is why we decided to focus on the idea below that seemed more promising rather than on multialignments.

3 WORD TRANSLATION AS A RETRIEVAL TASK: POST-ALIGNMENT LEXICON INDUCTION

The core idea of the least-square problem of optimization in Wasserstein Procustes is to minimize the distance between a word and its translation. Hence, given the alignment, the inference part first just consisted in finding the nearest neighbors (NN). Yet, this criterion had a mayor issue: Nearest neighbors are by nature asymmetric: y being a K-NN of x does not imply that x is a K-NN of y. In high-dimensional spaces, this leads to a phenomenon that is detrimental to matching pairs based on a nearest neighbor rule: some vectors, called hubs, are with high probability nearest neighbors of many other points, while others (anti-hubs) are not nearest neighbors of any point. [28]
Two solutions to this problem have been brought through new criteria, aiming at giving similarity measure between two embeddings, thus allowing to match them appropriately. Among them, the most popular is Cross-Domain Similarity Local Scaling (CSLS) [5]. Other exist such as Inverted Softmax (ISF)[4], yet they usually require to estimate noisy parameter in an unsupervised setting where we do not have a direct cross-validation criterion.

The idea behind CSLS is quite simple: it is a matter of calculating a cosine similarity between the two vectors, subtracting a penalty if one or both of the vectors is also similar at many other points. More formally, we denote by $\mathcal{N}_{\mathrm{T}}\left(W x_s\right)$ the neighboors of x_S for the target language, after the alignment (hence the presence of $\mathbf{W}$). Similarly we denote by $\mathcal{N}_{\mathrm{S}}\left(y_t\right)$ the neighborhood associated with a word t of the target language. The penalty term we consider is the mean similarity of a source embedding x_s to its target neighborhood:

$$r_{\mathrm{T}}\left(W x_s\right) = \frac{1}{K} \sum_{y_t \in \mathcal{N}_{\mathrm{T}}\left(W x_s\right)} \cos\left(W x_s, y_t\right)$$

where cos(...) is the cosine similarity. Likewise we denote by $r_{\mathrm{S}}\left(y_t\right)$ the mean similarity of a target word y_t to its neighborhood. Finally, the CSLS is defined as:

$$\mathrm{CSLS}\left(W x_s, y_t\right) = 2\cos\left(W x_s, y_t\right) - r_{\mathrm{T}}\left(W x_s\right) - r_{\mathrm{S}}\left(y_t\right)$$

However, it may seem irrelevant to align the embedding words with the NN criterion metric and to use the CSLS criterion in the inference phase. Indeed, it creates a discrepancy between the learning of the translation model and the inference: the global minimum on the set of vectors of one does not necessarily correspond to the one of the other. This naturally led to modify the least-square optimization problem to propose a loss function associated with CSLS [27]. By assuming that word vectors are ℓ_2- normalized, we have $\cos\left(\mathbf{W}\mathbf{x}_i, \mathbf{y}_i\right) = \mathbf{x}_i^\top \mathbf{W}^\top \mathbf{y}_i$. Similarly, we have

$$\left\|\mathbf{y}_j - \mathbf{W}\mathbf{x}_i\right\|_2^2 = 2 - 2\mathbf{x}_i^\top \mathbf{W}^\top \mathbf{y}_j.$$

Therefore, finding the k nearest neighbors of $\mathbf{W}\mathbf{x}_i$ among the elements of $\mathbf{Y}$ is equivalent to finding the k elements of $\mathbf{Y}$ which have the largest dot product with $\mathbf{W}\mathbf{x}_i$. This equivalent formulation is adopted because it leads to a convex formulation when relaxing the orthogonality constraint on $\mathbf{W}$. This optimization problem with

the Relaxed CSLS loss (RCSLS) is written as:

$$\begin{aligned}\min_{\mathbf{W}\in O_d} \frac{1}{n}\sum_{i=1}^{n} -2\mathbf{x}_i^\top \mathbf{W}^\top \mathbf{y}_i \\ +\frac{1}{k}\sum_{\mathbf{y}_j\in\mathcal{N}_Y(\mathbf{W}\mathbf{x}_i)} \mathbf{x}_i^\top \mathbf{W}^\top \mathbf{y}_j \\ +\frac{1}{k}\sum_{\mathbf{W}\mathbf{x}_j\in\mathcal{N}_X(\mathbf{y}_i)} \mathbf{x}_j^\top \mathbf{W}^\top \mathbf{y}_i\end{aligned}$$

A convex relaxation can then be computed, by considering the convex hull of O_d, i.e., the unit ball of the spectral norm. The results of the papers [5] point out that RCSLS outperforms the state of the art by, on average, 3 to 4% in accuracy compared to benchmark. This shows the importance of using the same criterion during training and inference.

Such an improvement using a relatively simple deterministic function led us to wonder whether we could go even further in improving performance. More precisely, considering Word translation as a retrieval task, the framework implemented was that of a ranking problem. In order to find the right translation, it was important to optimally rank potential candidates. This naturally led us to want to clearly define this ranking problem and to use the state of the art research on raking to tackle it. In this framework, the use of simple deterministic criteria such as NN, CSLS or ISF was a low-tech answer and left a large field of potential improvement to be explored.

However, we wanted to keep the unsupervised framework, hence the idea of training the learning to rank algorithms on the learning of the translation of a language pair, English-Spanish for instance, assuming the existence of a dictionary. This would then allow us to apply the learning to rank algorithm for another language pair without dictionary, English-Italian for instance. Similarly to the case of CSLS, the criterion can be tested first at the end of the alignment carried out thanks to the Procustes-Wasserstein method. Then, in a second step, it can be integrated directly through the loss function in the alignment step. The following will quickly present the learning to rank framework in order to understand our implementation in more detail.

4 LEARNING TO RANK

Although ranking problems can be often found similar to regression and classification, they are essentially different. When classification and regression models intend to predict a score for each individual datapoint as accurately as possible, ranking algorithms aim at sorting out the entire dataset, thus putting forward the more relevant points. Learning to rank methods can mostly be put into three different groups : pointwise, pairwise and listwise algorithms. Learning to rank algorithms are very important in the field of relevance ranking. This appears to be crucial in information retrieval and search engine optimization.[10].
Deep learning techniques have recently provided great help for information retrieval, and therefore the most modern learning to rank methods use deep neural networks. For example the DeepRank algorithm has been created to simulate the human judgement process in the relevance ranking of various elements. DeepRank includes three consecutive actions: a Detection Strategy, a measure network and an aggregation network.

4.1 Neural learning to rank with Tensorflow

We implemented a learning to rank algorithm from a library called Tensorflow which uses neural networks for ranking.
The benefits of using neural networks are of two kinds: First the performance is clearly boosted in comparison to standard Learning to Rank tasks, and secondly the algorithm will learn the model directly from the data without having to handcraft any feature (for instance, for a document these handcrafted features would be word scores, page quality, url length...) [7].

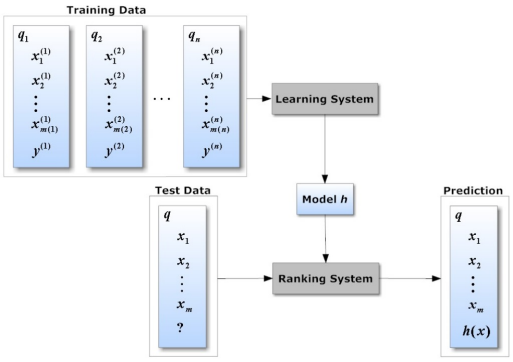

Figure 2: Learning to rank algorithms usual architecture [8].

Tensorflow is a deep learning framework in which computation is a dataflow graph with the operations being the nodes and the tensors the edges. Tensorflow enables us to use deep neural networks for our project and output various graphs and figures, since Tensorflow monitors particularly efficiently the performance of it's neural networks.
The TensorFlow Ranking Algorithm we implemented has the following architecture:

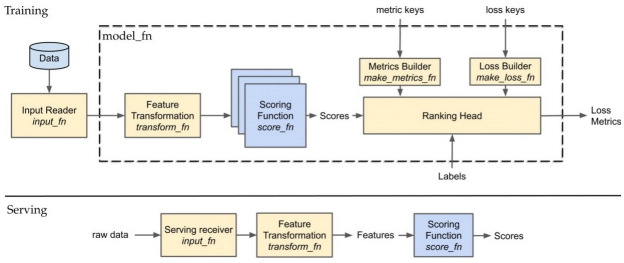

Figure 3: TensorFlow-Ranking Architecture [7].

One notable asset is that the algorithm requires very little transformation of the input data, thus allowing us to make it easily communicate with the alignment FastText data sets.
The core component of the TensorFlow ranking framework is the model-fn function, which receives labels and features and outputs loss, prediction, metrics and training ops according to the chosen mode (Train, Eval or Predict).
The goal of learning-to-rank is to learn a ranking function f from

training data so that items as ordered by f reach maximal utility. Practically, this ranking function appears to be more of a scoring function. This function is learned by minimizing a loss function, or maximizing utility, which are calculated thanks to ranking metrics. We will now go through the different components of the algorithm's architecture.

4.2 Scoring Function

Since finding a direct permutation in order to sort the input data appears to be excessively costly, a score-and-sort approach is used instead.
If $h : X^n \rightarrow \mathbb{R}^n$ is the scoring function taking a list of items x as input and providing a list of scores $\hat{y}$. Then $h(.)|_k$ will represent the k^{th} dimension of $h(.)$. h induces therefore a permutation π such that $h(x)|_{\pi^{-1}(r)}$ is monotonically decreasing for increasing ranks r. This allows us to build a shrewd scoring function without having a high complexity.

The scoring function is a parametrized function that can be either single item or multi item. A single item scoring function will be alike:

$$h(X) = [f(x1); f(x2); \ldots; f(xn)]$$

with the x_i being the features of one data point, h outputs a global score. In comparison a multi item scoring function takes for input a group of data points:

$$h(X) = [f(x_{11}, x_{12}...); f(x_{21}, x_{22}..); \ldots; f(x_{n1}, x_{n2}, x_{n3}....)]$$

The group size of the input sample will have its importance: Indeed the higher the group size is, the more interactions between data points will be taken into account. However big sized groups have a high computation cost since each variable is considered regarding the other ones in the group. The prize of a rise in efficiency is here a higher computation time. The relation between group size and efficiency will be studied in Figure 5.
The purpose of the scoring function is to map a set of input items x_i to a set of scores. Scoring functions can be linear functions [19]., boosted weak learners [20], gradientboosted trees [21] [22], support vector machines [23] [24], and neural networks [25] for instance. TensorFlow provides a neural network as basis of the scoring function.

4.3 Ranking metrics

The ranking head's purpose displayed on the TensorFlow architecture figure is meant to compute ranking metrics, scores and ranking losses.
The ranking metrics will enable us to measure the utility of an ordered list of items. The goal is to have as few as possible errors at higher ranked positions. Below are some examples of most used ranking metrics:

$$RR(\pi, y) = \frac{1}{\min_j \left\{ y_{\pi^{-1}(j)} > 0 \right\}}$$

$$RP(\pi, y) = \frac{\sum_{j=1}^{n} y_j \pi(j)}{\sum_{j=1}^{n} y_j}$$

$$\text{DCG}(\pi, y) = \sum_{j=1}^{n} \frac{2^{y_j} - 1}{\log_2(1 + \pi(j))}$$

$$\text{NDCG}(\pi, y) = \frac{\text{DCG}(\pi, y)}{\text{DCG}(\pi^*, y)}$$

where $y_i \in \boldsymbol{y}$ are the labels allowing us to obtain π^*, and $\pi(i)$ is the rank of the i^{th} item in x.RR is the reciprocal rank of the first relevant item. RP is the positions of items weighted by their relevance values. DCG is the Discounted Cumulative Gain, and NDCG is DCG normalized by the maximum DCG obtained from the ideal ranked list π^*. The mean of these metrics is calculated and provided if many example samples are given. For instance the mean of the reciprocal rank would be $MRR = \frac{1}{m} \sum_{k=1}^{N} RR\left(\pi_k, \boldsymbol{y}_k\right)$

4.4 Loss functions

The intern goal of the learning to rank algorithm will be to find a specific f^* function that minimizes the empirical loss obtained thanks to the training data:

$$f^* = \underset{f: X^n \rightarrow \Pi^n}{\text{argmin}} \frac{1}{m} \sum_{(\boldsymbol{x}, \pi^*) \in S^m} \ell\left(\pi^*, f(\boldsymbol{x})\right)$$

with $S^m = \{(\boldsymbol{x}, \pi^*) \,|\boldsymbol{x} \in X^n, \pi^* \in \Pi^n\}$ being the training data set of m items. π^* is a permutation that can be seen as a ranking of the data x induced by a list of labels. The exact form of this loss function will depend on the approach chosen should it be pointwise, pairwise or listwise [9].

4.5 Pointwise approach

In a pointwise learning to rank algorithm, the X input is an embedding and the Y output is a relevance degree. The algorithm will be trained to maximise the relevance degree of a given input. For instance logistic regression is an example of a pointwise algorithm. In our context, we need to get as an output a classification of the neighbouring word embeddings of our word input X. A relevance numerical estimation will not provide enough accuracy for our task and will require sorting algorithms to end up with the desired output.

4.6 Pairwise approach

Pairwise algorithms such as RankSVM receive in input a pair of embeddings and outputs the preferred one between both of them. This allows us to choose between two embeddings. However our need can concern a number n of input embeddings with n being higher than 2 if we want to train a function intelligent enough to choose a translation word among many surrounding ones.
The goal here is to order correctly given pairs of embeddings. [11].This task is done by minimizing the following loss function:

$$\sum_{q} \sum_{i,j, l_i^q > l_j^q}^{m_q} \ell\left(f\left(\mathbf{x}_i^q\right) - f\left(\mathbf{x}_j^q\right)\right)$$

The RankSVM algorithm uses $\ell(t) = \max(0, 1 - t)$ [12] whereas the RankNet program has $\ell(t) = \log(1 + \exp(-t))$ [13] and the GBRank method uses a quadratic loss, close to the Rank SVM one: $\ell(t) = \max(0, 1 - t)^2$

4.7 Listwise approach

The Listwise method receives as an input a list of embeddings and outputs the same list sorted.
In this approach the loss function recieves in input all the embeddings given by the query: $\ell\left(\left\{f\left(\mathbf{x}_j^q\right)\right\}, \left\{l_j^q\right\}\right)$ for $j = 1 \dots m_q$
There are mostly two types of listwise programs : the first type take no notice of Information Relevance measures while the training is running. This is the case for ListNet [15] and ListMLE [16] . The other type attempts to optimize the Information Relevance measure throughout the training. The famous examples are AdaRank [17] and SoftRank [18] . We observe that, for a majority of problems, a listwise loss performs better than a pairwise loss, which is in turn better than a pointwise loss.

Taking into consideration the previous points, the following section will present our work, combining these different techniques in an innovative way.

5 RUBI: RANKED UNSUPERVISED BILINGUAL INDUCTION

Motivations: Let's describe more precisely the functioning of our algorithm, denoted RUBI, although already mentioned in previous sections. Two points guided our approach:

- From a linguistic point of view, there is obviously a learning to learn phenomenon for languages. We observe that by assimilating the structure of the new language, its grammar and vocabulary to one of the already known languages, it is easier for us to create links that help learning. It is the search for these links that motivates us and we are convinced that they can be useful when inferring vocabulary.
- improvement induced by the use of the CSLS criterion suggests that there are complex geometrical phenomena (going beyond the above-mentioned existence of hubs) within the representations of languages, both ante and post-alignment. Understanding these phenomena can lead to greatly increased efficiency.

Framework: Our goal is the same as for unsupervised bilingual alignment: we have a source language A and a target language B with no parallel data between the two. We want to derive an A-B dictionary, a classic BLI task. The specificity of our study is to assume that we also have a C language and an A-C dictionary at our disposal. To set up the learning to learn procedure, we proceed in 2 steps:

- **Learning:** Using the Procustes-Wasserstein algorithm, we align languages A and C in an unsupervised way. We then build a corpus of queries between the words from language A known from our dictionary and their potential translation into language C. Classical methods proposed the translation that maximized the NN or CSLS criteria. In our case, we use deep learning as part of our learning to rank framework to find a more complex criterion. One of the innovative features of our work is therefore to allow access to a much larger class of functions for the vocabulary induction stage. A sub-part of the dictionary is used for cross-validation. The way of rating the relevance of the potential translations, the inputs of the algorithm, the loss functions are all parameters that we studied and that are described in the next section.
- **Prediction:** We thus have at the end of the training an algorithm taking as input a vocabulary word, in the form of an embedding as well as a list of potential translations. The output of our algorithm is the list sorted according to the learned criteria of these possible translations, the first word corresponding to the most probable translation and so on. We first perform the alignment of languages A and B using again the Procustes-Wasserstein algorithm. In a second step, thanks to the learning to rank, we perform the lexicon induction step.

Choices and expected results:A number of important points should be made:

- We assume that C's learning from A is of interest to B's learning. The truthfulness of this hypothesis is an interesting fact that we will be able to study. Realistically, it seems that learning Chinese from English will help us less to learn Italian than if we had chosen Spanish in the first place. However, it gives us a practical measure of proximity between languages, which is not obvious to infer at first glance and can be very interesting.
- We choose to use the Procustes-Wasserstein algorithm in the translation stage because we believe that maintaining the alignment method throughout our study allows us to truly evaluate its effectiveness and the specific geometric changes it induces. Since we assume that we have a dictionary at our disposal, we could use supervised methods, but they would not work in the same way.
- The multi-alignment scenario does not assume the existence of this dictionary. Realistically, however, it is consistent with existing use-case to assume that it is available.

Finally, a final conceptual point is important to raise. In the context of the CSLS criterion, we have seen in the above that its use after alignment has improved. However, actually incorporating it in the alignment phase by modifying the loss function has allowed for greater consistency and a second improvement. However, these two changes were separated. Yet, the learning to rank framework is quite different. The main reason is the non-linearity resulting from deep-learning, unlike CSLS. The global optimization is therefore much more complex and does not allow a relaxation to get back to a convex case. However, it is an area for improvement to be considered very seriously for future work.

6 EXPERIMENTS AND RESULTS

6.1 Unsupervised word translation

Implementation details: The general parameters used are described bellow. They are studied in more depth in the following subsection.

- Fasttext Word embeddings, learned on Wikipedia corpus, dimension 300, size 200 000 words. Dictionary of the 5,000

Method	EN-ES	ES-EN	EN-FR	FR-EN
Wass. Proc. - NN	77.2	75.6	75.0	72.1
Wass. Proc. - CSLS	79.8	81.8	79.8	78.0
Wass. Proc. - ISF	80.2	80.3	79.6	77.2
Adv. - NN	69.8	71.3	70.4	61.9
Adv. -CSLS	75.7	79.7	77.8	71.2
RCSLS+spectral	83.5	85.7	82.3	84.1
RCSLS	84.1	86.3	83.3	84.1
RUBI	**93.3** (*DE*)	**91.6** (*FR*)	**93.8** (*NL*)	**91.9** (*IT*)
	EN-DE	DE-EN	EN-RU	RU-EN
Wass. Proc. - NN	66.0	62.9	32.6	48.6
Wass. Proc. - CSLS	69.4	66.4	37.5	50.3
Wass. Proc. - ISF	66.9	64.2	36.9	50.3
Adv. - NN	63.1	59.6	29.1	41.5
Adv. -CSLS	70.1	66.4	37.2	48.1
RCSLS+spectral	78.2	75.8	56.1	66.5
RCSLS	79.1	76.3	57.9	67.2
RUBI	**93.6** (*HU*)	**89.8** (*FR*)	**83.7** (*HU*)	-

Table 1: Benchmark Results for Bilingual Lexicon Induction

most frequent words for training and assessment, dictionary of the next 1,500 words for cross validation.

- Alignment: Procustes-Wasserstein algorithm, 5 epoch of 5000 iterations. Learning rate of 0.5. Batch size of 500.
- Learning to Rank: 100000 iterations, batch size de 32, dropout rate of 0.5 for regularization, 3 hidden layers of size 256, 128 and 64, Adagrad Optimizer, group size of 4, queries of 10 potential translations (selected using the NN) for each 5000 words in dictionary. 11 Features (Similarity and CSLS(i) for i in [1,10] relative to the query word).

Baselines: We compare our method with Wasserstein-Procrustes (Wass. Proc.) [2], as well as two unsupervised approaches: the adversarial training (adversarial) of Conneau et al.[5] and the Relaxed CSLS loss (RCSLS) [27]. All the numbers are taken from their papers. The reference benchmark is the translation from English to French, Spanish, Russian and German as well as the reverse translation.

Mains results: In order to quantitatively assess the quality of each approach, we consider the problem of bilingual lexicon induction. Following standard practice, we report the precision at one. Contrary to other methods, we use an auxiliary language, which we show in brackets next to the result. We then specifically study this choice in relation to the translation. The general parameters used are described above. Table 1 summarizes our results, compared to existing state of the art on reference BLI task. Our method outperforms all existing methodology, with a vast margin. Hence, Our assumption that learning a language for training purposes brings a lot is confirmed. Without even going into the alignment phase, our criterion brings a real gain for unsupervised translation, for a reasonable computational complexity (about fifteen minutes per translation). the major contribution is for English-Russian translation, with a gain of 35% compared to the best existing method (Russian-English translation is not included because we had no access to dictionaries required for the learning phase).

Pivot	ES-EN	FR-EN	DE-EN	PT-EN
ES	-	91.6	83.3	88.8
FR	**91.6**	-	**89.8**	**89.8**
IT	91.3	**91.9**	88.5	89.4
DE	90.0	91.5	-	89.8
PT	91.4	91.6	89.8	-

Table 2: BLI efficiency for reverse translation to English

Table 2 summarizes for French, Italian, Russian, Portuguese and German the BLI for translation to English. These were the only languages for which we had dictionaries between each pair. This allowed us to set up a learning step without English being one of the two languages, which was impossible for the more complete list of languages we used afterwards. The best result is the one shown above in Table 1.

6.2 Ablation Study

In this section, we evaluate the impact of some of our implementation choices on the performance of RUBI. We focus in particular on the loss function, the query size, the group size as well as the features and relevance system used .

6.2.1 ***Impact of Loss function:*** As described above, the learning to rank framework allows the use of numerous loss functions corresponding to different scenarios (0-1 relevance, pointwise, listwise and pairwise...). In our case, we wanted to maximize the BLI criterion, i.e. to have the maximum performance just for the top of the list in order to be able to compare our work with the existing literature. Other objectives can be considered, such as maximizing the presence of the right translation among the first X suggestions and then manually look for the correct translation in this subset. We have therefore added the BLI criterion among the other criteria (NDCG, RP, RR...). The closest existing one was NDCG@1 (1st position). We then tested a large majority of the existing loss functions to see which one was the most efficient given our objective. The graph below presents the results for 5 of these functions, representative of the different existing categories. The evaluation criterion is the BLI for EN-ES translation. The two functions that perform best are the Approximate NDCG loss (which maximizes a differentiable approximation of NDCG) and the MLE loss list (which maximizes the likelihood loss of the probability distribution). In other experiments, we found that these two loss functions continue to perform similarly, hence our default choice of Approx NDCG. We have also shown this function with a group size of 1 and 2 (parameter described in the learning to rank section).

6.2.2 ***Impact of group size:*** As seen above, group size has a great influence on our criterion. This is quite consistent given our concern for ranking compared to other embeddings. The dilemma is however to optimize the computation time because increasing the group size exponentially increases the number of calculations. The graph below presents the BLI criterion for EN-ES, using two loss functions and varying the group size. A clear improvement can be observed. However, the biggest increase seems to occur when changing the group size from 1 to 2 and then the curve stabilizes.

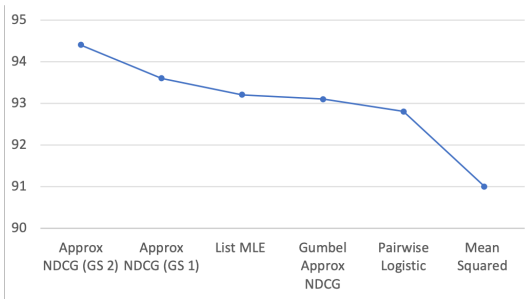

Figure 4: Loss function Impact.

Therefore, in our computations, we used mainly a group size of 2 and often of 4 when looking for greater accuracy.

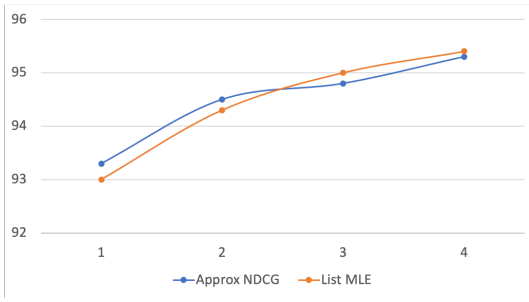

Figure 5: Group Size Impact.

6.2.3 ***Impact of feature system:*** As stated above, our framework receive as input a query list i.e. for each word in the dictionary, a list of potential translation. For each of these potential translation, we compute a relevance label (estimated using ground truth) and a list of features. The relevance label is used only for the training and we be studied bellow. The features for the each potential translation in a query can incorporate several elements:

- the word embedding of the potential translation (size 300)
- the word embedding of the query (size 300)
- pre-computed features such a distance to query word in the aligned vector space, CSLS distance, ISF....

Those features are crucial for the learning as it will fully rely on it. At first, we decided to only use the word embedding of the potential translation and of the query. That gave us a 600 feature list. However, after several experiments, we noticed that the learning to rank algorithm, despite the variation of the parameters, was not able to learn relevant information from these 600 features, the performance was poor. The function learned through deep learning was less efficient than a simple Euclidean distance between the potential translation and the query (NN criterion). In fact, after consulting the literature, we realised that using such a number of features is not very common. Most algorithms were only using pre-computed features (often less than a hundred). Although this information is already interesting in itself, we therefore turned to the second approach. We chose to restrict ourselves to certain well-specified types of pre-computed features in order to evaluate their full impact. More precisely, for a fixed k parameter, we provided as features the euclidean distance to the query, as well as the CSLS(i) "distance" for i ranging from 1 to k. In other words, we provided information about the neighborhood through the penalties described in the section on CSLS. In the context of this work, we did not want to use other features (ISF in particular) to focus specifically on the contribution of CSLS but this is an easy improvement path to exploit for future work. As outlined above, this simple framework allows a considerable improvement of the BLI. Below, we describe the evolution of the BLI for EN-ES translation by varying the k parameter. k=0 correspond to the use only of euclidean distance to the query. We observe a major increase for k from 0 to 1, a lesser increase for k from 1 to 4-5 and stabilization thereafter, with a slight maximum towards k=14.

This leads us to believe that the relevant information for the algorithm is just in the close neighborhood of the point (i.e. the few closest neighbors) and the addition of features describing a more distant neighborhood brings only marginal information. This preliminary conclusion is clearly worth further study. Indeed, the functioning of the neural network at the very heart of the learning to rank algorithm is that of a black box and this conclusion deserves to be reinforced. Its veracity would allow a better understanding of the phenomenon of hubbness in the context of word embeddings clouds.

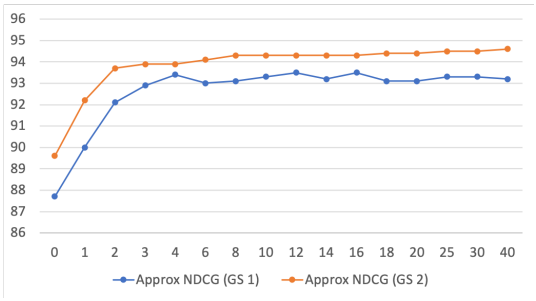

Figure 6: CSLS feature Impact.

6.2.4 ***Impact of query relevance:*** The learning to rank framework used makes it possible to rate the relevance of a word embedding in relation to a query thanks to a system of integers. A relevance of 0 translates a non relevance, then the higher the relevance is, the more the word embedding will be relevant i.e. close to the translation. For the ranking of the list of potential translations, it is these relevancies that will be used. For the pointwise loss function (i.e. regression problem), it will be a matter of predicting these labels. In the peerwise and listwise functions, it is more complex. We have considered 3 relevance scenarios. This relevance setting is the key of a good algorithm as it represent the function we aim to learn. With this in mind, we tried three different options:

- **Binary Relevance:** The correct translation and its synonyms have a relevance of 1, the other candidates have a relevance of 0 (not relevant). This system allows to use some specific loss functions very efficiently in these scenarios (like sigmoid cross entropy loss). However, the context of our study was

not very appropriate for this relevance system. Indeed, we often had only one correct translation per query, for about ten erroneous translations. The algorithm was therefore not able to learn about the corpus of good translations and almost systematically predicted that the translation was erroneous, i.e. a relevance of 0.

- **Continuous Relevance:** On the other hand, we asked ourselves how to translate the relevance of a potential candidate for translation, even if it is not the right translation or a synonym. In other words, how do you extract the information it contains? The previous approach only considered the question: "is this a correct translation ?" instead of asking "to what extend is this a correct translation ?" In this context, we had 3 suggestions:
  - **Intra-distance:** The core idea of word embeddings is to translate the contextual proximity of 2 words into a proximity in terms of distance in a space. Therefore, if we are interested in the relevance of a word ("dog") in the context of a translation ("chat"), we can consider the distance between the embeddings of the candidate ("dog") and the correct translation ("cat"). We talk about intra-distance because it's a distance in the target space. When these distances are calculated, we can then classify the words by proximity and give them a label of relevance thanks to this. Although appealing on paper, this approach led to poor results for a hidden reason: the learning to rank algorithm sought to maximize the ordering of the response list. When it was given too many distinct possible labels, it made it easier to make a large number of correct predictions about the relative ranking in the answer list. Maximizing the top of the list, which is our main interest, was no longer a priority at all. The other concern was that in very large dimensions, the notion of distance also lost its meaning. It was therefore less relevant to note these distances.
  - **Extra-distance:** In a logic close to the previous method, we wondered if it could be interesting to consider the distances in the source space instead of those in the target space. In other words, instead of looking at the distance between our candidate ("dog") and the correct translation ("cat") of the target word ("chat"), we could look at the translation of the candidate in the source space ("chien") and look at the distance between this translation and the word to be translated. This idea was therefore based on the word embeddings in the source space and did not use the word embeddings in the target space. Unfortunately, we didn't have a dictionary for each of the 200,000 words used, so this passage in the source space was complicated to settle.
  - **Exogenous distance:** We finally wondered if we could incorporate an exogenous distance thanks to external algorithms that could provide a proximity between words other than the one given by the embeddings (syntactic, etymological...). This track remains in our suggestions for improvement but the poor results obtained with the intra-distance method dissuaded us from exploring it right away.
  - **Semi-binary:** The solution chosen for the implementation was a trade off between the two previous approaches: the correct translation and its synonyms receive a label of 2 and the other words receive a label of 1 (very low relevance). This pushes the algorithm to focus as much as possible on the correct translation while using the above information in the words (to a lesser extent).

6.2.5 ***Impact of query size:*** Finally, we studied the impact of the size of the query i.e. the number of potential translations provided for each word. A large number of potential translations gives you more choice but the risk of getting it wrong is greater. One must also ask whether the algorithm is able to learn in a relevant way if it is provided with a large amount of information (given that a semi-binary relevance system is used). The experience setting is the same as for the previous points (BLI induction for EN-ES). There is a low incidence of the number of queries on the results, a very slight but perceptible decrease. The algorithm is therefore able, despite a large number of candidates, to discern the correct information.

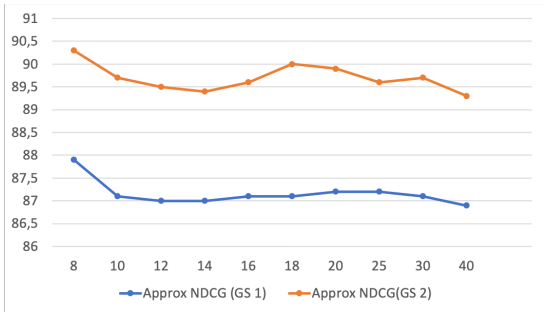

Figure 7: Query Size Impact.

6.3 Impact of learned idiom in inference

Experiment details: In this section, we evaluate the impact of the idiom used for learning when it comes to translation. As it is very costly to compute, we only used our 4 reference languages as target to begin: Spanish (ES), French (FR), German (DE) and Russian (RU). More computation are currently on their way. Facebook data gave us access to dictionary from English to more than 20 idioms. The following table present the BLI criterion for the translation using the idiom in the left for the learning process. The last line is the benchmark for the translation associated with this pair, using one of the other previously described unsupervised learning methodology. The last columns deals with the learning part: the first digit is the is the BLI criterion obtain for the learning language at the end of the training. The second one is the BLI criterion for the learning language obtained when aligning and translating this idiom with English, using Wasserstein Procrustes Unsupervised Alignment and CSLS criterion)[2]. If the first number does not appear, it means that the alignment is of poor quality and that the learning step leads to blatant over-feating, so the number makes little sense.

Mains results: Several conclusions can be drawn from this table:

- **Prediction stability:** There is a strong stability for the prediction of a given language. Except for languages very poorly aligned with English, the results seem to be globally identical when the idiom used for learning varies (difference of less than 1%). This leads to more remarks. Although the learning to rank algorithm is a black box, it seems that it always leads to a similar criterion at the end of the learning process, a criterion much more powerful than those used so far (NN, CSLS, ISF among others). The proximity between the learning language and the prediction is not in play, it is a very strong result. The findings in the next section support this observation. This reinforces the hypothesis of the similarity of the word cloud structure and thus legitimizes the global approach of the alignment implemented. The performance obtained during the learning process does not seem to be correlated with the performance of the predictions either. We can also ask ourselves whether using several languages during the learning phase is really worthwhile. This seems to be a logical next step for the project, but this remark suggests a lesser increase in terms of efficiency.
- **Impact of Target Language:** There is, however, a strong variation in the BLI criterion depending on the language to be predicted. This point will be studied in more detail later. Yet, this variation seems to be positively correlated with the quality of the alignment of this language with English. Thus, Russian performs less well than German or French in this process. However, these results should be highlighted: the greatest contribution of our method is precisely for Russian when compared to existing benchmarks. We observe a gain of more than 25%, while those for French, Spanish and German are around 10%.
- **Learning step:** The training performance (last columns) seems to depend directly on the quality of the alignment of the language used for learning with English. Figure 8 plots the BLI criterion in the training step according to the CSLS criterion, i.e. the quality of the alignment of the language used for learning with English. The trend that emerges is that of a very clear positive correlation (linear trend plotted in red, $\mathbf{R}^2 = 0.82$). We have also shown the averages per language family (Romance, Germanic and Uralic). In conclusion, it seems easier to learn using a language that is well aligned with English. Although this seems logical, it is not that obvious. Three clusters seem to appear in conjunction with the different families. Romance languages are associated with a high rate of alignment with English and therefore with high performance in the learning stage. The Germanic language cluster has a lower performance combined with a slightly lower quality alignment. Knowing that English belongs to the Germanic language type, it is interesting to note this slight underperformance in alignment compared to Romance. Finally, the Slave cluster shows the worst performance in terms of alignment with English and therefore also the worst for the learning step.

Pivot idiom	EN-ES	EN-FR	EN-DE	EN-RU	*Training*
Romance					
French	95.0	-	92.9	81.0	*92.5 / 80.2*
Italian	95.2	93.6	93.0	82.1	*89.6 / 76.3*
Portuguese	94.8	93.6	92.6	81.1	*91.3 / 81.3*
Spanish	-	93.7	93.1	81.5	*92.4 / 82.1*
Catalan	94.8	93.3	92.4	81.6	*87.0 / 63.4*
Romanian	94.9	93.5	92.1	80.7	*84.4 / 60.0*
Germanic					
Dutch	**95.3**	**93.8**	93.0	81.4	*88.1 / 74.3*
German	94.6	93.0	-	82.7	*90.7 / 70.9*
Norwegian	95.2	93.7	93.0	81.1	*84.7 / 62.4*
Danish	95.0	93.4	93.1	81.6	*85.6 / 63.7*
Swedish	48.2	40.2	67.9	64.4	*- / 0.2*
Slavic					
Russian	95.2	93.6	93.5	-	*76.0 / 42.6*
Ukrainian	94.9	93.2	93.1	83.2	*77.4 / 32.4*
Slovak	94.2	92.5	92.8	81.7	*- / 14.8*
Polish	95.1	93.4	93.5	82.6	*79.1 / 49.1*
Bulgarian	**95.3**	93.5	92.8	82.2	*80.3 / 47.1*
Czech	95.2	93.7	93.5	82.9	*78.9 / 49.7*
Croatian	94.9	93.5	93.1	82.7	*76.6 / 34.1*
Slovenian	94.9	93.4	93.1	82.2	*79.1 / 33.9*
Macedonian	47.4	40.6	68.0	63.4	*- / 0.6*
Others					
Hungarian	95.2	93.7	**93.6**	**83.7**	*79.6 / 48.0*
Estonian	47.6	41.8	68.7	65.2	*- / 0.4*
Greek	95.2	93.5	93.0	82.5	*83.5 / 48.5*
Arabic	94.5	93.4	93.3	82.6	*84.2 / 38.4*
Hebrew	94.7	93.4	93.1	82.3	*78.3 / 42.1*
Indonesian	94.7	93.1	92.7	80.6	*85.4 / 71.7*
Turkish	46.5	38.0	68.2	62.9	*- / 0.3*
Vietnamese	41.5	36.6	66.3	59.2	*- / 0.1*
Benchmark	*84.1*	*83.3*	*79.1*	*57.9*	

Table 3: BLI Results for English Translation using all available idioms

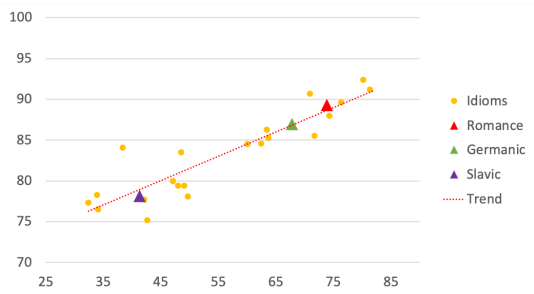

Figure 8: Impact of Alignment Quality on Training.

6.4 Impact of target idiom in inference

Experiment details: Finally, we evaluate the impact of the idiom used for prediction when it comes to translation. We used 4 different languages for the learning step: French, Spanish, German and Russian. The previous section had shown that the language used for learning had little impact on prediction, hence the limited choice of languages. We then try to deduce for more than twenty languages the dictionary with English. Table 4 presents the results of this study. The languages to be predicted are in the left-hand column and those used for learning are in the first row. The last line is the benchmark used. This is again the BLI at the end of the alignment and reflects the quality of the alignment with English.

Mains results: We can make the following observations:

- **Performances:** The algorithm is very efficient and allows to obtain a BLI criterion of at least 80% up to 95%, even more for all languages whereas the previous techniques were much less efficient. RUBI makes it possible to establish a new benchmark of quality to be surpassed in this task, which we hope can become a reference. The minimum gain observed is 10%, which in the literature on the subject is considerable.
- **Efficiency Forecasting:** We wondered whether we could quantify the contribution of our method in relation to the existing benchmark. Figure 9 shows the gains in terms of BLI points (e.g. +20% corresponds to a change from a 60% BLI benchmark to an 80% BLI benchmark) as a function of the quality of the initial alignment, i.e. the benchmark. We observe a very strong negative correlation (linear trend plotted in red, $R^2 = 0.96$). This negative correlation is quite logical: it is easier to have a big BLI gain for languages initially misaligned with English. This line gives access to a first prediction, given the alignment of a language with English on the contribution that our method can give. This prediction seems very stable and relevant. Here again, we observe three clusters: the Romance and Germanic languages have initially a good alignment with English and thus present a relatively weak grain. The Slavic cluster gathers languages that are less well aligned with English and therefore has a lot to gain from our method. It also includes most of the category other languages.
- **Distant idioms:**It should be noted, however, that we did not use some accessible languages with poor alignment with English: Estonian (0.46%), Macedonian (0.58%), Swedish (0.24%), Turkish (0.28%) or even Vietnamese (0.08%). These languages are interesting because they may represent the type of languages for which there is no parallel data with English. However, this is another type of scenario, which deserves to be studied separately. Some parameters need to be adapted for this distinct use case. For example, it is rarer that the correct translation is so close in the aligned space to the query word. It is necessary to look more for each query in the hundred candidates than in the ten, as we do at present. However, this is an exciting avenue to explore in the future.

Learning idiom	Spanish	French	German	Russian	*Bench.*
Romance					
French	93.7	-	93.0	93.6	*80.2*
Italian	91.5	91.8	91.4	91.6	*76.3*
Portuguese	94.0	93.8	93.8	93.9	*81.3*
Spanish	-	95.0	94.6	95.2	*82.1*
Catalan	87.3	87.1	85.3	86.8	*63.4*
Romanian	88.4	88.5	87.7	87.9	*60.0*
Germanic					
Dutch	90.5	90.3	90.1	90.3	*74.3*
German	93.5	92.9	-	93.5	*70.9*
Norwegian	86.1	85.4	86.0	86.6	*62.4*
Danish	89.4	89.7	89.0	89.8	*63.7*
Slavic					
Russian	81.5	81.0	82.7	-	*42.6*
Ukrainian	82.3	80.2	82.7	83.3	*32.4*
Polish	86.2	85.6	86.3	87.3	*49.1*
Bulgarian	84.5	84.3	84.5	85.1	*47.1*
Czech	82.5	82.9	83.6	84.4	*49.7*
Croatian	78.4	77.8	77.9	78.8	*34.1*
Slovenian	80.6	79.8	80.5	82.2	*33.9*
Others					
Hungarian	79.4	79.5	79.6	79.6	*48.0*
Greek	86.0	86.0	86.6	87.3	*48.5*
Arabic	87.1	85.4	87.4	88.6	*38.4*
Hebrew	79.4	79.0	79.6	80.3	*42.1*
Indonesian	90.2	89.9	90.1	89.9	*71.7*

Table 4: BLI Results for all idioms Translation using ES, FR, DE and RU for learning

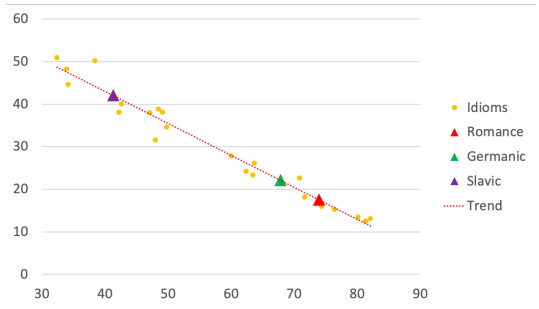

Figure 9: Gain (points of BLI) after the use of RUBI in function of the quality of alignment.

7 CONCLUSION

This paper formulate a new approach to the unsupervised bilingual lexicon induction problem, using learning to rank tools. Our approach, leveraging knowledge of previous idioms acquisition, significantly improves the quality of the induction, outperforming state of the art methodology and setting a new benchmark. As a result, it produces high-quality dictionaries between different pairs of languages, with up to 93.8% on the Spanish-French word translation task. Moreover, we point out the stability of the prediction when the idiom used for learning varies, a strong argument in favor of the similarity of the structure of word embeddings clouds.

REFERENCES

[1] Jean Alaux, Edouard Grave, Marco Cuturi and Armand Joulin: Unsupervised hyperaligment for multilingual word embeddings, 2019
https://arxiv.org/pdf/1811.01124.pdf

[2] Quentin Berthet, Edouard Grave and Armand Joulin: Unsupervised Alignment of Embeddings with Wasserstein Procrustes, 2018
https://arxiv.org/pdf/1805.11222.pdf

[3] Xin Lian: Unsupervised Multilingual Alignment using Wasserstein Barycenter, 2020

[4] Smith, S. L., Turban, D. H., Hamblin, S., and Hammerla, N. Y.: Bilingual word vectors, orthogonal transformations and the inverted softmax., 2017
https://arxiv.org/pdf/1702.03859.pdf

[5] Conneau, A., Lample, G., Ranzato, M., Denoyer, L., and Jégou, H: Word translation without parallel data, 2017
https://arxiv.org/pdf/1710.04087.pdf

[6] A. Lardilleux, F. Yvon, and Y. Et-lepage: Generalizing sampling-based multilingual alignment, Machine Translation, vol.26, issue.2, p.123, 2013

[7] Rama Kumar Pasumarthi, Sebastian Bruch, Michael Bendersky, Xuanhui Wang: Neural Learning to Rank using TensorFlow, ICTIR 2019

[8] Jesus Rodriguez: Introducing TF-Ranking
https://towardsdatascience.com/introducing-tf-ranking-f94433c33ff

[9] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, Stephan Wolf: TF-Ranking: Scalable TensorFlow Library for Learning-to-Rank
https://arxiv.org/pdf/1812.00073.pdf

[10] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, Xueqi Cheng: DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval
https://arxiv.org/pdf/1710.05649.pdf

[11] Olivier Chapelle and Yi Chang. 2011. Yahoo! learning to rank challenge overview. In Proceedings of the Learning to Rank Challenge. 1–24

[12] R. Herbrich, T. Graepel, and KR. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In Smola, Bartlett, Schoelkopf, and Schuurmans, editors, Advances in Large Margin Classifiers. MIT Press, Cambridge, MA, 2000.

[13] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In Proceedings of the International Conference on Machine Learning, 2005.

[14] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In Advances in Neural Information Processing Systems 20, pages 1697–1704. MIT Press, 2008.

[15] Z. Cao, T. Qin, T-Y. Liu, M-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In International Conference on Machine Learning, 2007.

[16] Z. Cao, T. Qin, T-Y. Liu, M-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In International Conference on Machine Learning, 2007.

[17] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007.

[18] M. Taylor, J. Guiver, S. Robertson, and T. Minka. SoftRank: optimizing non-smooth rank metrics. In WSDM '08: Proceedings of the international conference on Web search and web data mining, pages 77–86. ACM, 2008.

[19] Thorsten Joachims. 2002. Optimizing Search Engines Using Clickthrough Data. In 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 133–142.

[20] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 391–398.

[21] Christopher J.C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. Technical Report Technical Report MSR-TR-2010-82. Microsoft Research.

[22] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. Annals of Statistics 29, 5 (2001), 1189–1232.

[23] Thorsten Joachims. 2006. Training linear SVMs in linear time. In 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 217–226.

[24] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In 10th ACM International Conference on Web Search and Data Mining. 781–789

[25] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In 22nd International Conference on Machine Learning. 89–96.

[26] Ilya Sutskever, Thomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean 2013. Distributed Representations of Words and Phrases and their Compositionality

[27] Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou and Edouard Grave 2018. Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion

[28] Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. Improving zero-shot learning by mitigating the hubness problem. International Conference on Learning Representations, Workshop Track, 2015.

[29] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. Advances in neural information processing systems, pp. 2672–2680, 2014.

[30] David Alvarez-Melis and Tommi Jaakkola. Gromov-wasserstein alignment of word embedding spaces. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018.

[31] Mikhail Gromov. Metric structures for Riemannian and non-Riemannian spaces. Springer Science Business Media, 2007.

[32] Cedric Villani. Topics in optimal transportation. Number 58. American Mathematical Soc., 2003.

A ONLINE SOFTWARE RESOURCES

To make the discussed results useful and reproducible, our code and the software resources used are freely available online.

- The code, main examples and the files used for tables and graphs are accessible at https://github.com/Gguinet/semisupervised-alignement.git
- It is still a private directory, however, and we are currently reformatting the code so that it can be made publicly available in the coming days. All you have to do is ask us for permission to add you to it if you can't access it.
- All data used (word embeddings and dictionary) are coming from Facebook public files on the topic. A part of it can be found using the link https://fasttext.cc.
- As the simulations were very demanding in term of computed power, we used Google Compute Engine, with the following settings: 8 virtual processors, n1-highmem-8, high memory capacity, 500 Go of memory, Ubuntu, Version 18.04. More information on how to use it is on the github file of the project.